Evaluation of optimal solutions in multicriteria models for intelligent decision support

Author:
Aarón López García

Supervisor:
Dr. Rafael Benítez Suárez

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor from the Universitat de València

Departamento de Matemáticas para la Economía y la Empresa
Programa de doctorado en Estadística y Optimización
Facultad de Ciencias Matemáticas

Universitat de València

April 2023

Evaluation of optimal solutions in multicriteria models for intelligent decision support ©
April, 2023

Author:
Aarón López García

Supervisor:
Dr. Rafael Benítez Suárez

Institution:
Universitat de València

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SCRIPTS

LIST OF ALGORITHMS

ABSTRACT

The work is framed within optimization and its use for decision making. The logical sequence has been modeling, implementation, resolution, and validation leading to a decision. For this, we have used tools of Multi-Criteria Analysis, Multi-Objective Optimization, and Artificial Intelligence techniques.

The work has been structured in two parts (divided into three chapters each) corresponding to the theoretical and experimental parts. The first part analyzes the context of the field of study with an analysis of the historical background, followed by a chapter on Multi-Criteria Optimization in which known models are included, together with original contributions to this work. In the third chapter, dedicated to Artificial Intelligence, the fundamentals of Statistical Learning, Machine Learning, and Deep Learning techniques necessary for the contributions in the second part are presented.

The second part contains seven real cases to which the described techniques have been applied. In the first chapter, two cases are studied: the academic performance of students at the Universidad Industrial de Santander (Colombia) and an objective system for the selection of the MVP award in the NBA. In the next chapter, Artificial Intelligence techniques are used for musical similarity (plagiarism detection in Youtube), the prediction of the closing price of a company in the New York stock market, and the automatic classification of acoustic immersive spatial audio. In the last chapter, Multiple-Criteria Decision Making techniques are incorporated to the power of Artificial Intelligence to detect university school failure early (at the Industrial University of Santander), and Multiple-Criteria Decision Making methods are used to establish a ranking of Artificial Intelligence models.

To conclude the work, although each chapter contains a partial conclusion, Chapter 8 contains the main conclusions of the entire report and a fairly exhaustive bibliography of the topics covered. In addition, the work concludes with three appendices containing the programs and tools, which, although useful for the understanding of the report, we have preferred to put them separately to make the chapters more fluid.

RESUMEN

La memoria se enmarca dentro de la optimización y su uso para la toma de decisiones. La secuencia lógica ha sido la modelación, implementación, resolución y validación que conducen a una decisión. Para esto, hemos utilizado herramientas del análisis multicrerio, optimización multiobjetivo y técnicas de inteligencia artificial.

El trabajo se ha estructurado en dos partes (divididas en tres capítulos cada una) que se corresponden con la parte teórica y con la parte experimental. En la primera parte se analiza el contexto del campo de estudio con un análisis del marco histórico y posteriormente se dedica un capítulo a la optimización multicriterio en el se recogen modelos conocidos, junto con aportaciones originales de este trabajo. En el tercer capítulo, dedicado a la inteligencia artificial, se presentan los fundamentos del aprendizaje estadístico , las técnicas de aprendizaje automático y de aprendizaje profundo necesarias para las aportaciones en la segunda parte.

La segunda parte contiene siete casos reales a los que se han aplicado las técnicas descritas. En el primer capítulo se estudian dos casos: el rendimiento académico de los estudiantes de la Universidad Industrial de Santander (Colombia) y un sistema objetivo para la asignación del premio MVP en la NBA. En el siguiente capítulo se utilizan técnicas de inteligencia artificial a la similitud musical (detección de plagios en Youtube), la predicción del precio de cierre de una empresa en el mercado bursátil de Nueva York y la clasificación automática de señales espaciales acústicas en entornos envolventes. En el último capítulo a la potencia de la inteligencia artificial se le incorporan técnicas de análisis multicriterio para detectar el fracaso escolar universitario de manera precoz (en la Universidad Industrial de Santander) y, para establecer un ranking de modelos de inteligencia artificial de se recurre a métodos multicriterio.

Para acabar la memoria, a pesar de que cada capítulo contiene una conclusión parcial, en el capítulo 8 se recogen las principales conclusiones de toda la memoria y una bibliografía bastante exhaustiva de los temas tratados. Además, el trabajo concluye con tres apéndices que contienen los programas y herramientas, que a pesar de ser útiles para la comprensión de la memoria, se ha preferido poner por separado para que los capítulos resulten más fluidos.

RESUM

La memòria s'emmarca dins de l'optimització i el seu ús per a la presa de decisions. La seqüència lògica ha sigut el modelatge, la implementació, la resolució i la validació que condueixen a una decisió. Per això, hem utilitzat eines de l'anàlisi multicreri, l'optimització multiobjectiu i tècniques d'intel·ligència artificial.

El treball s'ha estructurat en dues parts (dividides en tres capítols cadascuna) que es corresponen amb la part teòrica i amb la part experimental. En la primera part s'analitza el context del camp d'estudi amb una anàlisi del marc històric i posteriorment es dedica el capítol 3 a l'optimització multicriteri en el qual es recullen models coneguts, juntament amb aportacions originals d'aquest treball. En el quart capítol, dedicat a la intel·ligència artificial, es presenten els fonaments de l'aprenentatge estadístic, les tècniques d'aprenentatge automàtic i d'aprenentatge profund necessàries per a les aplicacions.

La segona part conté set casos reals als quals s'han aplicat les tècniques descrites. En el capítol cinquè s'estudien dos casos: el rendiment acadèmic dels estudiants de la Universidad Industrial de Santander (Colòmbia) i un sistema objectiu per a l'assignació del premi MVP en la NBA. En el capítol 6 s'utilitzen tècniques d'intel·ligència artificial a la similitud musical (detecció de plagis en Youtube), la predicció del preu de tancament d'una empresa en el mercat de valors de Nova York i la classificació automàtica de senyals espacials acústiques en entorns envolupants. En el capítol setè, a la potència de la intel·ligència artificial se li incorporen tècniques multicriteri per a detectar el fracàs escolar universitari de manera precoç (a la Universidad Industrial de Santander) i, per a establir un rànquing de models d'intel·ligència artificial es recorre a mètodes d'optimització multicriteri.

Per a acabar la memòria, a pesar que cada capítol conté una conclusió parcial, en el capítol 8 es recullen les principals conclusions de tota la memòria i una bibliografia prou exhaustiva dels temes tractats. A més, el treball conclou amb tres apèndixs que contenen els programes i eines, que malgrat ser útils per a la comprensió de la memòria, s'ha preferit posar per separat perquè els capítols resulten més fluids.

STATEMENT OF AUTHORSHIP

I, Aarón López García, born September 6, 1995, in Valencia, declare that this thesis titled *Evaluation of optimal solutions in multicriteria models for intelligent decision support* and the work presented in it are my own. I confirm that this work was done mainly while in candidature for a research degree at Universitat de València.

Except where reference is made in the text of the thesis, this thesis contains no material published elsewhere or extracted in whole or in part from a thesis accepted for the award of any other degree or diploma. No other person's or research team's work has been used without due acknowledgment in the main text of the thesis. This thesis has not been submitted for the award of any degree or diploma in any other tertiary institution.

Aarón López García

Rafael Benítez Suárez

April 20, 2023

DEDICATION

A mis padres

ACKNOWLEDGMENTS

Durante mi etapa como estudiante de doctorado, he tenido el placer de colaborar con personas maravillosas que, de una manera u otra, han hecho que saque lo mejor de mí. Me siento sumamente afortunado por la calidad humana con la que he tenido el gusto de trabajar. Por tanto, esta sección está dedicada a todos ellos.

En primer lugar, me gustaría agradecer a mi tutor Dr. Rafael Benítez Suárez por toda la ayuda brindada durante estos últimos años. En 2019 tuve la suerte de tenerte como tutor de trabajo de fin de máster, donde despertaste mi interés por la programación y la matemática aplicada. En esa etapa tan complicada de mi vida, hiciste que pareciera fácil compaginar una jornada laboral junto con el duro (y poco reconocido) trabajo de investigación que hacíamos. De todos modos, no había problema que no se solucionase con un café y una napolitana de chocolate. Una vez superado el confinamiento que supuso la pandemia, en noviembre de 2020 decidimos empezar esta tesis con ciertas dudas, pero muchísima ilusión. Ahora, después de dos años investigando codo con codo, esta memoria es el resultado de nuestro trabajo.

Otra maravillosa persona a la que le debo el 100% de mi vida investigadora es el Excmo. Dr. Vicente Liern Carrión. Él fue quien me introdujo en este mundo y quien me ha respaldado durante todos estos años, confiando plenamente en mi potencial y en mis ideas. Siendo sincero, me sería imposible redactar en un párrafo todo lo que has hecho por mí. Por si fuera poco todo tu apoyo profesional, como persona me has ayudado aún más. Sin ti esta tesis no hubiese sido posible. Estaré eternamente agradecido contigo y con tu arroz con pollo.

Recordando la parte de divulgación y publicación, me gustaría destacar la gran labor que ha hecho la Dra. Olga Blasco Blasco conmigo. Gracias a ti publiqué mis primeros artículos y congresos, creando un entorno de trabajo súper acogedor. A día de hoy, no sé qué sería de mi trayectoria como investigador sin tu colaboración. Muchas gracias por orientarme y aguantarme durante esta etapa de mi vida.

Durante estos dos años como estudiante de doctorado, he tenido el placer de trabajar junto con el Dr. Máximo Cobos Serrano. Me gustaría agradecerte todo lo que has hecho por mí y por la confianza que me has dado durante el año y medio que hemos trabajado juntos. Además de darme la oportunidad de colaborar en un gran equipo, también despertaste mi interés por el mundo de la informática y la inteligencia artificial, cosa con la que no puedo estar más agradecido. He disfrutado muchísimo todo este tiempo en la ETSE y tú eres a la principal persona a la que darle las gracias por ello.

Un dato clave que me no me gustaría pasar por alto es que este trabajo no tendría sentido de no ser por mi amor Marina Liern García. Si no fuera por tu apoyo incondicional, jamás me habría apuntado al curso de doctorado. Aunque de aquellas a mí me parecía una locura,

queda claro que ha sido posible y tú eres la principal persona a la que agradecérselo. En gran parte, esta tesis es por ti. Te quiero.

Por último, agradecer la Universidad Industrial de Santander y en especial a la Dra. Sandra Evely Parada Rico por toda la transferencia de información y por su gran detalle al ofrecerme una estancia en su país. Desde entonces, sigo enamorado de Colombia y de su gente.

PUBLICATIONS

During the drafting of this thesis, there have been a series of works and collaborations in research projects that have been published or accepted for publication. Such contributions have favored the development of this thesis whether directly or indirectly. For this reason, this chapter is entirely devoted to the presentation of the different publications that we consider important and/or relevant for the writing of this work. All the publications listed below are the own work of the author of this thesis, who has the permission of the corresponding contributors to share and reproduce the contents of these publications for academic purposes.

The author's contributions that have been published before the submission of this thesis are divided into papers and software repositories. Concerning the authorship and permissions, the author has full acceptance by the other participants to include it in the thesis.

PAPERS: The published works that have helped either in the preparation of this thesis or during the process of solving any case study are listed here.

- Blasco-Blasco, O., Parada-Rico, S.E., Liern-García, M., López-García, A. *Characterization of university students through indicators of adequacy and excellence. Analysis from gender and socioeconomic status perspective*. In ICERI2020 Proceedings (pp. 8030-8037). IATED, DOI: 10.21125/iceri.2020.1780 (2020) Blasco-Blasco et al., 2020.

- Blasco-Blasco O., Liern-García M., López-García A., Parada-Rico SE., "*An Academic Performance Indicator Using Flexible Multi-Criteria Methods*", Mathematics, 2021; 9(19):2396, DOI: 10.3390/math9192396, Blasco-Blasco et al., 2021.

- Naranjo-Alcazar J., Pérez-Castaños S., López-García A., Zuccarello P., Cobos M., Ferri F.J., "*Squeeze-Excitation Convolutional Recurrent Neural Networks for Audio-Visual Scene Classification*", arXiv preprint arXiv:2107.13180, (2021). DOI: 10.48550/ARXIV.2107.13180 DCASE2021 link. Naranjo-Alcazar et al., 2021.

- López-García A., Martínez-Rodríguez B., Liern V., "*A Proposal to Compare the Similarity Between Musical Products. One More Step for Automated Plagiarism Detection?*". In International Conference on Mathematics and Computation in Music (pp. 192-204). Springer, Cham, 2021, ISBN: 978-3-031-07015-0, DOI: 10.1007/978-3-031-07015-0_16, López-García et al., 2022.

- López-García, A., Benítez, R. "*Reputation analysis of news sources in Twitter: Particular case of Spanish presidential election in 2019*". arXiv preprint arXiv:2201.08609, (2022). DOI: 10.48550/ARXIV.2201.08609. López-García and Benítez, 2022.

- De Rus J.A., López-García A., López-Ballester J., Lopez J.J., Torres A.M., Ferri F.J., Montagud M., Cobos M., "*On the Application of Explainable Artificial Intelligence Techniques on HRTF Data*". In 24th International Conference on Acoustics, ICA 2022.

- López-García, A. "*SpectroMap: Peak detection algorithm for audio fingerprinting.*" arXiv preprint arXiv:2211.00982 (2022). DOI: 10.48550/ARXIV.2211.00982. López-García, 2022.

PER-REVIEW WORKS: Owing to the constant implementation of methodology that concerns the content of this thesis, some parts of the work are not published yet. However, the following works have been submitted and are pending review by the editorials.

- López-García A., Blasco-Blasco O., Liern-García M., Parada-Rico S.E., "*Early detection of students' failure using Machine Learning techniques*", submitted on July 27, 2022, to the Expert Systems with Applications journal.

- López-García, A., Benítez, R. "*uwVIKOR: An Unweighted Multi-Criteria Decision Making Approach for Compromise Solution*", submitted on September 29, 2022, to the Expert Systems with Applications journal.

SOFTWARE: Here is the list of all the open-source repositories under my authorship that have been applied in any particular case regarding this thesis. Additional information can be found in my GitHub profile.

- SpectroMap: Peak detection algorithm that computes the constellation map for a given signal. First release: 26-02-2022. GitHub repository: https://github.com/Aaron-AALG/spectromap. PyPI repository https://pypi.org/project/spectromap/ López-García, 2022b.

- GPMM: Collection of generalized *p*-mean models with classic, fuzzy and unweighted approach . First release: 08-08-2022. GitHub repository: https://github.com/Aaron-AALG/GPMM. PyPI repository https://pypi.org/project/GPMM/ López-García, 2022a.

- uwVIKOR: Unweighted VIKOR method. First release: 01-07-2022. GitHub repository: https://github.com/Aaron-AALG/GPMM. PyPI repository https://pypi.org/project/GPMM/ López-García, 2021b.

- uwTOPSIS: Unweighted TOPSIS method. First release: 16-06-2021. GitHub repository: https://github.com/Aaron-AALG/uwVIKOR. PyPI repository https://pypi.org/project/uwVIKOR/ López-García, 2021a.

Except where reference is made throughout the thesis, this work contains no other material published elsewhere or extracted in whole or in part from a thesis accepted for the award of any other degree or diploma. No other person's work has been used without due acknowledgment in the main text of the thesis. This thesis has not been submitted for the award of any degree or diploma in any other tertiary institution.

ACRONYMS

AI	Artificial Intelligence
ANN	Artificial Neural Network
BA	Business Analytics
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DT	Decision Tree
FNN	Feed-forward Neural Network
GBM	Gradient Boosting Machine
KNN	K-Nearest Neighbours
LDA	Linear Discriminant Analysis
LSTM	Long Short-Term Memory
MAUT/MAVT	Multi-Attribute Utility/Value Theory
MCDA	Multiple-Criteria Decision Analysis
MCDM	Multiple-Criteria Decision Making
ML	Machine Learning
MLP	Multi-Layer Perceptron
NN	Neural Network
OR	Operations Research
PCA	Principal Component Analysis
PDA	Preference Disaggregation Analysis
RF	Random Forest
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TOPSIS	Technique For Order Preference By Similarity To Ideal Solution
VIKOR	VIseKriterijumska Optimizacija I Kompromisno Resenje (Multicriteria Optimization And Compromise Solution)
WMM	Weighted Mean Models
WPM	Weighted Product Models
WSM	Weighted Sum Models
XAI	EXplainable Artificial Intelligence
XGBoost	Extreme Gradient Boosting

1 INTRODUCTION

Business and technological developments are evolving at inordinate levels, where the uncertainty of the future and the cyber-connected world generate a rather challenging scenario. World economies are also undergoing major transformations to adapt to this type of change. Even though last technological and research advances provide us with tools that help institutions manage their countries' politics, implementing these techniques does not turn out to be easy at all. For this reason, the field study of decision analysis has become an attractive field of study from a mathematical point of view, not only for solving the decision stage but also for offering efficient and easy-to-use methodologies. A clear example of this concern is the modern agendas for environmental and sustainable development, where the plan of action for people, planet, and prosperity is clearly defined. Still, the manner to achieve such goals is not established and/or is not globally possible.

Since the global business environment directly impacts all socioeconomic and technological policies, implementing enhanced tools and procedures is required for support in such a challenging task. In most cases, the target is broadly known. However, the priorities, tradeoffs, and complications in this regard remain unclear, thus complicating the decision process. Decision analysis is the discipline that encompasses the theory, methodology, and professional practices needed for addressing a decision-making problem. This subject was first studied in the 1940s by the mathematician John von Neumann and economist Oskar Morgenstern, who established the basis for utility theory. They defined the principles of game theory and mathematically represented a cooperative framework in which n decision-makers took actions considering their individual preferences over uncertain outcomes. Over the years, this branch of Operations Research considerably evolved, offering us a multitude of methods for decision aid.

Nowadays, two fields have attracted much interest regarding decision analysis. On the one hand, Multiple-Criteria Decision Making (MCDM) is a major discipline in Operations Research that provides extensive support to decision-makers in complex, ill-structured problems involving conflicting criteria. MCDM has techniques designed to attach every single aspect of the decision process, from the problem structuring/modeling to the selection of the final decision. On the other hand, Artificial Intelligence (AI) covers the theory and development of computer systems able to solve complex tasks characterized by being usually performed by humans. In this regard, the fields of computer vision, speech recognition, and language translation are understood, among others.

The two fields of study above-mentioned represent most of the methodology studied in this thesis. Then, Chapter 3 describes the methodology of Multiple-Criteria Decision

Making understood from different approaches, and Chapter 4 contains the theory related to statistical learning and the principles required to build intelligent systems. Finally, several case studies are presented as the experimental part of this thesis to show real-world applications with new contributions to both fields.

1.1 OBJECTIVES

This work aims to achieve the following objectives:

1. Combine the fields of study of Multiple-Criteria Decision Analysis and Artificial Intelligence for decision support.
2. Study the new field of research of the Unweighted Multiple-Criteria Decision Making.
3. Analyze the resultant optimal solutions of own-developed decision-making techniques in real-world environments.

1.2 CONTRIBUTIONS

Our main contributions include:

1. Design and implementation of ranking techniques from the emerging field of study of Unweighted Multiple-Criteria Decision Making.
2. Incorporation of the new field of Fuzzy-Unweighted Multiple-Criteria Decision Making together with a real-life application.
3. Application of customized Artificial Intelligence models for solving highly complex problems and analyzing the corresponding optimal solution.
4. Combination of the fields of Multiple-Criteria Decision Making and Artificial Intelligence in two case studies with the aim of offering new tools in the area of decision analysis.
5. Python implementation of the unweighted version of TOPSIS and VIKOR, the fuzzy-unweighted version of the Weighted Mean Models, and an audio fingerprinting algorithm named SpectroMap. (Published as GitHub repositories in my page Aaron-AALG).

1.3 THESIS OUTLINE

The remainder of this thesis is organized as follows:

Part i : Theory and Methodology

Chapter 2 provides the basis of our research. It introduces the four fields of knowledge necessary for the development of this thesis, together with their historical background.

Chapter 3 contains the methodology related to Multiple-Criteria Decision Analysis and Decision Making discussed in this thesis. It introduces the motivation and branches of this field. The theoretical background developed for application purposes is presented in three parts corresponding to the three approaches: classic, fuzzy, and unweighted.

Chapter 4 contains the methodology related to Artificial Intelligence techniques developed in this thesis. In addition to focusing on practical aspects, this chapter describes various mathematical concepts and statistical notions that build the underlying functioning of this field. The theory is divided into Machine Learning and Deep Learning areas to distinguish the complexity of this area of knowledge.

Part ii : Experimental Study

Chapter 5 shows the applicability of the unweighted versions of the decision-analysis techniques in two particular real-world applications. The first section analyzes the unweighted TOPSIS, and the second section introduces the unweighted VIKOR technique.

Chapter 6 contains a total of three case studies from the Artificial Intelligence discipline. Different Machine Learning topics have been studied in each section. The main techniques used have been, firstly, an unsupervised learning technique and, secondly, two supervised learning problems focusing on classification and regression, respectively.

Chapter 7 introduces the combination of the Multiple-Criteria Decision Making and Artificial Intelligence approaches for solving two case studies. On the one hand, a Multiple-Criteria Decision Making technique is utilized as a feature extractor in a classification problem. On the other hand, an integrated end-to-end system is presented for benchmarking Machine Learning models with Multiple-Criteria Decision Making.

Chapter 8 completes the thesis remarking on the relevant points discussed during this work and the main contributions obtained thanks to our research. Moreover, we introduce some interesting topics that were not fully covered in this thesis. We finish the chapter discussing potential future research areas related to the field of statistics and operations research.

Finally, Appendices A, B, and C contain the scripts, routines, and programming strategies conducted for this work. Each appendix has the required code to carry out the experiments and/or case study presented throughout the thesis.

Part I

THEORY AND METHODOLOGY

2 HISTORICAL BACKGROUND

Most of the theory required for decision making is highly related to multidisciplinary knowledge. Both applicability and methodology have been developed by combining multiple approaches. For this reason, we want to emphasize every single aspect that concerns the state-of-the-art up to the prescriptive stage. Then, we have decided to divide this chapter into four sections that primarily build the underlying methodology of decision theory. Each chapter contains the historical background involved in the main fields of statistics and probability, operations research, computer science, and business analytics.

2.1 STATISTICS AND PROBABILITY

Statistics and probability are two branches of mathematics that generate a strong synergy for solving problems in multiple real-life environments. Despite the fact that both disciplines are widely combined in mathematical applications, it is necessary to understand the main difference between them. Probability is fundamentally based on measure theory and it aims to quantify the likelihood of an event occurring. Statistics relies on the applicability of data analysis for inference purposes, hence it concerns the random sampling of data for the interpretation and presentation of the obtained results. Once we know their definition, we can realize why statistics and probability are simultaneously utilized for data analysis. In this section, we will present the developments of them altogether since most of the advancements occurred when mathematicians combined both disciplines at once.

Over the years, humans have attempted to measure the likelihood of a certain event taking place. Although the beginning of this study was completely experimental, empiricism has helped to develop further analysis. Due to several fields utilizing statistics as the main data-gathering source, we can find examples of its application in ancient times. The most representative case is the study of the census for controlling and studying the population's behavior. The oldest sample was discovered in India with an approximated origin in 330 B.C, and the oldest preserved was held in China in AD 2 with high accuracy in its calculations (Bowman, 2000). However, it took a long time until 1663 John Graunt and William Petty developed the foundations of demography. In regard to probability, the study of permutations and combinations has been a common interest in various cultures. On the one hand, cryptography took a central role in research during the Islamic Golden Age. In the 8th century, Al-Khalil wrote about the combinations of all possible Arabic words, and subsequently, in the 9th century, Al-Kindi made the earliest known implementation of frequency analysis and statistical inference for encryption-decoding techniques. On the other hand, gambling has been a standard when referring to probability because the process of quantifying success-failure rates helped to set the basis of this science. In the 16th century, Gerolamo Cardano wrote his *Liber de ludo aleae*, in which he analyzed various games of chance by incorporating odds and calculus of probabilities (Ore, 1953).

The year 1654 is globally accepted as the birth of probability theory due to the correspondence established between Pierre Fermat and Blaise Pascal (Fienberg, 1992). They originally discussed a gambling problem, subsequently called the problem of points or the problem of the stakes. They reasoned it using the expected value, albeit they did not prove it with sufficient rigor. It is considered a turning point for the field of probability since, up to this date, the term probable just indicated the general approval of some event based on past events. Shortly thereafter, Christiaan Huygens published in 1657 the first books on probability theory, defining important concepts as the expectation value (Hacking, 2006).

In the 18th century, there was important mathematical progress in the theory of probabilities. Such remarking point is due to the mathematical approach mainly given by two authors, Jacob Bernoulli, who wrote *Ars Conjectandi* in 1713, and Abraham de Moivre, who wrote *Doctrine of Chances* in 1718. A decisive contribution was the representation of event probability p as a number between 0 and 1, indicating the odds and the concept of null and absolute certitude. The definition of a Bernoulli trial as a random event with output based on a dichotomy with the same probability generated the so-called Bernoulli process with binomial (or Bernouilli) distribution. Such distribution laid out a discrete-time stochastic process, marking a turning point in the foundations of probability. In the middle of the century, Thomas Bayes developed the so-called Bayes' Theorem, although he formulated a particular case, and later Laplace stated it as the actual version. For two studied events A and B with non-zero probability, it gives us a manner to get the conditional probability associated with the events. It can be mathematically stated as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{2.1}$$

Thanks to his contributions, nowadays, there exists a branch of probability called Bayesian probability, which interprets probability as an expectation of degrees of certainty, contrary to the frequentist approach (Howson and Urbach, 2006). In other words, the concept of probability can be understood via logical statements where the spectrum of truth values is unknown a priori. At the end of the century, Pierre-Simon Laplace introduced two laws of error for measuring the uncertainty associated with some studied processes, which followed the theory of errors discussed by Roger Cotes and Thomas Simpson. The laws stated that the frequency of an error could be expressed as an exponential function of the numerical magnitude of the error (1st law), and such error representation is the exponential function of the square of the error (2nd law) (Wilson, 1923).

In the 19th century, the development of the least squares method for applications in regression tasks made a breakthrough in the field of statistical modeling. It was first published and formally stated by Adrien-Marie Legendre in 1805 as a fitting problem. However, the major credit is associated with Carl Friedrich Gauss due to his contribution to model the orbit of Ceres, made in 1795 but published in 1809 (Stigler, 1981). In regard to theoretical advances, three chief findings deserve additional emphasis. Firstly, Pierre Simon Laplace laid the basis of modern probability and statistics in his *Théorie analytique des probabilités* in 1812, in which he rigorously developed several results that are crucial nowadays as the exponential distributions or hypothesis testing. Secondly, Siméon Denis Poisson formalized the law of large numbers (LLN), which was previously approached by Bernoulli and subsequently reformulated by relevant mathematicians such as Khinchin and Kolmogorov (Seneta, 2013). LLN has two different versions that rely on mathematical analysis convergence concepts that lead to weak and strong laws. Both versions states that for an infinite sequence $\{X_i\}_{i\geq 1}$ of independent and identically distributed (i.i.d.) Lebesgue integrable random variables such that $\mathbb{E}[X_i] = \mu$ per each $i \geq 1$, then the sample average $\bar{X}_k = \frac{1}{k}\sum_{i=1}^{k} X_i$ also converges to μ. The two laws can be mathematically written as in Hsu and Robbins (1947):

$$\text{Weak law} \quad \lim_{k\to+\infty} P\left[|\bar{X}_k - \mu| < \varepsilon\right] = 1, \quad \text{per each } \varepsilon > 0. \tag{2.2}$$

$$\text{Strong law} \quad P\left[\lim_{k\to+\infty} \bar{X}_k = \mu\right] = 1. \tag{2.3}$$

The third key contribution was the concept of normal distribution or Gaussian distribution, named after Karl Gauss for his publishing *Theoria combinationis observationum erroribus minimis obnoxiae*, although de Moivre and Laplace already developed other underlying ideas. Its general formulation is $N(\mu, \sigma^2)$ for a given by a mean value μ and a standard deviation σ, with a probability density function of:

$$f(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{2.4}$$

Whose graphical representation yields the output illustrated in Fig. 2.1.

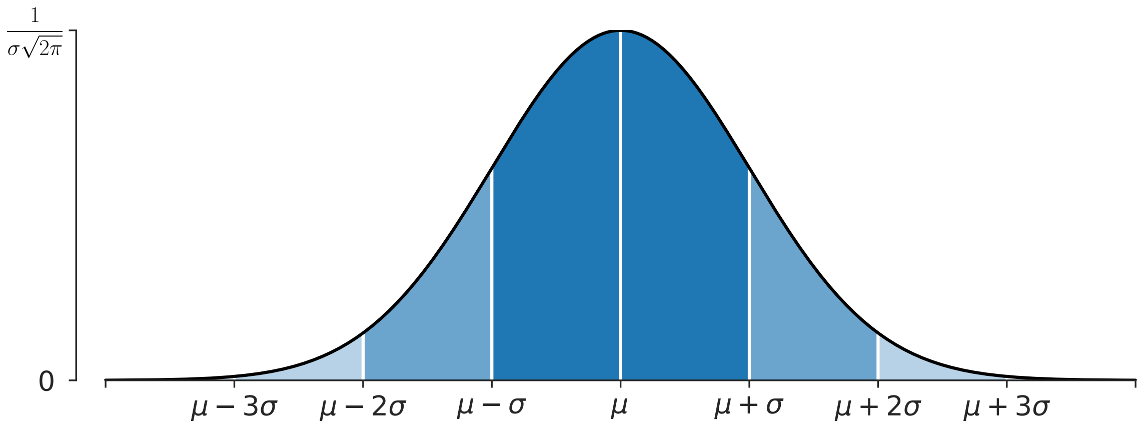


Figure 2.1: Normal distribution $N(\mu, \sigma^2)$ centered in μ and divided by deviation intervals. Source: Own elaboration.

An interesting point that we would like to emphasize about the normal distribution is the resultant coverage probability between $\mu \pm k\sigma$ intervals. Table 2.1 shows the accounting values attached.

Table 2.1: Coverage probability of the normal distribution computed by intervals and truncated to the 4th decimal place.

Interval	Probability	Complementary
$\mu \pm \sigma$	0.6826	0.3174
$\mu \pm 2\sigma$	0.9544	0.0455
$\mu \pm 3\sigma$	0.9973	0.0027
$\mu \pm 4\sigma$	0.9999	0.0001

We can say that the 18th century got an impressive development in the area of statistical theory. In fact, a statistical framework was developed with the aim of describing the randomness of physical events with probability. This field was called statistical mechanics, and the precursors were Ludwig Boltzmann and Willard Gibbs. There were other multiple mathematicians that contributed to the applied field of statistics. Among many others is essential to mention Bessel, Lacroix, Laurent, Cournot, Peirce, Quetelet, and De Morgan.

At the end of the 18th century, the definition of likelihood function by Thorvald N. Thiele (1880) and correlation by Francis Galton (1888) meant a remarkable step towards modern statistical concepts. Moreover, Galton proposed using linear regressions to perform statistical modeling in various applied fields. Before presenting the following century, it is necessary to highlight the important contributions made so far in the subfield of data visualization. Despite the fact it does not depend on scientific background, it makes it easier to follow some experiments by means of charts that visually summarize the results. Among many contributors, we would like to mention John Venn, who designed the concept Venn diagram, and Florence Nightingale, a pioneer in applied statistics for sanitary issues and developer of polar area diagrams.

In the 20th century, the mathematical community aimed to establish the foundations of probability theory by means of rigorous axiomatization. It took a long time until Andrey Kolmogorov defined three basic axioms by using the measure theory as a reference. Later, he wrote *Foundations of the theory of probability* (Kolmogorov, 1950) in 1933, in which all these concepts were formalized. In general, let (Ω, F, P) be a measure space; we say such space is a probability space with sample space Ω, event space F, and probability measure P as long as the following axioms hold true:

1st axiom (Non-negativity): Each event $E \in F$ has associated a non-negative measure, i. e. $P(E) \geq 0$.

2nd axiom (Unit measure): The entire sample space has probability one, i. e. $P(\Omega) = 1$.

3rd axiom (σ-additivity): Given a countable mutually exclusive sequence $\{E_i\}_{i=1}^{\infty}$ of events in F, then:

$$P\left(\bigcup_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty} P(E_i). \tag{2.5}$$

Following the same line as Kolmogorov, in 1946 Richard Cox develop an alternative way to formulate probability theory by means of postulates that helped to develop their famous Cox theorem (Van Horn, 2003). His work was motivated by logical theory where he discussed the use of degrees of plausibility with a consistent approach.

Apart from the mathematical axiomatization of probability theory, there was a fundamental contribution to applied statistics thanks to the introduction of the central limit theorem (CLT). It was first referenced in 1920 by Pólya, who was motivated by past Laplace's work, but its general formalization was finally drawn up in 1935 with the contributions of Lindeberg, Lévy, and Bernshtein (Fischer, 2011). The different versions of CLT state the converge on probability distributions of functions of an increasing number of one or multi-dimensional random variables (or abstract elements) to a normal distribution. Classic CLT states that, let $\{X_i\}_{i\geq 1}$ be i.i.d random variables so that $\mathbb{E}[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2$ per each $i \geq 1$, then $\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$ satisfies that:

$$\text{Classic CLT:} \quad \lim_{n\to+\infty} \frac{\bar{X}_n - \mu}{\frac{\sigma^2}{\sqrt{n}}} \sim N(0,1). \tag{2.6}$$

For later versions, it is considered the series $s_n^2 = \sum_{i=1}^{n} \sigma_n^2$ to guarantee stronger conditions. If for every $\delta > 0$ such conditions hold true:

$$\text{Lyapunov:} \quad \lim_{n\to+\infty} \frac{1}{s_n^{2+\delta}} \sum_{i=1}^{n} \mathbb{E}\left[|X_i - \mu_i|^{2+\delta}\right] = 0. \tag{2.7}$$

$$\text{Lindeberg:} \quad \lim_{n\to+\infty} \frac{1}{s_n^2} \sum_{i=1}^{n} \mathbb{E}\left[(X_i - \mu_i)^2 \mathbf{1}_{\{X_i : |X_i - \mu_i| > \delta s_n\}}\right] = 0. \tag{2.8}$$

Then, $\lim_{n\to+\infty} \frac{1}{s_n} \sum_{i=1}^{n} (X_i - \mu_i) \sim N(0,1)$.

Before CLT was stated, statistical tests were hard to compute since it is difficult to find appropriate conditions for the approximation of random variables via Gaussian distributions (Cam, 1986). Nowadays, the assumption that a sample data has fixed parameters that determine its probability distribution gives parametric tests higher statistical power because of the CLT (Kwak and Kim, 2017).

During the first half of the 20th century, we can highlight the contributions in probability and statistics from Michel Plancherel (ergodic theory), Andrey Markov (Markov process), Louis Bachelier (mathematical finance via stochastic process), Ronald Fisher (foundations for modern statistical science), Karl Pearson (mathematical statistics), Jerzy Neyman (confidence interval for testing), William Gosset (t-student test) and P. C. Mahalanobis (statistical distance).

From the second half of the 20th century, the interest in statistics and probability increased due to the incorporation of modern computers, since they reduced the computational times. As a result, the field of artificial intelligence grew for the incorporation of statistical theory in real-life applications, giving customized responses to complex tasks. Then, the field of statistical learning was founded, with the cases of machine learning (for known responses) and data mining (for unknown outcomes). Both the historical background and the foundations are explained in Chapter 4 with further cases and examples.

2.2 OPERATIONS RESEARCH

Operations Research (OR) is a subfield of Mathematics that encompasses the disciplines that optimize problems in order to make decisions. This field provides solutions for decision making by taking into account the structure of real-life events. Then, the main objective is to model the behavior of the casuistry studied and give the best solution possible according to the conditions stated. The numerical experiments and the systems that carry out the solution will consider as many characteristics as possible to describe reality. However, it depends on the complexity of the situation. Most part of the work is conducted by analytical processes that require computational techniques to perform the search for optimal solutions.

The historical development of the Operations Research field started at the beginning of the 20th century (Gass and Assad, 2014). However, the theoretical background and the mathematical techniques implemented were developed in Mathematical Analysis and Differential Calculus. Hence, I would like to introduce some background of the Calculus discipline to focus history behind OR.

The study of optimization problems has been one of the main applications of mathematics. From ancient times, people have always attempted to maximize or minimize anything that could be profitable for them. Once civilizations were built, their citizens faced daily situations that required them to take better actions or responses for a given situation. Trading, harvesting, and management were common activities that were required to maximize the benefits. Conversely, transportation and production involve tasks requiring the minimum time possible. Therefore, it is understood that the search for optimal solutions by means of logical properties or methodological steps is as old as human history.

Years later, humans started to analyze the behavior of physical laws. Since the first findings in classic mechanics, the idea of minimizing either distances or times has been a challenging problem for scientists. When focusing on harder problems, we can note that the universe follows a series of rules that rely on equilibrium points, that is to say, solutions that optimize some magnitude. Actually, most of the physical phenomena are governed by the principle of least action. When referring to physical systems, the principle of minimum energy is an empirical finding that is obeyed in thermodynamics. As we can see, the human perception of the universe is in connection with optimization regardless of the matter of study.

Before the 16th century, major advances in calculus were undertaken in the Middle East, India, China, and Japan, where Europe took a secondary role. Historically, we can point to the 17th century as the turning point in the development of analysis and calculus theory. At the beginning of the 17th century, Pierre de Fermat stated a logical implication to find local maxima and minima of real differentiable functions defined over open sets. It is called Fermat's Theorem, and it marked a major step forward in Mathematical Optimization. Basically, it states that if a function is differentiable at some point and this point is a local extremum, then the derivative of the function at such point is zero. In mathematical language, given a differentiable function $f : (a, b) \subset \mathbb{R} \to \mathbb{R}$ with local extremum in $x_0 \in (a, b)$, then the derivative is null in its point (i. e. $f'(x_0) = 0$). Although it only gives us a necessary condition, it was one of the first propositions that evaluate the stationary points. Additionally, when the second derivative function exists, i. e. f'', we can determine whether the stationary point is either a maximum or a minimum. In a physical context, Fermat discussed the law of refraction, which was before approached by W. van Royen Snell and René Descartes. He stated that in any medium studied, light always travels in the shortest time. Such contribution, referred to as Fermat's principle, is now considered the basis of modern optics. From an academic perspective, it is important to highlight the contribution of Maria Gaetana Agnesi, the first woman appointed as a mathematics professor at a university. She wrote *Institutioni Analitiche* in 1748, the first handbook to contain extended works on various mathematical fields such as algebra, analysis, and geometry.

At the end of the 17th century, Sir Isaac Newton and Gottfried Wilhelm Leibniz independently developed infinitesimal calculus as a field of mathematics that studies the continuous changes of mathematical objects. Thanks to their joint effort, the notions of derivative and integral were rigorously defined. The key idea was the definition of infinitesimal (dx), which is a quantity that is closer to zero than any other real number, but it is not equal to zero. Due to such contributions in mathematical calculus, the field of optimization considerably progressed as well. As a demonstration of the interest implicated in the field, I. Newton stated that "*When a quantity is the greatest or the smallest, at that moment its flow is neither forward nor backward*," which is quite closer to the formal definition. As a consequence, there were many advantages in this area of knowledge. Some of them were the fundamental theorem of calculus, the products and chain's rule, or the formal definition of mathematical limit and gradient. Nowadays, the basis of Calculus is still based on the solid conceptualization developed on such days (Brinkhuis and Tikhomirov, 2005).

In the 18th century, Joseph-Louis Lagrange introduced the concept of Lagrange multipliers as a methodology for finding optimal points of a function subject to equality constraints. This work in the calculus of variations meant a great advance in optimization theory. In mathematical language, given an objective function $f : D \subset \mathbb{R}^n \to \mathbb{R}$ and a constraint function $g : D \subset \mathbb{R}^n \to \mathbb{R}^k$ with $k < n$ such that their derivatives are continuous, i. e. both belong to $\mathcal{C}^1$, we can define the Lagrangian function $\mathcal{L}(x, \lambda) = f(x) + \lambda g(x)$. Here, λ is the Lagrange multiplier associated with the problem (Föllmer and Kabanov, 1997). Now, if x_0 is a solution for the constrained optimization problem, then x_0 is also a saddle point of $\mathcal{L}$.

One of the main advantages of the method was the design of a strategy to solve optimization problems with multiple variables. Moreover, the solution does not require an explicit parametrization of the constraints implicated. The Lagrange multiplier method can also be generalized for nonlinear programming purposes by adding some rules for inequality constraints. Within the field of calculus variations, authors such as Johann Bernoulli, Jacob Bernoulli, Adrien-Marie Legendre, and Leonhard Euler significantly contributed to its further evolution. Actually, the field received this name after the book *Elementa Calculi Variationum* written by Euler in 1733. Some remarkable applications were the Seven Bridges of Königsberg problem (which laid the foundations of graph theory), the Knight's Tour problem, and the Transportation problem.

In the 19th century, there were countless contributions by important scientists such as Charles Babbage, Joseph Fourier, Denis Poisson, and Jacob Jacobi. The rigorous foundation of Mathematical Analysis is due to the contributions of Karl Weierstrass, who improved the notation that prevailed in such days by formalizing the ϵ, δ infinitesimals and some axioms and definitions. Augustin Louis Cauchy developed various mathematical notions that helped to formalize the axioms of analysis. Moreover, he proposed iterative methods for solving systems of equations. Carl Friedrich Gauss studied the method of least squares to predict the orbital location of the asteroid Ceres. As a consequence, such a method is now extended in applied mathematics as the objective function. Bernhard Riemann defined the concept of Riemann's integral and gave rigorous properties for its application in real analysis. Among many applications, it is noteworthy to mention the formulation of geodesic and brachistochrone curves. The first is the curve that minimizes the distances between points, and the second is the curve that minimizes the time required to travel from one point to another, i. e. the fastest descent. Related to the linear methodologies, around the 1830s, the Fourier–Motzkin elimination algorithm was implemented for solving systems of linear inequalities via the elimination of variables. A turning point in applicability was the transportation cost problem studied by Babbage for the Royal Mail of the United Kingdom and the Diminishing return principle proposed by D. Ricardo for harvesting purposes.

With the background of Mathematical Analysis and Calculus developed by remarkable mathematicians over the last three centuries, mathematical optimization became a well-defined discipline. At the beginning of the 20th century, the field of Operations Research arose as an application to give a quick answer to military conflicts. We must bear in mind that most applications of modern mathematics emerged due to the impact of the First World War (1914–1918) and the Second World War (1939–1945), as stated in McCloskey (1987). Actually, it was not a sheer coincidence since most of the scientific revolutions have been largely due to historical crises. Around 1936, European countries had one thing in common, take smart countermeasures that prevent themselves from external attacks. The interdisciplinary groups of scientists had to follow a plan: collect data from direct observation, mathematical modeling of the ongoing operations, give the best solution for a given problem, and give feedback on the impact of the strategy conducted. As a result, a scientific-based procedure had born. The approach developed is the same that is used nowadays to carry out projects in the Operations Research field. Not only focused on belligerent or military strategies, today we can find applications of OR in logistics, finances, marketing, R&D management, industrial environments, pharmacist, computer science, applied physics, and so on.

One of the most known applications of OR is linear programming, which is a set of techniques that provides algorithms and tools to solve mathematical optimization problems represented by linear equations. In a precise context, we optimize a linear cost function subject to linear inequality constraints that define the feasible region, that is to say, a convex polyhedron. The canonical form of a linear programming problem can be written as:

$$\begin{aligned} \text{Minimize} \quad & c^T x, \\ \text{Subject to} \quad & Ax \leq b, \\ & x \geq 0, \end{aligned} \tag{2.9}$$

where $x \in \mathbb{R}^n$ is the decision vector, $c \in \mathbb{R}^n$ is the coefficients of the objective function, $A \in \mathbb{R}^{m \times n}$ is the argument matrix, and $b \in \mathbb{R}^m$ the vector of constraints. Some examples of linear programming applications are the following examples:

Problem 1. Diet problem (Pulliam, 1975): Motivated by the USA army, the goal is to minimize the cost of food menus satisfying basic nutrient requirements for their soldiers.

Problem 2. Supply Chain problem (Zhou et al., 2000): Based on the idea of a company that needs to reschedule their production, the goal is to select the best combination of items to produce so that they maximize the income with limited time and material.

Problem 3. Travelling salesman problem (Laporte and Martello, 1990): Given a list of points, what is the shortest cycle path that crosses each city exactly once?

Problem 4. Vehicle routing problem (Federgruen and Simchi-Levi, 1995): It is a generalization of the above mentioned problem. Given a fleet of vehicles, what is the optimal set of routes that deliver some product to a set of customers?

In 1939 Leonid Kantorovich defined the formulation of the linear programming problem with a method for solving it. In a similar way, T. C. Koopmans formulated applied economic problems with linear programming formulation and Frank Lauren Hitchcock studied transportation problems as linear problems with a procedure to solve them. During the 1940s, George B.Dantzig mathematically analyzed the planning problems concerning the United States Air Forces as part of the Headquarters Statistical Control department. During this decade, Dantzig defined the general formulation of linear programming methodology independently from Kantorovich. A great advance came when, in 1947, he developed the Simplex algorithm (Dantzig, 1990). The name of the algorithm is derived from the geometrical concept of simplex for arbitrary dimensions, in particular, the adjacent vertices of a point generate a convex n-polytope. The Simplex algorithm operates in the canonical form of the linear problem and seeks the optimal point in the feasible region defined by the constraints for a given cost function. The implementation solves linear programming, under some conditions, via recursive iterations. Dantzig later discussed his method with

John von Neumann, who quickly linked the Simplex with his work in game theory. As a consequence, Neumann defined the theory of duality which converts the primal problem (Eq. 2.9) into a dual problem with representation:

$$\begin{aligned} \text{Maximize} \quad & b^T y, \\ \text{Subject to} \quad & Ay \leq c, \\ & x \geq 0. \end{aligned} \tag{2.10}$$

This transformation provides us with information about the boundaries of the optimal solution to the primal problem. The dual problem reflected in Eq. 2.10 is called symmetric due to the $x \geq 0$ condition stated in Eq. 2.9, but the problem may be asymmetric when such condition is not presented. It is easy to note that, for every primal problem with symmetric duality, the dual of the dual is the primal problem. In addition to this result, there exists vast literature that gives a mathematical theory for finding optimal solutions to linear programming, for instance, the fundamental theorem of programming theory. In 1951, Harold Kun and A.W. Tucker formalized the nonlinear optimization problems and, in 1957, Richard Bellman developed the concept of dynamic programming.

From then on, the interest in Operations Research exponentially increased and so the literature related with. Consequently, new branches of OR arose such as simulation, expert systems, machine learning, econometric processes, and multi-criteria decision analysis. All of them have a common goal: convert human decision making into an automated, rigorous, and objective procedure.

2.3 COMPUTER SCIENCE

Computer Science is the study of information by means of algorithms and computation techniques. This academic research covers both theoretical and practical approaches, which makes it a complete science. Its applicability is highly related to mathematical calculus due to its objective is to solve a problem by means of a finite number of steps. However, over the years and the industrial developments, various branches with absolutely different focuses have been added to this field. Actually, it is proven that the history of computing is way longer than the history of either software or hardware.

Historically, when we refer to any mechanism or tool that allows humans to perform calculations, we are implicitly talking about computation. The use of stones for accounting purposes was humanity's first attempt to build a computation machine. The first goal was to aid people to compute calculations correctly and quickly. It was a simple but effective technique for doing mathematics due to it only requires to have a bunch of stones as great as needed. Advanced cultures also developed advanced ideas for computation. Another tool that aided to store mathematical identities and methods was the tally sticks. Such artifacts were carved to store relevant information and their history is quite ancient. Two examples are the Lebombo bone, which originated 37000 years ago, and the Ishango bone, founded in the Democratic Republic of Congo and estimated to be about 8500 years old

(Pejlare and Bråting, 2019). With the advent of counting tables, humans could perform fast calculations with a portable tool. The most representative example is the abacus, which is assumed to be designed in the III century in Sumer. The abacus is utilized for addition, subtraction, multiplication, and division; with around 2000 years of documentation use (Samoly, 2012). However, it still has modern educational applications, although with the Chinese decimal abacus version. For a long time, it has been the major computational tool built by humans. Even though it may seem poorly elaborated, most of the basic real-life problems that concern mathematics have been solved by the use of these basic counting tables.

No accurate history of computer science can be told without recognizing the importance of binary logic. That is not only important for the development of modern computer systems, but also for establishing the basis for the information age. Thanks to the developments made in the 18th century by Gottfried Wilhelm Leibniz and in the 19th century by George Boole. Nowadays, the internal system of computers works by means of *on-off* states. In the first place, Leibniz wrote about binary systems as logical operators to execute actions. Secondly, Boole developed the field of boolean algebra due to their book *The Laws of Thought* in 1845, providing consistent algebraic systems and properties to handle binary statements.

In regard to the first contributions to physical technological devices for computation, we can mention Charles Babbage as the first scientist to create a mechanical computer that executed operations up to eight decimal points. In 1834, he designed a machine able to execute operations by means of a punch-card input system which was called an "Analytic engine" (Bromley, 1998). Four years later, he has reached the operative functioning of the machine, which can be considered the first implementation of a modern computer-like instance. At the same age, Ada Lovelace evolved the first computer algorithm for computing Bernoulli numbers, making her the first computer programmer in history. Owing to the research conducted between Babbage and Lovelace, the field of computer science advanced considerably. A major turning point was made in 1886 by Charles Sanders Peirce, who meticulously described how logical operators have to be conducted through electrical switching circuits (Peirce and Eisele, 1976). That revolutionary concept is currently used to produce digital computers. Afterward, he defined the NAND and NOR universal logic gates that could replicate the functioning of every other logic gate, although they were proved and formalized by Henry M. Sheffer.

The 20th century is considered the most prolific era of computer science due to the main technological developments. In the field of electronics, the introduction of capacitors, diodes, inductors, resistors, and transistors in the industry allowed humans to create integrated circuits. In 1936, Alan Turing together with Alonzo Church refined the notion of algorithm, under the so-called Church-Turing thesis, giving a proper definition and remarking the limitations attached to them (Copeland, 1997). In the same year, Turing published his work on the Turing machines, which stated the principles of modern computers and the concept of stored-program computer instructions. Hence, a machine able to execute a given task is considered Turing computable. Thus, we can say that the contributions of Alan Turing in the field of computer science laid the foundations of this subject, which is why he is considered the father of theoretical computer science (Cooper and van Leeuwen, 2013). In addition to him, during the middle of the twentieth century, there were three scientists whose contributions to this field should be highlighted. John von Neumann (1993) developed the known von Neumann architecture/model, which describes an electronic digital computer designed with a processing unit, control unit, memory, mass storage, and input-output devices. Claude Shannon (1948) applied probability theory to electrical applications of Boolean algebra in his work "*A mathematical theory of communication*" and so funding the field of information theory. Norbert Wiener defined the concept of cybernetics from warlike applications. In addition, he theorized about the possibility of machine-like intelligence by means of feedback mechanism (Li et al., 2019), which meant a significant step towards the modern concept.

From the second half of the XX, the number of invents and developments is countless. All the applications, machines, and theories designed up to that point were very useful for engineers and scientists at that time. The computer revolution is still a growing process in both hardware and software. As a consequence, nowadays computers are part of our daily work.

2.3.1 *Python programming language*

At the beginning of 1990, Guido van Rossum created open-source Python with the aim of developing a friendly programming language based on the readability of the code (Van Rossum and Drake Jr, 1995). It is an object-oriented and high-level programming language with general-purpose scope. Python supports multiple programming paradigms like procedural, functional, and object-oriented programming. Currently, it is broadly used due to its elegant syntax and human-readable typing. For instance, the white space indentation to delimit blocks and statements makes it understandable.

Table 2.2: Release versions of Python over the years.

	Release		**End of support**	
Version	Subversion	Date	Subversion	Date
0	0.9	20-02-1991	0.9.9	29-07-1993
1	1.0.4	26-01-1994	1.6.1	05-09-2000
2	2.0.1	16-10-2000	2.7.18	01-01-2020
3	3.0.1	03-12-2008	3.10.4	01-10-2026

In its early stage, Rossum started the deployment at Centrum Wiskunde & Informatica (CWI) in the Netherlands. Thus, the Python interpreter is freely available on their website https://www.python.org/, which is also freely distributed. That site also contains information related to the different versions, documentation, third-party modules, programs, tools, and many more data. The interpreter also has data types previously implemented in C or

C++ with the reference implementation of CPython written in C. The released versions can be divided into four main blocks displayed in Table 2.2.

In 2022, the 3rd version is still maintained and remains the current latest version, which is considered the launching of 3.11 and 3.12 subversions. During the experiments conducted within this thesis, we used the 3.8, 3.9, and 3.10 versions of Python.

The Python Package Index, known as PyPI, is a repository of software for the Python programming language. It helps to find and install software developed by the Python open community. Among all the libraries and packages implemented in Python, we would like to emphasize the ones that have helped to carry out the case study and our experiments:

- NumPy: It is a library for numerical computing in Python (Harris et al., 2020). Numpy is considered referent and a de facto standard for array computing. It offers multiple objects and functions for data vectorization, as well as other algebra and analytic operators. Official website: https://numpy.org/.

- Pandas: It is a powerful data analysis library used for data import and manipulation (Wes McKinney, 2010). It highlights its IO tools that handle multiple data formats. Official website: https://pandas.pydata.org/.

- Matplotlib: It is a library that aims to ease the data visualization procedure (Hunter, 2007). It helps to create static, animated, and interactive plots in a straightforward way. Official website: https://matplotlib.org/.

- SciPy: It contains fundamental algorithms for scientific computing in Python (Virtanen et al., 2020). SciPy provides algorithms for algebra, mathematical calculus, and statistics. Official website: https://scipy.org/.

- Scikit-learn: It is the referent Python package for data analysis and Machine Learning (Pedregosa et al., 2011). Scikit-learn aims to provide not only statistical estimators but also transformers and pre-processors widely used in Business Analytics. It is built on Numpy, SciPy, and Matplotlib. Official website: https://scikit-learn.org/stable/.

- PyTorch: It is an optimized tensor library, based on Torch, for deep learning using GPUs and CPUs (Paszke et al., 2019). By using Python class objects, it converts the graph-topology architecture into a computational model. Official website: https://pytorch.org/.

- TensorFlow: It is an end-to-end open source platform for deep learning (Abadi et al., 2015). It is based on intuitive high-level APIs like Keras for model generation. It makes use of tensors for data processing, thus making the execution efficient. Official website: https://www.tensorflow.org/.

- Statsmodels: It is a library with a complete toolbox for statistical and econometric analysis (Seabold and Perktold, 2010). It provides a large list of models, tests, and data exploration for statistical purposes. Official website: https://www.statsmodels.org.

2.3.2 *R programming language*

Ihaka and Gentleman (1996) designed and implemented a statistical computing language for data analysis and graphics called R. This programming language aims to offer statistical software in an open-source environment with a large variety of packages stored in the Comprehensive R Archive Network (CRAN). The scope of R applications has approached subfields of statistics as data analysis (Chambers, 2008), data science (Peng, 2016), and spatial data analysis (Kaya et al., 2019). Nonetheless, other disciplines that require the use of computational statistics, such as bioinformatics (Gentleman, 2008) or medicine (Sidey-Gibbons and Sidey-Gibbons, 2019), also make use of R. The invention of R was influenced by the S programming language (Chambers, 1998). In particular, R stated as software to overcome many of the shortcomings of the commercial software S-PLUS. The popularity of the R software is mainly associated to its applications in data mining, where statistical success has been obvious (Tippmann, 2015). In 1995, R was made free software under the GNU General Public License.

R language is an object-oriented and functional programming software whose approaches and capabilities are given by customized user-created packages. It can be easily downloaded from its official website https://www.r-project.org/, where its currently available version is 4.2.1 launched on the 23rd of June of 2022. The integrated development environment usually linked to R is RStudio Desktop, which has a free-based download on his website https://www.rstudio.com/. Additionally, it has an excellent reporting system generated by knitr (Xie, 2015), xtable (Dahl et al., 2019), RMarkdown (Xie et al., 2018), and Shiny (Chang et al., 2021b). Some of the most useful R packages are listed here:

- Tidyverse: It is a library that contains a collection of packages intended for data science (Wickham et al., 2019). In its collection we can find useful modules as dplyr, tidyr, readr, and purrr. The design of tidyverse has an underlying philosophy, grammar, and data structures. Official website: https://www.tidyverse.org/.

- Tidymodels: It is a collection of packages with a large framework for statistical modeling (Kuhn and Wickham, 2020). The functionality and principles are based on the tidyverse library. Official website: https://www.tidymodels.org/.

- ggplot2: It is a tidyverse module for data graphics (Wickham, 2016). It is a very famous package for data visualization due to the beauty and elegance of its aesthetics. Official website: https://ggplot2.tidyverse.org/.

- NLoptR: It is an R interface of the NLopt library (Johnson, 2020). It provides algorithms for local/global nonlinear optimization with nonlinear constraints and lower-upper bounds. Official website: https://astamm.github.io/nloptr/.

2.4 BUSINESS ANALYTICS

Business Analytics (BA) refers to the set of procedures and methodologies that develop models for organizations in order to provide intelligent systems that drives the decision making of the business strategy. With the development of modern business models of the XXI century, the interest in this subject has greatly raised, and so has the literature related (Power et al., 2018). The main objective is to join the industrial processes to give added value, which involves tasks from data digitalization to the final launch of the implementation. Due to a large amount of information to consider, BA involves many different methods that provide a correct use of the data acquired and a strategic response to improve the situation of a company (Lepenioti et al., 2020). Then, this field can be broken down into three major subfields:

Descriptive Analytics: It is the study of the current situation of some business event, although it could also analyze past related events. In general, the objective is to gain insight to make an exhaustive diagnosis of the studied scenario. The point is to collect raw data (either owned or from other sources) and convert it into valuable information for the organization. The analysis is usually presented as a summarized report that includes the most relevant items. For this stage, data visualization techniques are commonly applied, which gives an added value that helps the decision makers to understand their actual situation. Descriptive Analytics aims to answer the questions "What has happened?", "Why did it happened?", and "What is happening now?"

Predictive Analytics: It is the study that provides tools, such as algorithms or models, that allow the decision-makers to estimate the future situation of an event by means of forecasting and prognosis. That business part is highly connected with statistics and computer science because the main background is based on the mathematical computation of data. The complexity of the different approaches to performing has changed over the years. Currently, Artificial Intelligence has taken over the sector with the use of advanced data processing and the use of operations research models. As we can note, the level of complexity for this stage is considerably high, so the effort put into this process is also high. Predictive Analytics attempts to answer the questions "What will happen?", "How will happen?", and "Why will happen?"

Prescriptive Analytics: It is the analysis of actions and decisions to carry out from our current situation and probable future events. It can be understood as the last step towards smart decision making or R&D (research and development). It aims to provide the best possible response based on intelligent tools and experience in the business sector. In comparison to Descriptive or Predictive Analytics, this stage is not as developed nor has the same level of consistency. Moreover, this stage can be as accurate as our description and prediction of the studied scenario. Then, it is limited by the knowledge obtained in the last two stages. Prescriptive Analytics has as an objective answer the questions "What will we do?", "How should we do so?", "How can we take advantage of it?", "How can we avoid failure?", and "Assuming a bad prospect for the organization, could we deploy efficient countermeasures?"

The description of the Business Analytics synergy made between the three stages is depicted in Fig. 2.2.

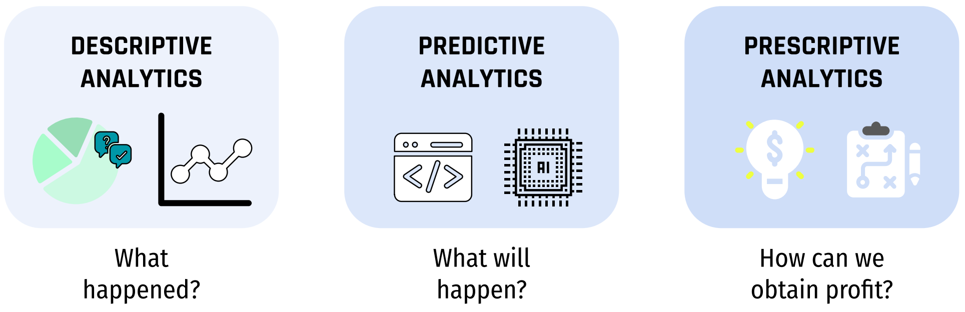


Figure 2.2: Standard stages within a Business Analytics approach for solving some decision-making problem.
Source: Own elaboration.

In the managing and planning of business processes, every single stage of BA is meticulously conducted. Most companies base their strategies on assessable and measurable goals, in such a way stakeholders can evaluate the global position of the actions made. Then, each step is as important as the others because it is necessary to control all the past, present, and future things that affect the business model. The more versatile and dynamic are our strategies, the more accurate and suitable our response will be.

Regarding the incorporation of smart techniques into the business sector, BA has to be understood as a constant loop of proactive implementation. In other words, the representation of Fig. 2.2 has to be constantly applied with respect to time, thus ensuring continuous improvement for the ongoing projects. On the contrary, a fixed business strategy will not take into consideration new competitors or new market niches, so it will not be able to fit the reality of future events.

As far as multi-criteria optimization and statistics are concerned, their case study is focused on the predictive analytics stage. The combination of advanced techniques such as Multiple-Criteria Decision Making, when facing small datasets, or artificial intelligence, when working in a Big Data environment, allow us to perform smart implementation in our industrial processes. Nonetheless, when dealing with predictions or responses of some sort, we usually decide whether to add them to our system or not. In most cases, we do not analyze nor evaluate why our intelligent systems have returned this particular output. In such a case, we have to deal with a less-mentioned problem, which is the explainability of the model's result against the case study.

In the stage between Predictive and Prescriptive Analytics, the evaluation and validation of the models is one of the steps which requires major attention and rigor. It is important to get results that help us to guide our business model, but in any case, it is essential to prove them in experimental tests or simulated scenarios in order to ensure their commissioning. Whenever we want to evaluate the obtained results, we have several approaches to perform it that depend on the level of complexity needed. In Fig. 2.3, we have illustrated the different steps to follow to pass from prediction to decision making. In general, this is a common standard for business strategies and is broadly accepted in data science. Even though it may seem a complete process, sometimes the response is not analyzed per se.

That is to say, even if we know that the answer is correct, we cannot guarantee that such a response is due to coincidence or a correct methodology.

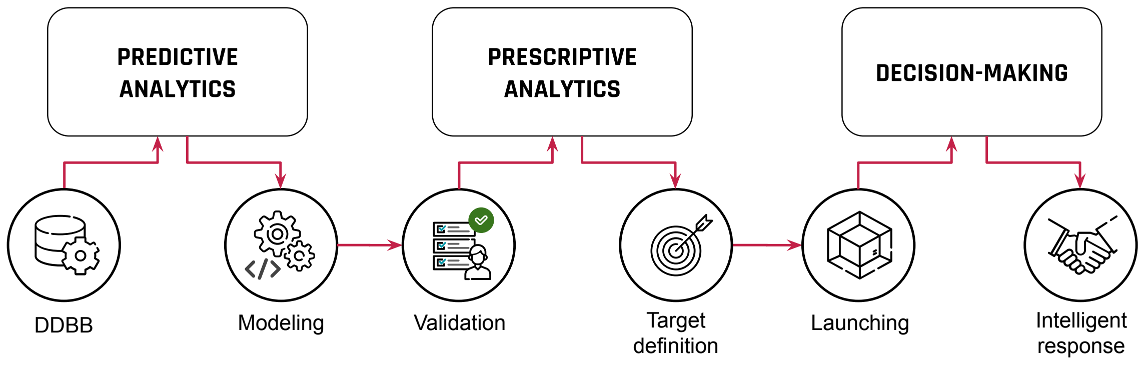


Figure 2.3: Business Analytics transition from predictive to prescriptive solutions and final decision-making application.
Source: Own elaboration.

In this thesis, we have focused our attention on the implementation of predictive models that allow us to give the best response possible for prescriptive purposes. With that purpose, we have designed multiple attribute optimization methodologies to obtain artificial intelligence and multi-criteria decision analysis models. In addition, our goal was not only the algorithms for decision making, but also novel techniques for evaluating how and why our model has returned some given response. That is why the thesis reinforces the prescriptive side of the methods by assessing and evaluating the response that leads to a decision.

2.5 SUMMARY

In this chapter, we have carried out a historical review with the aim of presenting the background related to Multiple-Criteria Decision Making and Artificial Intelligence. For this purpose, we have introduced elemental concepts of the fields of Probability, Statistics, Operations Research, Computer Science, and Business Analytics. Hence, it is assumed that the basic knowledge in each mentioned area is understood and also that the reader is acquainted with these subjects.

3 MULTIPLE-CRITERIA DECISION MAKING

In our daily lives, we face real-world problems with a high level of complexity, inaccuracy, and uncertainty. Human intuition when dealing with these problems makes us consider various points to take into account, which is taught through observation and examination of related past events. Then, it is essential to recognize that decisions are based on simultaneous variables that determine the degree of subjective assessment that people attribute to each of them. In fact, it would be vague to represent the nature of an event with a unique variable and expect that it describes the whole behavior of the process. As a consequence, the strategic planning for aiding a decision-making problem has to be approached by consensus for a multidisciplinary team expert in solving similar matters. They are the so-called decision-maker. Their experience with the problem will decide the external components to provide beforehand in order to utilize them during the computational phase. In turn, they are responsible to analyze the impact and the consequences of both selected criteria and their relevance over the final decision.

When talking about the relative importance of a criterion, we are implicitly pointing to a weighting scheme, which is defined as a numerical board containing a value linked to each criterion. That scheme has to convey the insight of the people in charge of the assessment that will be mathematically represented. Thus, regardless of the kind of choice (categorical or numerical), it must be consistent and robust in any case. Such values rely not only on how decisive a feature is but also on the modeling phase that involves their gathering and processing. The second part is not trivial due to the nature of the problem can corrupt the data, as well as their structure.

Multiple-Criteria Decision Analysis (MCDA) or Multiple-Criteria Decision Making (MCDM) is a branch of operations research defined as a set of algorithms and methodologies whose goal is to aid the decision-maker when there exist multiple conflicts of interests (Hwang and Yoon, 1981). In order to study, analyze, and solve the above-mentioned problems, the MCDA field focuses on the development and implementation of tools from various perspectives and strategies. It combines mathematical modeling with an economical underlying point of view to attach a strong level of consistency (Triantaphyllou et al., 1997). It is also combined with advanced computational calculus to automatize and speed up the decision times. As a whole, we have an interactive system of methods that allow the DMs to interactively select and process their datasets with satisfactory feedback that eases common agreement between the agents involved.

Although MCDA is developed as a distinguished subfield of OR, many other applied disciplines have utilized their algorithms and techniques to solve conflicting decision scenarios (Doumpos and Grigoroudis, 2013). The major advantage of its application is that it combines experienced-based background, by means of the selection of criteria, with a model-driven selection of alternatives based on mathematical reasoning, via discrete optimization, which gives quantitative confidence when reported in the decision making (Roy, 1996). A detailed procedure of the stages involved in a modeling process for MCDA is depicted in Fig. 3.1.

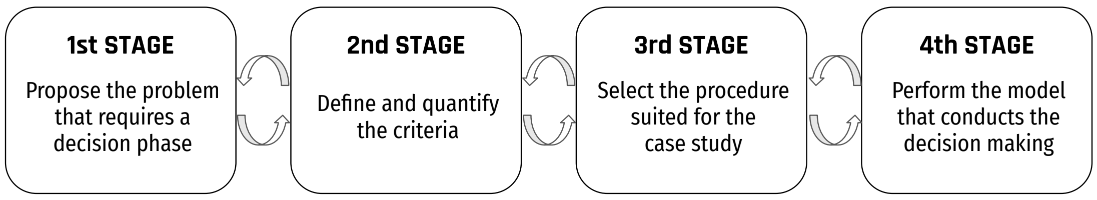

1st STAGE
Propose the problem that requires a decision phase
2nd STAGE
Define and quantify the criteria
3rd STAGE
Select the procedure suited for the case study
4th STAGE
Perform the model that conducts the decision making

Figure 3.1: Stages involved in an MCDA process.
Source: Own elaboration.

Due to the optimization background that relies on the MCDA methodology, this field is divided into two multi-objective optimization branches (Triantaphyllou, 2000). On the one hand, Multi-Objective Decision Making (MODM) concerns the decision problems for a continuous decision space. On the other hand, Multi-Attribute Decision Making (MADM) just refers to decision problems with discrete decision spaces. In particular, when talking about MCDM we are referring to MADM since we just study discrete optimization.

Despite the fact that MCDA offers many advantages, it is important to know some of its limitations. Given that the conflict of interests between DM's might be cumbersome, the procedures implemented during the methods do not always assure optimal solutions. Instead, the final decision is actually the result of a mutual accordance among each of the individuals.

3.1 APPROACHES OF MULTI-CRITERIA DECISION ANALYSIS

The different methodologies implemented in MCDA can be divided into 4 subfields as stated by Doumpos and Grigoroudis (2013). Depending on the data utilized, the sorting algorithm, the data processing, the comparisons between alternatives, and the scope of the investigation we can categorize the deployed method.

3.1.1 *Multi-Objective Mathematical Programming*

Before we present the list of MCDA algorithms developed in the thesis, we want to highlight the underlying mathematical optimization required to solve those problems. In the early 1960s (Charnes et al., 1963; Charnes et al., 1968) introduced simple linear programming for goal programming. Over the years, it meant an incredible advance in operations research, with increasing success in the 21st century (Aouni and Kettani, 2001). In the field of BA, we always have attributes or criteria conflicting with each other. Then, we can mathematically formulate a multi-objective optimization problem (MOP) as:

$$\begin{aligned} \text{minimize} \quad & F(x) = [f_1(x), \ldots, f_k(x)], \\ \text{s.t} \quad & g_j(x) \leq 0, \qquad 1 \leq j \leq m \\ & x \in \Omega, \end{aligned} \tag{3.1}$$

where $x \in \mathbb{R}^n$ is the vector of design variables, Ω is the feasible set of vectors, and each g_j the constraint functions. From now on, we have to define a vector sort criterion because the vector space $\mathbb{R}^n$ is not ordered per se.

Definition 3.1.1. *Given two vectors $X, Y \in \mathbb{R}^n$, the order relationship $\trianglelefteq$ in the vector space $\mathbb{R}^n$ is defined as:*

$$\begin{aligned} X \lhd Y \quad & \text{if and only if} \quad x_i < y_i, 1 \leq i \leq n. \\ X \trianglelefteq Y \quad & \text{if and only if} \quad x_i \leq y_i, 1 \leq i \leq n. \end{aligned}$$

Even though the $\trianglelefteq$ relationship is well-defined, in MOPs we cannot ensure the existence of a unique solution that optimizes each objective function f_i. Hence, the concept of Pareto dominance is presented to overcome that concern.

Definition 3.1.2. *An element of the feasible set $x^* \in \Omega$ in a MOP framework is said to be a Pareto optimal solution if x^* dominates any other $x \in \Omega$, i. e. $F(x) < F(x^*)$.*

The definition of the Pareto optimal solution allows us to define the Pareto frontier (or Pareto front) as the Pareto solutions over the feasible set associated with the MOP. In Fig. 3.2 is depicted how a Pareto frontier would be when solving an optimization problem regarding just two objectives.

One way to address the problem Eq. 3.1 is by means of goal programming formulation (Tamiz et al., 1995). The point is to optimize some defined goals, to which some deviations are attached. The concept of a goal has to describe some ideal scenario or reference point for comparison purposes. Once the goals are set by the decision-makers, the formulation is presented as:

$$\begin{aligned} \text{minimize} \quad & G_d(d_j^+, d_j^-, w), \\ \text{s.t.} \quad & f_j(x) + d_j^+ - d_j^- = s_j \quad \forall j \in \{1, \ldots, k\}, \\ & d_j^+, d_j^- \leq 0 \qquad \forall j \in \{1, \ldots, k\}, \\ & x \in \Omega, \end{aligned} \tag{3.2}$$

in which, the s_j values are the target levels for the objective j and each d_j^+, d_j^- are the target deviations. Finally, G_d is the function of deviations parameterized by a weighting scheme w.

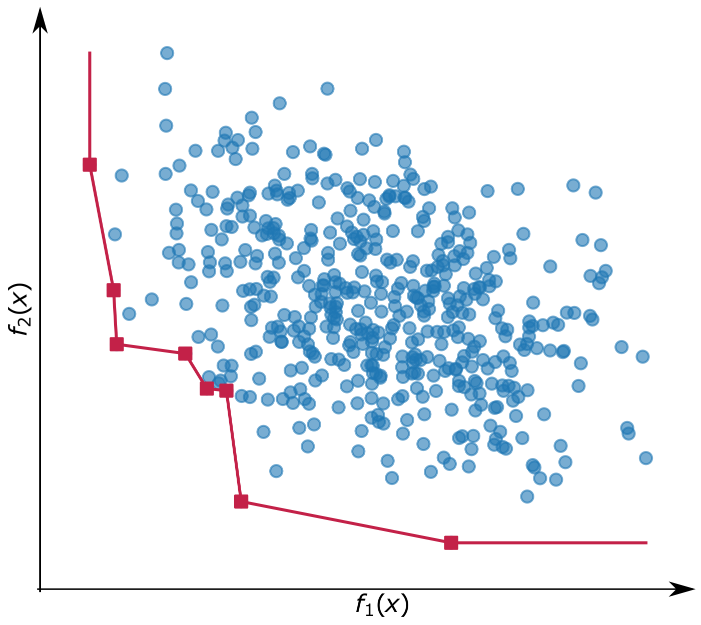


Figure 3.2: Pareto frontier (red color) for a bi-objective problem of f_1 and f_2. The remaining points represent the non-Pareto solutions.
Source: Own elaboration.

3.1.2 *Multi-Attribute Utility/Value Theory*

Multi-Attribute Utility/Value Theory, or MAUT/MAVT in short, conveys the classical case of utility theory in multidimensional spaces to MCDA. On the one hand, the utility refers to the uncertainty related to the decision matrix (Vincke, 1992). On the other hand, the value refers to the magnitudes in a known environment. The concept of utility was already studied in the 18th century by Jeremy Bentham, which stated that every action could be measured through six dimensions of value, i. e. criteria (Gass and Assad, 2005). In his proposal, he aggregated positive and negative categories to obtain the score associated with the action. Hence, it can be considered one of the first approaches to solving decision-making problems. In modern terminology, it is generalized to multidimensional spaces with functional operators as utilities or values.

Given a vector of decision data x the objective of both MAUT and MAVT is to aggregate the known attributes of each alternative to create a reference system. It is computed by means of a so-called value function $V(x)$, which is responsible to give meaningful information to the decision-makers. It is mathematically represented as:

$$V(x) = \sum_{j=1}^{M} w_j v_j(x_j). \tag{3.3}$$

Each w_j value is the trade-off constant, where it is mainly stated that their sum is equal to 1, and each v_j is the marginal value function per each criterion j. Such functions are set in a predefined scale, which is usually $[0,1]$. The particular case of MAUT/MAVT where $v_k(x) = x^p$, $p > 0$, is the weighted average mean per each criterion, in which $p = 1$ matches with the weighted sum model and $p = 0$ with the weighted product model.

Once we know how we asses each criterion v_j and how we aggregate them V, we can arrange each alternative, through their decision data available, by using the $\lhd$ pairwise relationship per each i, k element:

$$\begin{aligned} x_i \lhd x_k \quad & \text{if and only if} \quad V(x_i) < V(x_k) \quad \text{Domination} \\ x_i \sim x_k \quad & \text{if and only if} \quad V(x_i) = V(x_k) \quad \text{Indifference} \end{aligned} \tag{3.4}$$

Finally, an experimental analysis is applied to the outcome in order to evaluate the performance of the model. Differential techniques and sensitivity analysis are common practices that allow us to understand the results and discover new alternatives. In Fig. 3.3 it is depicted the procedure to carry out a Multi-Attribute Utility/Value technique.

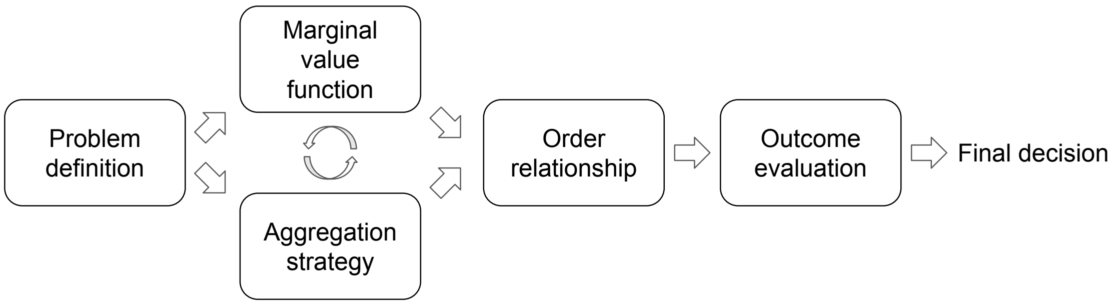


Figure 3.3: Workflow required to conduct a Multi-Attribute Utility/Value methodology. Source: Own elaboration.

3.1.3 *Outranking Relationships*

Roy (1990) stated the foundations of outranking relation theory (ORT) in the 1960s. The most representative method is the ELECTRE (ELimination Et Choix Traduisant la REalité) defined in Benayoun et al. (1966) and subsequently developed in Roy (1968), which deals with outranking relations via pairwise comparisons among alternatives under each one of the criteria separately. Afterward, it evolved into ELECTRE II, ELECTRE III, ELECTRE IV, ELECTRE IS, and ELECTRE TRI; among many other variants.

The prime ELECTRE method can be applied following the next steps stated in Triantaphyllou (2000).

ELECTRE method:

Step 1 Normalize the decision matrix $[x_{ij}]$, per each $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$, as the vector normalization:

$$r_{ij} = \frac{x_{ij}}{||x_j||_1} = \frac{x_{ij}}{\sum_{l=1}^{M} x_{lj}}. \tag{3.5}$$

Step 2 Calculate the weighted normalized decision matrix $[v_{ij}]$ according to the set of weights $\{w_j\}_{j=1}^{M}$ such that $\sum_{j=1}^{M} w_j = 1$. Then, we have $v_{ij} = w_j r_{ij}$ for each $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$.

Step 3 Determine the concordance and discordance sets that allow us to compute the pair-wise relationships:

$$\begin{aligned} C_{kl} &= \{j : v_{kj} \geq v_{lj}\} && \text{Concordance set,} \\ D_{kl} &= \{j : v_{kj} < v_{lj}\} && \text{Discordance set.} \end{aligned} \tag{3.6}$$

Step 4 Define the concordance and discordance matrices by means of the previously defined sets:

$$\begin{aligned} \mathbf{C}_{kl} &= \sum_{j \in C_{kl}} w_j && \text{Concordance matrix,} \\ \mathbf{D}_{kl} &= \frac{\max\limits_{j \in D_{kl}} |v_{kj} - v_{lj}|}{\max\limits_{j} |v_{kj} - v_{lj}|} && \text{Discordance matrix,} \end{aligned} \tag{3.7}$$

where the entries $k = l$ are not defined for either matrices **C** and **D**.

Step 5 Determine the concordance and discordance matrices via threshold comparisons. Such values are calculated through average concordance-discordance indexes, whose values are:

$$\begin{aligned} \bar{c} &= \frac{1}{M(M-1)} \sum_{k \neq l} \sum_{l \neq k} \mathbf{C}_{kl}, \\ \bar{d} &= \frac{1}{M(M-1)} \sum_{k \neq l} \sum_{l \neq k} \mathbf{D}_{kl}. \end{aligned} \tag{3.8}$$

Now, we can define the binary matrices by means of the $\bar{c}$ and $\bar{d}$ values as:

$$\begin{aligned} \mathbf{F}_{kl} &= \mathbf{1}[\mathbf{C}_{kl} \geq \bar{c}] && \text{Concordance dominance matrix,} \\ \mathbf{G}_{kl} &= \mathbf{1}[\mathbf{D}_{kl} \geq \bar{d}] && \text{Discordance dominance matrix.} \end{aligned} \tag{3.9}$$

Step 6 Determine the aggregate dominance matrix as the product of the dominance matrices, i. e.

$$\mathbf{E}_{kl} = \mathbf{F}_{kl} \cdot \mathbf{G}_{kl}. \tag{3.10}$$

Step 7 Iteratively eliminate the “less favorable” alternatives employing the $\mathbf{E}_{kl}$ entries. It is easy to note that if $\mathbf{E}_{kl} = 1$, then the alternative A_k dominates the A_l alternative, and so it is preferred for both concordance and discordance. As a consequence, the best alternative is the one that overpasses all other alternatives.

Another widely recognized instance of outranking models is the PROMETHEE (Preference Ranking Organization Method for Enrichment Evaluations) method (Brans, 1982; Mareschal et al., 1984). It is based on differences among the scores of the alternatives to get the best sample. There exist six types of preference functions defined by Brans and Mareschal (2005) for comparative purposes.

The most basic version of the PROMETHEE methods can be applied following the steps described in Behzadian et al. (2010).

PROMETHEE method:

Step 1 Compute the deviations between criteria per each alternative for each $j \in \{1, \dots, M\}$:

$$d_j(a, b) = g_j(a) - g_j(b) \text{ with } a, b \text{ alternatives,} \tag{3.11}$$

where each g_j is the evaluation of the criteria j expressed in their units and magnitude. Then, these functions are responsible to rescale the data and so avoid a normalization step.

Step 2 Apply the preference function of each criteria $j \in \{1, \dots, M\}$:

$$P_j(a, b) = F_j[d_j(a, b)], \tag{3.12}$$

where each F_j is a function with image in $[0, 1]$.

Step 3 Calculate the overall preference index:

$$\pi(a, b) = \sum_{j=1}^{M} w_j P_j(a, b), \quad a, b \text{ alternatives.} \tag{3.13}$$

It is interesting to highlight that $0 \leq \pi(a, b) \leq 1$ per each pair of alternatives a, b and $\pi(a, a) = 0$.

Step 4 Calculate the outranking flows, known as PROMETHEE I partial ranking, as the input-output average of preference index per each alternative a:

$$\begin{aligned} \phi^+(a) &= \frac{1}{N-1} \sum_b \pi(a, b) \quad \text{Positive outranking flow,} \\ \phi^-(a) &= \frac{1}{N-1} \sum_b \pi(b, a) \quad \text{Negative outranking flow.} \end{aligned} \tag{3.14}$$

The outranking flow expresses the dominance of an alternative over the resultant. So the higher $\phi^+(a)$ and the lower $\phi^-(a)$, the better the alternative a.

Step 5 Calculate the net outranking flow, known as PROMETHEE II complete ranking, as

$$\phi(a) = \phi^+(a) - \phi^-(a). \tag{3.15}$$

We would like to emphasize that $-1 \leq \phi(a) \leq 1$ and $\sum_a \phi(a) = 0$.

Step 6 Rank the alternatives in descending order of the values of $\phi(a)$ since positive values indicate that a outranks and negative values indicate that is outranked.

In general, an outranking relationship is understood as a binary relation that determines the dominance of an alternative. Then, an alternative is preferred to any other if it dominates it for some benchmark. Unlike MAUT/MAVT, the outranking relation methods have two particular features:

1. Outranking relationships are not transitive, while MAUT/MAVT is transitive.

2. Outranking relationships are not complete, while MAUT/MAVT has the relation Eq. 3.4. Then, it is possible to tackle incomparable relations among alternatives.

Other useful outranking methods are described in § 3.2, with further approaches and in-depth analysis of their properties.

3.1.4 *Preference Disaggregation Analysis*

When a problem with multiple criteria arises, the main challenge for both decision-makers and analysts is how we come up with the proper model that leads to the final decision (Jacquet-Lagreze and Siskos, 2001). Sometimes, the procedure to find a suitable method is impossible since it is a time-consuming task. On the contrary, analysts could asses the preferences of DMs by taking into account given successful past related instances. Then, disaggregation approaches are built on a data-driven system in order to infer from decision examples through regression-like schemes (Doumpos et al., 2022). Thus, Preference Disaggregation Analysis (PDA) is an indirect approach to extracting underlying preferential information, such as comparisons, weighting schemes, trade-offs, etc; that will aid to solve future decision-making problems.

Jacquet-Lagreze and Siskos (1982) developed the UTilité Additive (UTA) method, which is considered the introduction of PDA theory for MCDA. It is very helpful when time limitations exist or when the amount of information to process is massive. UTA basically generates MAUT/MAVT functions by applying linear programming techniques for optimal inference. In general terms, the Preference Disaggregation Analysis framework is formally defined as a set $G = \{g_1, \ldots, g_M\}$ of M criteria to maximize, where each $g_j(a)$ represents the performance of the alternative a over the criterion j (Doumpos et al., 2022). Depending on the scope of the problem, the objective of the PDA will be to:

1. Select the best alternatives.
2. Rank the set of alternatives.
3. Sort the alternatives through performance categories.

In such a way, the decision model will lead to functional models (MAUT/MAVT functions), relational models (ORT), or symbolic models (based on decision rules). The type of model has to return the best fit according to a predetermined set of parameters or criteria. Finally, the evaluation of such fit has to be conducted by means of the set of responses Y and a defined loss function $\mathcal{L}$. The set of parameters obtained can be mathematically written as:

$$P^* = \underset{P}{\operatorname{argmin}}\, \mathcal{L}(\hat{Y}_P, Y). \tag{3.16}$$

A classic example of a PDA method is the UTA-based model named MUlticriteria Satisfaction Analysis (MUSA), presented in Siskos et al. (1998). It was designed to measure customer satisfaction by using ordinal regression techniques.

3.1.5 *Multi-Criteria Decision Making properties*

In the last section, we have mentioned some approaches of MCDA depending on the strategy used to achieve the best result. Although they are very diverse according to their focus, they have many properties in common (Hwang and Yoon, 1981). Some of them are described in the following list in the same way as in Triantaphyllou (2000):

Alternatives: It is the set of available choices to consider for the decision markers. They are supposed to be finite and well-known for the agents involved in the selection phase. In general, they represent the candidates, elements, or items to select in MCDM. In this chapter, we suppose that we have $N \in \mathbb{N}$ different alternatives.

Multiple criteria: Each alternative has a series of associated attributes or features that allows them to be considered when conducting the appropriate comparisons. They are responsible for determining whether an alternative is preferred for some objective or goal. From a mathematical perspective, the criteria determine the dimension of the decision space. In this chapter, we assume that we have $M \in \mathbb{N}$ available attributes to use as criteria.

Conflict among criteria: Owing to the attribute representation, the behavior of the variables may conflict with each other. A clear example would be the benefits conflicting with the overhead. Thus, the confrontation among criteria is a common pattern in MCDM.

Incommensurable Units: It is important to highlight that the nature of the criteria is usually connected with a certain magnitude, so each attribute might have its own units. For example, we could have a scheme that considers distances (meters) and times (seconds). Hence, the incompatibility associated with the combination of criteria adds more complexity to the MCDM problems

Decision weights: Every MCDM method requires a weighting scheme that assigns the relative importance or impact over the model. It is broadly extended that they are normalized, i. e. the sum of them is equal to 1.

Decision matrix: The notation related to an MCDM can be mathematically represented as a matrix. It is an easy formalization since we have N alternatives and M associated criteria. Throughout this section, such decision matrix is denoted as $X = [x_{ij}]$ with $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, M\}$. In addition, the decision weights are attached to X as $w = (w_1, \dots, w_M)$, in which all the weights satisfies $w_j \in [0, 1]$.

Although these six properties appear in every MCDM problem, each method has its own singularities that determine the type of problem and the scope of the purpose. One way to classify them is depending on the dataset utilized we can distinguish among three possible scenarios: Deterministic (no randomness is involved in the problem), Stochastic (the information is defined as a family of random variables), or Fuzzy (the dataset has non-random uncertainty attached). Another way to classify the methods is by considering

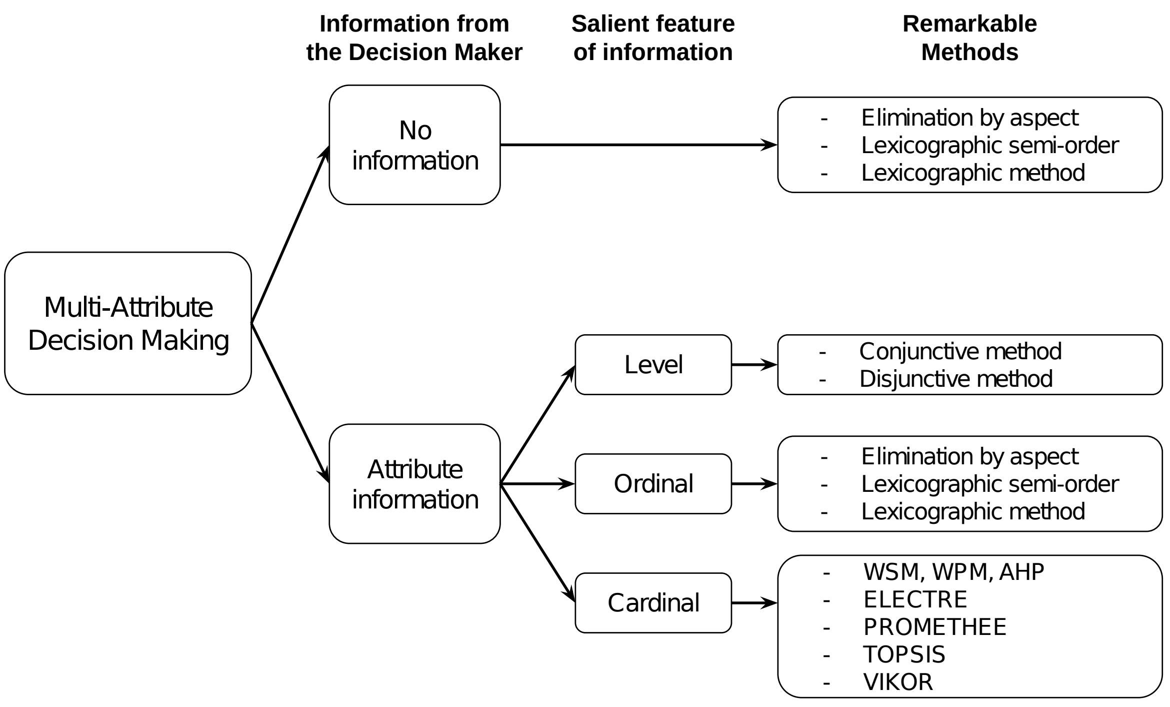


Figure 3.4: Principal classes of MCDM methods based on a priori knowledge of the problem. Source: Own elaboration, based on Hwang and Yoon (1981).

the number of DMs that take part in the decision process. It is basically divided by single or grouped decision-makers. In this thesis, we have focused our attention on single DM with deterministic and fuzzy approaches. In Hwang and Yoon (1981) the problems are distinguished according to the information known by the DM, and the Fig. 3.4 illustrates the structure described.

3.1.6 *Data preprocessing*

As far as MCDA is concerned, the decision-makers have to deal with conflict of interest that entails the decision process. The collection of data, in particular the criteria selected, is a hard task since it has to describe the reality of the problem and satisfy the requirements of the DMs at the same time. Then, the criteria obtained has usually a different nature, and so a different magnitude. It is easy to understand that the combination of distinct units is mathematically and physically incorrect. In order to avoid the inappropriate step of combining such magnitudes, we make use of the known process of normalization or standardization. It basically involves a criteria-based transformation of the data before it is computed through the decision-analysis algorithms.

In mathematical terms, normalization is a vector application $\psi : \mathbb{R}^N \to \mathbb{R}^N$ so that it unifies self-variables or feature ranges in data. In Table 3.1 we have described some of the most common normalization methods applied in both MCDM and Ia for the data preprocessing step. An interesting particularity is that most of the researchers agree on scaling the variables to $[0, 1]^N$, although it is not strictly necessary because it depends on the origin of the data.

Table 3.1: List of the most used normalization methods in statistics for a given decision matrix $[x_{ij}]$.

Normalization method	**Formulation** $[\mathbf{r_{ij}}]$
Vector ℓ^1 normalization	$\frac{x_{ij}}{\|\|x_j\|\|_1} = \frac{x_{ij}}{\sum_{l=1}^{M} \|x_{lj}\|}$
Vector ℓ^2 normalization	$\frac{x_{ij}}{\|\|x_j\|\|_2} = \frac{x_{ij}}{\sqrt{\sum_{l=1}^{M} x_{lj}^2}}$
Minkowski p-normalization	$\frac{x_{ij}}{\|\|x_j\|\|_p} = \frac{x_{ij}}{\left[\sum_{l=1}^{M} \|x_{lj}\|^p\right]^{\frac{1}{p}}}$
Feature (a, b)-rescaling	$a + (b-a)\frac{x_{ij} - \min\{x_{ij}\}}{\max\{x_{ij}\} - \min\{x_{ij}\}}$
Feature rescaling (Max-Min)	$\frac{x_{ij} - \min\{x_{ij}\}}{\max\{x_{ij}\} - \min\{x_{ij}\}}$
Feature Max-rescaling	$\frac{x_{ij}}{\max\{x_{ij}\}}$, with $\max\{x_{ij}\} > 0$
Feature Min-rescaling	$\frac{\min\{x_{ij}\}}{x_{ij}}$, with $x_{ij} > 0$
Feature mean rescaling	$\frac{x_{ij} - \frac{1}{M}\sum_{l=1}^{M} x_{lj}}{\max\{x_{ij}\} - \min\{x_{ij}\}}$
Feature reference rescaling	$\frac{x_{ij}}{\lambda_j}$, with known $\lambda_j \in \mathbb{R}\backslash\{0\}$
Standardization	$\frac{x_{ij} - \bar{x}_j}{\sigma_j}$
Gaussian normalization	$\frac{1}{\sigma_j\sqrt{2\pi}} \exp\left(-\frac{(x_{ij} - \bar{x}_j)^2}{2\sigma_j^2}\right)$
Fuzzy normalization	$\mu_{(x_j^L, x_j^R, \alpha_j^L, \alpha_j^R)_{L_j, R_j}}(x_{ij})$, as in Def 3.35

3.2 CLASSICAL APPROACH

As far as MCDM is concerned, the problem is performed by determining the decision matrix $X = [x_{ij}]$, so that $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$, where N and M are the number of alternatives (A_i) and criteria (C_j), respectively. Then, the starting point for a classical MCDM can be represented as:

$$\begin{array}{c|cccc|}
 & C_1 & C_2 & \dots & C_M \\
\hline
A_1 & x_{11} & x_{12} & \dots & x_{1M} \\
A_2 & x_{21} & x_{22} & \dots & x_{2M} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
A_N & x_{N1} & x_{N2} & \dots & x_{NM} \\
\hline
w & w_1 & w_2 & \dots & w_M \\
\hline
\end{array}$$

The relevance of each criterion within the model is transmitted by means of the w vector. It is known as the set of weights so that they aggregate and regulate the values of the decision matrix. In order to formalize a definition of weight in multicriteria optimization, we design $\Omega = \{(w_1, \dots, w_M) : \sum_{j=1}^{M} w_j = 1, 0 \leq w_j \leq 1\}$ as the set of weights.

3.2.1 *Weighted Mean Models*

Weighted Mean Models (WMM) are a generalization of the commonly used weighted sum models (WSM) formally introduced by Fishburn (1967), which is one of the classic MCDA techniques due to its simplicity (Doumpos and Grigoroudis, 2013). The main idea of the WMM is the additive utility assumption so that the value of a generalized p-mean determines the ranking score. The greater the value, the higher score for the alternative. Unlike the WSM, here we make use of normalization functions to avoid the main weakness of the magnitude units when working with non-homogeneous data. In order to show the mathematical properties of the method, Eq. 3.17 indicates its score function for a given $x = (x_1, \dots, x_M) \in \mathbb{R}^M$ of positive elements, $w = (w_1, \dots, w_M) \in \Omega$ and $p \in \mathbb{R}\backslash\{0\}$ to avoid non-rigorous definitions:

$$\mathcal{M}^p(x, w) = \left[\sum_{j=1}^{M} w_j x_j^p\right]^{\frac{1}{p}} \tag{3.17}$$

Proposition 3.2.1 (*Bullen*, 2003)**.** *According to* Eq. 3.17*, there exists the limit of* $\mathcal{M}^p$ *when* p *tends to* $-\infty$, 0, *and* $+\infty$.

Proof. We assume that $x_j > 0$ per each $1 \leq j \leq M$, because the 0-case would be trivial.

- If p tends to $+\infty$, we consider that $x^+ = \max\limits_{1\leq j\leq M} x_j > 0$, then,

$$\lim_{p\to+\infty} \mathcal{M}^p(x, w) = x^+ \lim_{p\to+\infty} \left(\sum_{j=1}^{M} w_j \left(\frac{x_j}{x^+}\right)^p\right)^{\frac{1}{p}} = x^+.$$

  The second equality is easy to prove since $\frac{x_j}{x^+} \leq 1, \forall j \in \{1, \dots, M\}$.

- When p tends to $-\infty$, we proceed analogously by considering the element $x^- = \min\limits_{1\le j\le M} x_j > 0$ over the vector $(\frac{1}{x_1}, \dots, \frac{1}{x_M})$. Thus $\frac{1}{x^-}$ is the maximal element of the reshaped vector, so we can notice that:

$$\max_{1\le j\le M} x_j = \min_{1\le j\le M} \left(\frac{1}{x_j}\right) = \frac{1}{\max\limits_{1\le j\le M}\left(\frac{1}{x_j}\right)}.$$

Therefore, we just have to apply it in the following statement:

$$\lim_{p\to-\infty} \mathcal{M}^p(x,w) = \lim_{p\to+\infty} \mathcal{M}^{-p}(x,w) = \frac{1}{\lim\limits_{p\to+\infty}\left(\sum\limits_{j=1}^{M} \frac{w_j}{x_j^p}\right)^{\frac{1}{p}}} = \frac{1}{\frac{1}{x^-}} = x^-.$$

- When p tends to 0, we just have to consider the exponential and logarithmic functions, because they are monotonic increasing functions in $\mathbb{R}^+$. Then,

$$\lim_{p\to 0} \mathcal{M}^p(x,w) = \lim_{p\to 0} e^{\log \mathcal{M}^p(x,w)} = \lim_{p\to 0} e^{\left(\frac{1}{p}\log\sum_{j=1}^M w_j x_j^p\right)}.$$

Applying L'Hôpital's rule, we obtain $\lim\limits_{t\to 1}\frac{\log t}{t-1} = \lim\limits_{t\to 0}\frac{\log t+1}{t} = 1$ and $\lim\limits_{t\to 0}\frac{a^t-1}{t} = \log a$. Hence,

$$\lim_{p\to 0}\frac{1}{p}\log\sum_{j=1}^{M} w_j x_j^p = \lim_{p\to 0}\frac{\log\sum_{j=1}^M w_j x_j^p}{\sum_{j=1}^M w_j x_j^p - 1}\cdot\frac{\sum_{j=1}^M w_j x_j^p - 1}{p} = \lim_{p\to 0}\frac{\sum_{j=1}^M w_j x_j^p - 1}{p} =$$

$$= \lim_{p\to 0}\sum_{j=1}^{M} w_j \frac{x_j^p - 1}{p} = \sum_{j=1}^{M} w_j \log x_j = \log\left(\prod_{j=1}^{M} x_j^{w_j}\right).$$

So we can conclude that the limit exists and its value is,

$$\lim_{p\to 0}\mathcal{M}^p(x,w) = \lim_{p\to 0} e^{\log\left(\prod_{j=1}^M x_j^{w_j}\right)} = \prod_{j=1}^{M} x_j^{w_j}.$$

□

Proposition 3.2.2 (*Bullen, 2003*). *Let X be a vector in $\mathbb{R}^M$, $p \in \mathbb{R}\backslash\{0\}$ and $w \in \Omega$ a vector of weights, then the following statements hold true:*

i. $\min\{x_1, \dots, x_M\} \le \mathcal{M}^p(X,w) \le \max\{x_1, \dots, x_M\}$.

ii. $\mathcal{M}^p(\lambda X, w) = \lambda\mathcal{M}^p(X,w)$, *per each* $\lambda \in \mathbb{R}$.

iii. $\mathcal{M}^p\left((x_1, \dots, x_M), w\right) = \mathcal{M}^p\left((x_{\sigma(1)}, \dots, x_{\sigma(M)}), w\right)$, *per each 1-homogeneous permutation function* σ *in* $\{1, \dots, M\}$.

Once we know that the generalized p-mean is well defined, given a real number p, we can proceed with the WMM by means of the following three steps.

Weighted Mean Model method:

Step 1 Normalize the decision matrix as $[r_{ij}]$, per each $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, M\}$.

Step 2 Calculate the weighted p-mean of each alternative:

$$\mathcal{M}_i^p = \mathcal{M}^p(r_{ij}, w) = \left[\sum_{j=1}^{M} w_j r_{ij}^p\right]^{\frac{1}{p}}, \quad 1 \leq i \leq N. \tag{3.18}$$

Step 3 Make a ranking of the alternatives in descending order of the values of $\{\mathcal{M}_i^p\}_{i=1}^N$.

Although we follow the hypothesis of maximizing the value of each criterion, we can transform the features by using monotonic decreasing functions. Thus, we are modifying the optimal direction of our functions. Depending on the scope of the problem, the Step 3 may be suppressed to select just the maximal element of the sequence.

Applying the Proposition 3.2.1, the WMM algorithm can be formalized for every $p \in \mathbb{R}$ as:

$$\begin{aligned}
\mathcal{M}^{-\infty}(r_{ij}) &= \min_{1\leq j\leq M}\{r_{ij}\} && \text{Minimum}\\
\mathcal{M}^{-1}(r_{ij}) &= \frac{1}{\sum_{j=1}^{M}\frac{w_j}{r_{ij}}} && \text{Harmonic mean}\\
\mathcal{M}^{0}(r_{ij}) &= \prod_{j=1}^{M} r_{ij}^{w_j} && \text{Geometric mean}\\
\mathcal{M}^{1}(r_{ij}) &= \sum_{j=1}^{M} w_j r_{ij} && \text{Arithmetic mean}\\
\mathcal{M}^{2}(r_{ij}) &= \sqrt{\sum_{j=1}^{M} w_j r_{ij}^2} && \text{Quadratic mean}\\
\mathcal{M}^{+\infty}(r_{ij}) &= \max_{1\leq j\leq M}\{r_{ij}\} && \text{Maximum}
\end{aligned}$$

For instance, Fig. 3.5 depicts the $\mathcal{M}^p$ value of the sequence $X = \{0.01n\}_{n=1}^{100}$ when the p is varied within the interval $[-5, 5]$. In addition, we have plotted the cases above mentioned to observe how much the function varies.

Then, WMM can be understood as the generalization of Weighted Sum Models (WSM) and Weighted Product Models (WPM), since they are particular cases of the cases $p = 1$ and $p = 0$ respectively. Moreover, the cost function linked to the ranking function (WMM Step 2) is the p-norm of the sequence space ℓ^p. Therefore, we can establish a proposition that relates the p selected with the results in terms of norms.

Definition 3.2.1. *Given $0 < p < +\infty$ and a vector space $\mathbb{K}^{\mathbb{N}}$, the sequence space $\ell^p(\mathbb{K}^n)$ is the subspace of all the sequences $x = \{x_k\}_{k\in\mathbb{N}}$ so that*

$$\sum_{k\in\mathbb{N}} |x_k|^p < +\infty.$$

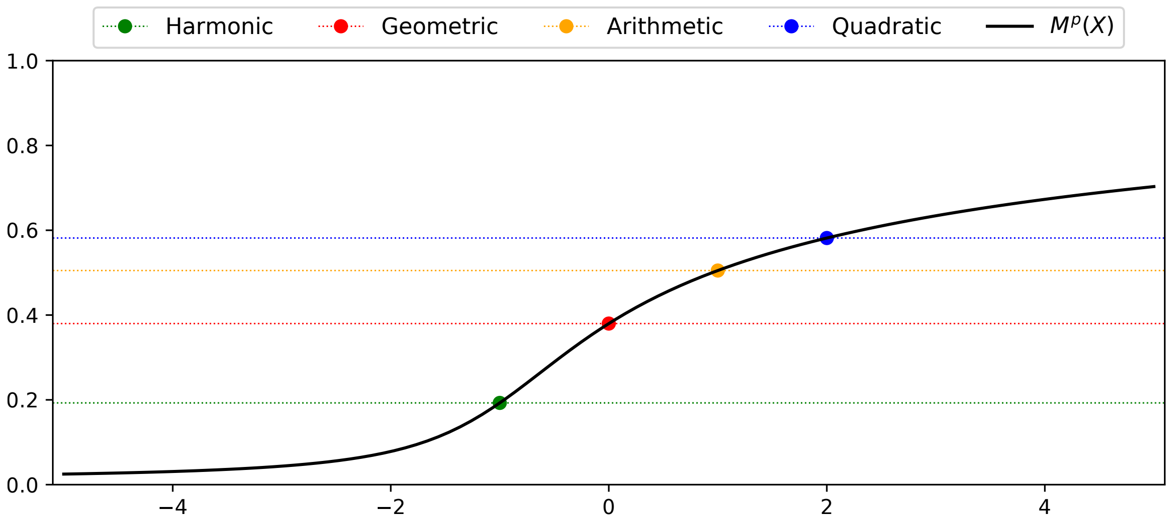


Figure 3.5: Image of the generalized p-mean of the $\{0.01n\}_{n=1}^{100}$ sequence with uniform weights. The X-axis represents the value of p and the Y-axis the image of $\mathcal{M}^p$. The examples highlighted with dots are the cases of $p \in \{-1, 0, 1, 2\}$.
Source: Own elaboration.

Then, per each $p \geq 1$, we can define the norm $||\cdot||_p : \mathbb{K}^{\mathbb{N}} \to \mathbb{R}$ operation as a real-valued application with formulation:

$$||x||_p = \left[\sum_{k\in\mathbb{N}} |x_k|^p\right]^{\frac{1}{p}}.$$

The space given by $\ell^p(\mathbb{K}^n)$ is a complete metric space for the norm $||\cdot||_p$, and so it is also a Banach space. For the particular case of MCDA, we solve problems defined in $[0,1]^M$, so we have a bounded space with a finite dimension.

Proposition 3.2.3. *Given two vectors $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ with elements in $\mathbb{R}$ or $\mathbb{C}$ and two real numbers p, q such that $\frac{1}{p} + \frac{1}{q} = 1$. Then,*

$$|\langle x, y\rangle| \leq ||x||_p ||y||_q. \tag{3.19}$$

Eq. 3.19 is known as Hölder's inequality and it generalizes the Cauchy–Schwarz inequality, obtained by making $p = q = 2$. As we are considering a finite n-dimensional Euclidean space, we can rewrite Eq. 3.19 as:

$$\sum_{i=1}^{n} |x_i y_i| \leq \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}} \left(\sum_{i=1}^{n} |y_i|^q\right)^{\frac{1}{q}}.$$

Proposition 3.2.4. *Given a vector $X = (x_1, \ldots, x_n)$ of elements in $\mathbb{R}$ or $\mathbb{C}$ and two real numbers p, q such that $1 \leq p \leq q$. Then,*

$$||X||_p \leq ||X||_q \leq N^{\frac{1}{p}-\frac{1}{q}} ||X||_p. \tag{3.20}$$

As a consequence of Eq. 3.20, if $p < q$ we can say that ℓ^p dominates ℓ^q. Thus, each space is embedded in a decreasing way.

Now, we can extend the concepts obtained by Propositions 3.2.3 and 3.2.4 in order to figure out some properties of the $\mathcal{M}^p(x, w)$ functions.

Proposition 3.2.5. *Let $X = (x_1, \ldots, x_n)$ be a vector in $\mathbb{K}^n$ and $w = (w_1, \ldots, w_n)$ their weighting values. Given two real numbers p, q such that $1 \leq p \leq q$. Then,*

$$\mathcal{M}^p(X, w) \leq \mathcal{M}^q(X, w), \tag{3.21}$$

and so

$$\frac{\partial}{\partial p}\mathcal{M}^p(X, w) \geq 0. \tag{3.22}$$

Proof.

$$\mathcal{M}^p(X, w) = \left[\sum_{j=1}^{M} w_j x_j^p\right]^{\frac{1}{p}} = \left[\sum_{j=1}^{M}\left(w_j^{\frac{1}{p}} x_j\right)^p\right]^{\frac{1}{p}} = \left|\left|w^{\frac{1}{p}} X\right|\right|_p .$$

□

We would like to emphasize the contribution of each component of the $\mathcal{M}_i^p$ formulation, we can note the variation of each value of the formula via partial differential calculus:

$$\begin{aligned}
\frac{\partial}{\partial x_{ij}}\mathcal{M}_i^p(X, w) &= w_j x_{ij}^{p-1}\left[\sum_{j=1}^{M} w_j x_{ij}^p\right]^{\frac{1}{p}-1} \\
\frac{\partial}{\partial w_j}\mathcal{M}_i^p(X, w) &= \frac{1}{p} x_{ij}^p \left[\sum_{j=1}^{M} w_j x_{ij}^p\right]^{\frac{1}{p}-1} \\
\frac{\partial}{\partial p}\mathcal{M}_i^p(X, w) &= \mathcal{M}_i^p(X, p)\left[\frac{\sum_{j=1}^{M} w_j x_{ij}^p \log(x_j)}{p\sum_{j=1}^{M} w_j x_{ij}^p} - \frac{\log\left(\sum_{j=1}^{M} w_j x_{ij}^p\right)}{p^2}\right]
\end{aligned} \tag{3.23}$$

In regards to the value of p, we can also interpret the p-distance between two vectors $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$, with $X, Y \in \mathbb{K}^n$ as

$$d_p(X, Y) = \left[\sum_{i=1}^{n}(x_i - y_i)^p\right]^{\frac{1}{p}} . \tag{3.24}$$

So we can basically reformulate Eq. 3.24 as:

$$d_p(X, Y) = n^p \cdot \mathcal{M}^p(X - Y, w), \quad \text{with } w_j = \tfrac{1}{n}, \quad \forall j \in \{1, \ldots, n\}. \tag{3.25}$$

A visual representation of the spheres resultant from fixing the origin and varying the radius and the p is depicted in Fig. 3.6.

The interpretation of the illustrated curves of Fig. 3.6 is very important in MCDA since it is formerly established as a distance for vector comparisons among the different alternatives. As a consequence, the sphere representation of such distance will determine *"how*

Table 3.2: List of the most used metrics in statistics for two non-null vectors $x, y \in \mathbb{R}^M$.

Distance function	**Formulation $\mathbf{d(x,y)}$**
Manhattan/Taxicab distance or d_1	$\sum_{i=1}^{M} \lvert x_i - y_i \rvert$
Euclidean distance or d_2	$\sqrt{\sum_{i=1}^{M} (x_i - y_i)^2}$
Minkowski distance or d_p	$\left[\sum_{i=1}^{M} \lvert x_i - y_i \rvert^p\right]^{\frac{1}{p}}$
Chebyshev distance or d_∞	$\max_{1 \leq i \leq M} \{\lvert x_i - y_i \rvert\}$
Mahalanobis distance or d_M	$\sqrt{(x-y)^\top M^{-1}(x-y)}$, M covariance matrix
Bray-Curtis distance	$\dfrac{\sum_{i=1}^{M} \lvert x_i - y_i \rvert}{\sum_{i=1}^{M} \lvert x_i + y_i \rvert}$
Canberra distance	$\dfrac{\sum_{i=1}^{M} \lvert x_i - y_i \rvert}{\sum_{i=1}^{M} \lvert x_i \rvert + \lvert y_i \rvert}$
Cosine distance	$1 - \dfrac{x \cdot y}{\lvert\lvert x \rvert\rvert_2 \cdot \lvert\lvert y \rvert\rvert_2}$
Correlation distance	$1 - \dfrac{(x - \mu_x)(y - \mu_y)}{\lvert\lvert x - \mu_x \rvert\rvert_2 \lvert\lvert y - \mu_y \rvert\rvert_2}$, being μ_x, μ_y means
Discrete distance	$\mathbf{1}_{\{x=y\}}(x,y) = \begin{cases} 1 & \text{if} \quad x \neq y \\ 0 & \text{if} \quad x = y \end{cases}$
Hamming (1950) distance	$\frac{1}{M} \sum_{i=1}^{M} \mathbf{1}_{\{x_i = y_i\}}(x_i, y_i)$
Jensen-Shannon divergence (Fuglede and Topsoe, 2004)	$\sqrt{\frac{1}{2}\left[D(x \lvert \mu) + D(y \lvert \mu)\right]}$ D as in Eq. 4.6 and μ the mean of $\frac{1}{2}(x+y)$

distant two alternatives are" or *"how closely an alternative is to some compromise solution"*. Then, it is commonly used in outranking methods for both alternative and attribute pairwise comparisons.

Even though it is very common to apply d_p distances in MCDM problems, it is also possible to resort to many other alternative spaces than the ℓ^p. It happens because sometimes the d_p metric over usual vector spaces is not suitable for local comparisons. Depending on the motivation of the DMs or the data handled, we can utilize the distance functions defined in Table 3.2.

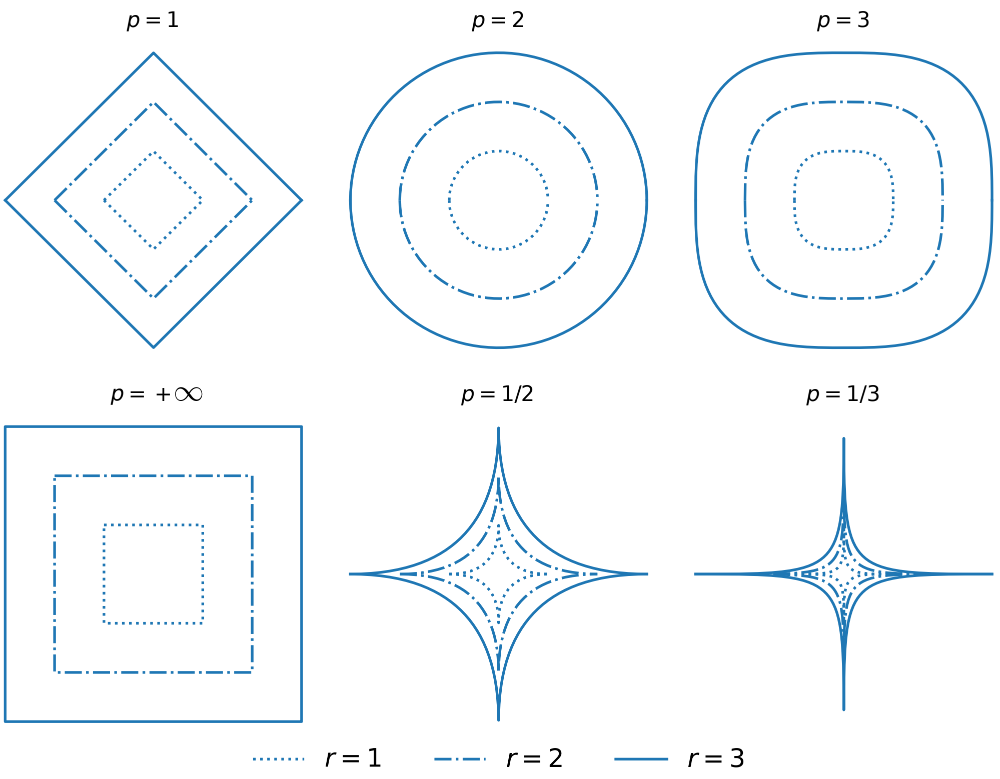


Figure 3.6: Sphere representations in $\mathbb{R}^2$ when we modify the radius (r) and the p of the distance function.

Source: Own elaboration.

3.2.2 *TOPSIS*

The Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) defined in Hwang and Yoon (1981) is a method to rank alternatives regarding positive and negative ideal solutions. With respect to the weighting scheme, TOPSIS evaluates the dataset by making full use of the selected criteria. For its application, we just need to consider a normalization and a vector distance. Then, it computes the relative proximity to the ideal solutions to provide a cardinal ranking of the alternatives in descending order. It was originally defined with the Euclidean distance and vector normalization as described in the following steps.

TOPSIS method:

Step 1 Normalize the decision matrix as $[r_{ij}]$, where each $r_{ij} \in [0,1]$ for each $i \in \{1,\dots,N\}$ and $j \in \{1,\dots,M\}$. The authors originally defined the method with the specific ℓ^2 vector normalization.

Step 2 Calculate the weighted normalized decision matrix $[v_{ij}]$ according to the set of weights $\{w_j\}_{j=1}^{M}$ such that $(w_1,...,w_M) \in \Omega$. Then, we have $v_{ij} = w_j r_{ij}$ for each $i \in \{1,\dots,N\}$ and $j \in \{1,\dots,M\}$.

Step 3 Positive ideal and negative ideal are determined as $PIS = (v_1^+,\dots,v_M^+)$ and $NIS = (v_1^-,\dots,v_M^-)$ where

$$v_j^+ = \begin{cases} \max\limits_{1\le i\le n} v_{ij} & \text{if} \quad j \in J_{max} \\ \min\limits_{1\le i\le n} v_{ij} & \text{if} \quad j \in J_{min} \end{cases} \qquad 1 \le j \le M,$$

$$v_j^- = \begin{cases} \min\limits_{1\le i\le n} v_{ij} & \text{if} \quad j \in J_{max} \\ \max\limits_{1\le i\le n} v_{ij} & \text{if} \quad j \in J_{min} \end{cases} \qquad 1 \le j \le M,$$

where J_{max} is the set of criteria to be maximized and J_{min} is the set of criteria to be minimized.

Step 4 Calculate the separation measures with regard to PIS and NIS:

$$D_i^+ = \left[\sum_{j=1}^{M}(v_{ij} - v_j^+)^p\right]^{\frac{1}{p}} \text{ and } D_i^- = \left[\sum_{j=1}^{M}(v_{ij} - v_j^-)^p\right]^{\frac{1}{p}}, \ 1 \le i \le N. \tag{3.26}$$

Originally, the distance selected is the Euclidean ($p = 2$) but it may vary depending on the problem.

Step 5 Calculate the relative proximity to the ideal solutions using the relative index:

$$R_i = \frac{D_i^-}{D_i^+ + D_i^-}, \ 1 \le i \le N. \tag{3.27}$$

Step 6 Make a ranking of the alternatives in descending order of the values of $\{R_i\}_{i=1}^{N}$.

TOPSIS transforms a set of features $N \times M$ dimensional into a family of N real numbers that yields decisive information about the global situation of the alternatives involved. Moreover, we could modify the Step 3 so that we get $PIS = (1,\dots,1)$ and $NIS = (0,\dots,0)$. Then, we would avoid the rank reversal problem since we are removing results dependent on data (Cables et al., 2016).

3.2.3 *VIKOR*

The *VIseKriterijumska Optimizacija I Kompromisno Resenje* (VIKOR) method defined in Opricovic (1998), in English translated as Multicriteria Optimization And Compromise

Solution, is an MCDM technique which ranks alternatives but emphasizing in the search of a compromise solution. By using a fixed normalization scheme, it makes use of the norms $\{1, +\infty\}$ per each alternative to defining two scores (S and R) to finally make a convex linear combination of them and use it as score function (Q) (Opricovic and Tzeng, 2004). Once, we have computed the Q-score per each alternative, we recursively define the number of alternatives to be considered in the compromise solution. The procedure to follow is described in the next steps.

VIKOR method:

Step 1 Determine the ideal solutions $PIS = (x_1^+, \dots, x_M^+)$ and $NIS = (x_1^-, \dots, x_M^-)$ per each criterion $j \in \{1, \dots, M\}$ as:

$$x_j^+ = \begin{cases} \max\limits_{1 \le i \le n} x_{ij} & \text{if} \quad j \in J_{max} \\ \min\limits_{1 \le i \le n} x_{ij} & \text{if} \quad j \in J_{min} \end{cases} \qquad 1 \le j \le M,$$

$$x_j^- = \begin{cases} \min\limits_{1 \le i \le n} x_{ij} & \text{if} \quad j \in J_{max} \\ \max\limits_{1 \le i \le n} x_{ij} & \text{if} \quad j \in J_{min} \end{cases} \qquad 1 \le j \le M,$$

where J_{max} is the set of criteria to be maximized and J_{min} is the set of criteria to be minimized.

Step 2 Normalize the decision matrix as $[r_{ij}]$ per each $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, M\}$, where:

$$r_{ij} = \frac{x^+ - x_{ij}}{x^+ - x^-}. \tag{3.28}$$

Step 3 Compute the S and R scores per each alternative $i \in \{1, \dots, N\}$:

$$S_i = \sum_{j=1}^{M} w_j r_{ij} \quad \text{Utility measure.} \tag{3.29}$$

$$R_i = \max_{1 \le j \le M} \{w_j r_{ij}\} \quad \text{Regret measure.} \tag{3.30}$$

Step 4 Calculate the Q-score as the convex combination according to the maximum group utility parameter $\nu \in [0, 1]$:

$$Q_i = \nu \frac{S_i - S^-}{S^+ - S^-} + (1 - \nu) \frac{R_i - R^-}{R^+ - R^-} \ , \ 1 \le i \le N, \tag{3.31}$$

where:

$$S^- = \min_{1 \le i \le N} S_i \ , \quad R^- = \min_{1 \le i \le N} R_i,$$
$$S^+ = \max_{1 \le i \le N} S_i \ , \quad R^+ = \max_{1 \le i \le N} R_i.$$

Step 5 Sort the vectors (S, R, Q) in ascending order to get the arranged arguments per each score. We consider the sorting arguments as $\{\sigma_S, \sigma_R, \sigma_Q\}$ respectively. Then, we can say that the alternative $a_{\sigma_Q(1)}$ is a compromise solution if satisfies the following statements:

a) Acceptable advantage: $Q_{\sigma_Q(2)} - Q_{\sigma_Q(1)} \geq \dfrac{1}{M-1}$.

b) Acceptable stability: $\sigma_Q(1) = \sigma_S(1)$ or $\sigma_Q(1) = \sigma_R(1)$.

In case both conditions are satisfied, we can say that the compromise solution is *"stable within a decision-making process"* if $v > 0.5$, or *"by consensus"* if $v = 0.5 \pm \epsilon$, or *"with veto"* if $v < 0.5$.

When one of the conditions is not satisfied, we can propose a new scheme of compromise solutions. If the solution has no stability (Step 5b is false), we need to select k alternatives so that $Q_{\sigma_Q(k)} - Q_{\sigma_Q(1)} \geq \frac{1}{M-1}$, then $(a_{\sigma_Q(1)}, \ldots, a_{\sigma_Q(k)})$ would be compromise solution. Conversely, if we have no advantage (Step 5a is false), we can select $(a_{\sigma_Q(1)}, a_{\sigma_Q(2)})$ as compromise solution.

In 1990, Opricovic presented the VIKOR method as an MCDM outranking strategy to rank alternatives based on the search for a feasible compromise solution (Opricovic, 1998). From then on, many variations of the method with multiple perspectives have arisen (Mardani et al., 2016). Opricovic and Tzeng (2003a) introduced crisp sets to solve conflicting cases of fuzzy approaches for defuzzification methods, later on, Opricovic and Tzeng (2003b) added strategies with incomplete information. A scheme for compromise solutions under fuzzy logic was written in Opricovic (2007). For membership of the weights, Devi (2011) added triangular fuzzy numbers to give the intuition of feasible importance. We can also find papers that extend the VIKOR problem to interval numbers in Sayadi et al. (2009). Finally, there are combinations of VIKOR methodology with other fields such as gray relational analysis (Kuo and Liang, 2011) and group decision making (Park et al., 2011).

3.3 FUZZY APPROACH

In 1965, Zadeh proposed the concept of fuzzy mathematics with the aim of controlling the uncertainty that involves real-world events. The idea of Zadeh was to study such imprecise events by making use of new mathematical sets, called fuzzy sets (Zadeh, 1996). Since the beginning of fuzzy logic, we can find an enormous list of literature linked to the topic in so many different research areas, which suggests that most applied science deals with unstable factors to take into account (Bojadziev and Bojadziev, 1995; Mittal et al., 2020). Moreover, what makes fuzzy logic a very interesting approach is that not only external sources determine what is non-accurate but also the human perception of what we think or reason about something specific.

Fuzzy logic also attempts to give a response to vague or shallow statements that are commonly extended among people, and so, among decision-makers. We usually refer to subjective terms as if they were standards for everybody, but it is known that human insights rely on context, culture, and background. For example, if I say that Juan is successful, would everyone coincide with me? It is obvious that it depends on the person, but in any case, my perception would be the same regardless of the people I am talking to. Given that human brains have their particular way of reasoning, Zadeh attempted to model that behavior with the construction of fuzzy memberships, which basically ascertain the truth value of a statement. That assigned degree of membership indicates the truthfulness of the event. It is typically represented in the $[0, 1]$ interval so that the 0 value represents a totally false case and the 1 value is an absolute truth. Thus, all the range between 0 and 1 indicates an intermediate degree of truth, in a way that partial truth can be analyzed and also accepted when appropriate.

In general, the incorporation of fuzzy logic allows us to control and limit the degree of reliability of the case study, hence we can take advantage of uncertainty when the conditions and hypothesis are well stated. To be more precise, this theory formalizes the notion of human reasoning in a mathematical way. People usually deal with the absolute truth of fake statements, therefore a large number of daily decisions are made under conflicting and incomplete information. When facing an environment of imprecision it is not an easy task to realize which of the possible alternatives will lead us to our goal or, at least, to an acceptable final stage. Additionally, the capabilities of human reasoning have some limitations when mixing mental and physical concepts. That is why Zadeh proposed a methodology to perform accurate strategies under uncertain conditions. In fact, a powerful approach in fuzzy logic is the transformation from subjectivity to measurable results.

In many real-world applications, uncertainty takes over the data selection procedure. Sometimes it happens when the measurements cannot be as accurate as it is required, but in many other cases, non-observed events indirectly affect the extraction process that we are analyzing. It is not necessarily related to the kind of data we are handling, but the degree of subjectivity related to the experiment. For values, we can distinguish between singular values, which are numerically formulated, and granular values, which can be obtained from words, signals, colors, etc. So far, we can transform non-numerical values (granular) to real stated numbers (singular) by means of scales. However, we would be losing information since we are assuming the lack of intermediate values, and so it means an imprecise interpretation of reality. Besides, the temporal behavior, the magnitudes, or the physical context does not have to follow a monotone pattern, which introduces the concept of crisp fuzzy sets. In either case, decision-makers need to take all these details into consideration since diffuse data may lead to non-significative results, especially when it is not considered in the hypothesis of the model. Another problem that is overcome with fuzzy logic is the measurement study of an event through membership functions. We all agree that probability is focused on measuring how likely is something to occur, however, probability functions cannot attach the degree or impact of an event. For example, with probability, we can estimate how likely is tomorrow to snow, but we cannot answer the question "How cold will it get tomorrow?". Therefore, fuzzy logic can be understood as a way to complement statistics and probability because of the concept of degrees of truth.

In the field of multi-objective optimization, Zimmermann (1978) proposed the notion of fuzzy-linear programming (FLP) based on linear fuzzy constraints and k linear objective functions as:

$$\begin{aligned} \text{minimize} \quad & \mathbf{z}(x) = [z_1(x), \dots, z_k(x)] \\ \text{s. t.} \quad & Ax \leq b, \\ & x \geq 0. \end{aligned} \tag{3.32}$$

The k objective functions are formulated as $z_i(x) = c_i x$ per each $i \in \{1, \dots, k\}$. Then, each goal is a linear fuzzy cost function. For multicriteria optimization, the DMs associate an acceptable value of membership through the functions:

$$\mu_i^L(z_i(x)) = \begin{cases} 0 & \text{if} \quad z_i(x) \geq z_i^0, \\ \dfrac{z_i(x) - z_i^0}{z_i^1 - z_i^0} & \text{if} \quad z_i^0 \leq z_i(x) \leq z_i^1, \\ 1 & \text{if} \quad z_i(x) \leq z_i^1, \end{cases} \tag{3.33}$$

That set of functions establishes the degree of membership of each i-objective function and each pair z_i^0 and z_i^1 indicate the limits where the uncertain events are measured (Zimmermann, 2001).

Based on the model presented in Eq. 3.32, Bellman and Zadeh (1970) reformulated the FLP problem as:

$$\begin{aligned} \text{maximize} \quad & \min_{1 \leq i \leq k} \left\{ \mu_i^L(z_i(x)) \right\} \\ \text{s. t.} \quad & Ax \leq b, \\ & x \geq 0. \end{aligned} \tag{3.34}$$

From this point on, the development of fuzzy optimization was considerably raised and it was not just focused on obtaining solutions, but on theories of the existence and consistency of such solutions. Additionally, it began the incorporation of fuzzy sets to both attributes and constraints.

As far as multicriteria optimization is concerned, dealing with such a problem is troublesome because the mathematical formalization of the objective is very sensitive to precision errors. In addition, the error propagation during the computational process would return a non-representative output. As a response to solving such concerns, Fuzzy Multiple-Criteria Decision Making (FMCDM) emerged to provide algorithms and tools to tackle non-precise data or shallowly defined criteria.

Definition 3.3.1. (Zadeh, 1965) *Let $\mathscr{X}$ be a space of points or objects. A fuzzy set A in $\mathscr{X}$ is characterized by a membership function μ_A which associates with each point x in $\mathscr{X}$ a real number in the interval $[0, 1]$, with the value of μ_A at x representing the "grade of membership" of x in A.*

Depending on the grade of a given $x \in A$, we can say that x is:

$$\begin{array}{lll} \text{Not included} & \text{if} & \mu_A(x) = 0, \\ \text{Partially included} & \text{if} & 0 \leq \mu_A(x) \leq 0, \\ \text{Fully included} & \text{if} & \mu_A(x) = 0, \end{array}$$

in relation to the fuzzy set (A, μ_A).

Definition 3.3.2. *Let (A, μ_A) be a fuzzy set and $0 \leq \alpha \leq 1$. We define the following sets:*

$$\begin{array}{ll} \alpha - cut: & A_\alpha = \{x \in A | \mu_A(x) \geq \alpha\}, \\ Support: & Supp(A) = \{x \in A | \mu_A(x) > 0\}, \\ Core: & Core(A) = \{x \in A | \mu_A(x) = 1\}. \end{array}$$

Definition 3.3.3. (Dubois and Prade, 1978) *A LR-fuzzy number $\tilde{X}$ is a 4 component array $\tilde{X} = (x^L, x^R, \alpha^L, \alpha^R)_{L,R}$ whose membership function has the following form:*

$$\mu_{\tilde{X}}(t) = \begin{cases} L\left(\dfrac{x^L - t}{\alpha^L}\right) & \text{if} \quad t < x^L, \\ 1 & \text{if} \quad x^L \leq r \leq x^R, \\ R\left(\dfrac{t - x^R}{\alpha^R}\right) & \text{if} \quad t > x^R, \end{cases} \tag{3.35}$$

where $L, R : \mathbb{R}^+ \to [0, 1]$ are the so-called reference functions which are decreasing in the support of $\tilde{X}$ and upper semi-continuous with $L(0) = R(0) = 1$.

In most cases, the LR-fuzzy representation is denoted as trapezoidal numbers, that is to say when the spread of each side can be formulated by linear interpolation.

$$L(t) = \begin{cases} \dfrac{t - \alpha^L}{x^L - \alpha^L} & \text{if} \qquad \alpha^L \leq t \leq x^L, \\ 0 & \text{otherwise.} \end{cases}$$

$$R(t) = \begin{cases} \dfrac{\alpha^R - t}{\alpha^R - x^R} & \text{if} \qquad x^R \leq t \leq \alpha^R, \\ 0 & \text{otherwise.} \end{cases}$$

The nature of the problem defines the structure of the number $\tilde{X}$. Fig. 3.7 depicts six common cases in fuzzy methodology. It is easy to note that we can formalize the notion of regular numbers by means of fuzzy logic, we just have to set the number with neither spread nor interval. In mathematical terms, $(x, x, 0, 0)$ is the LR-fuzzy form for any real number x. Moreover, with regard to the membership function, fuzzy logic also generalizes the notion of the characteristic function of any subset. Among many examples, we can highlight the Heaviside step, Kronecker delta, and Dirac delta functions.

Given two LR-fuzzy numbers $\tilde{X}_1$ and $\tilde{X}_2$ with bounded support and non-null components, we can define some basic operations for fuzzy theory. For compatibility purposes, we assume that all the components of both numbers are positive numbers. Moreover, we consider an increasing monotonic non-zero function F defined in $\tilde{X}_1$ and a positive scalar $\lambda \in \mathbb{R}^+$.

$$
\begin{array}{lrcl}
\text{Addition:} & \tilde{X}_1 \oplus \tilde{X}_2 & = & (x_1^L + x_2^L,\ x_1^R + x_2^R,\ \alpha_1^L + \alpha_2^L,\ \alpha_1^R + \alpha_2^R)_{L_1+L_2,R_1+R_2}, \\
\text{Scalar product:} & \lambda \odot \tilde{X}_1 & = & (\lambda x^L, \lambda x^R, \lambda \alpha^L, \lambda \alpha^R)_{L,R}, \\
\text{Fuzzy product:} & \tilde{X}_1 \otimes \tilde{X}_2 & = & (x_1^L x_2^L,\ x_1^R x_2^R,\ \alpha_1^L \alpha_2^L,\ \alpha_1^R \alpha_2^R)_{L_1 L_2, R_1 R_2}, \\
\text{Negation:} & \ominus \tilde{X}_1 & = & (-x_1^R,\ -x_1^L,\ -\alpha_1^R,\ -\alpha_1^L)_{R_1,L_1}, \\
\text{Inversion:} & 1/\tilde{X}_1 & = & (1/x_1^R,\ 1/x_1^L,\ 1/\alpha_1^R,\ 1/\alpha_1^L)_{R_1,L_1}, \\
\text{Division:} & \tilde{X}_2 \oslash \tilde{X}_1 & = & \tilde{X}_2 \otimes 1/\tilde{X}_1, \\
\text{Application:} & F(\tilde{X}_1) & = & (F(x^L),\ F(x^R),\ F(\alpha^L),\ F(\alpha^R))_{L,R}.
\end{array}
$$

Now, instead of using the LR-fuzzy numbers, we can define some basic operators for their respective membership functions $\mu_{\tilde{X}_1}$ and $\mu_{\tilde{X}_2}$:

$$
\begin{array}{lrcl}
\text{Intersection:} & \mu_{\tilde{X}_1 \cap \tilde{X}_2}(t) & = & \min\{\mu_{\tilde{X}_1}(t_1), \mu_{\tilde{X}_1}(t_2)\}, \\
\text{Union:} & \mu_{\tilde{X}_1 \cup \tilde{X}_2}(t) & = & \max\{\mu_{\tilde{X}_1}(t_1), \mu_{\tilde{X}_1}(t_2)\}, \\
\text{Complement:} & \mu_{\tilde{X}_1^{\mathsf{C}}}(t) & = & 1 - \mu_{\tilde{X}_1}(t).
\end{array}
$$

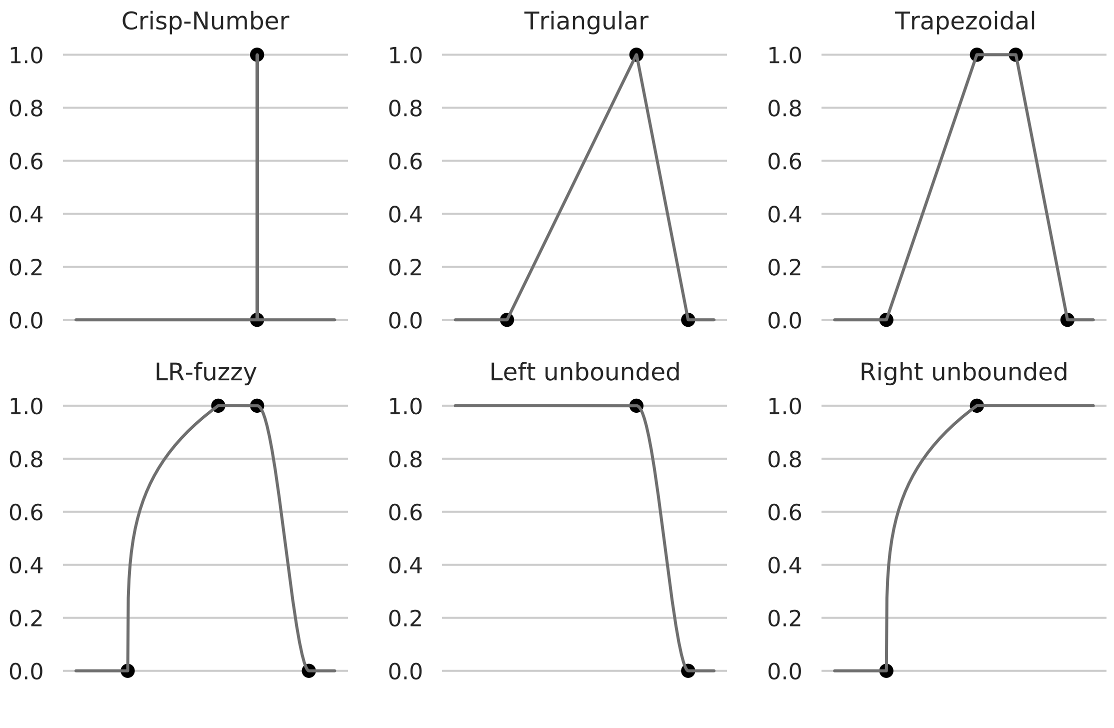


Figure 3.7: Different representations of LR-fuzzy numbers $\tilde{X} = (x^L, x^R, \alpha^L, \alpha^R)_{L,R}$.
Source: Own elaboration.

Definition 3.3.4. *Let $\tilde{X}$ and $\tilde{Y}$ be two LR-fuzzy numbers. Then, we can define the $\tilde{X} \vee \tilde{Y}$ and $\tilde{X} \wedge \tilde{Y}$ fuzzy numbers whose membership function are:*

$$\mu_{\tilde{X}\vee\tilde{Y}}(t) = \sup_{s,t}\{\mu_{\tilde{X}}(t_1) \wedge \mu_{\tilde{Y}}(t_2)\},$$
$$\mu_{\tilde{X}\vee\tilde{Y}}(t) = \inf_{s,t}\{\mu_{\tilde{X}}(t_1) \vee \mu_{\tilde{Y}}(t_2)\},$$

respectively.

Definition 3.3.5. *Let $\tilde{X}$ be an LR-fuzzy number. We say that $\tilde{X}$ is normal if and only if*

$$\sup_t \mu_{\tilde{X}}(t) = 1.$$

Definition 3.3.6. *Let $\tilde{X}$ be an LR-fuzzy number. We say that $\tilde{X}$ is convex if per each t_1 and t_2 its membership function satisfies:*

$$\mu_{\tilde{X}}\left(\lambda t_1 + (1-\lambda)t_2\right) \geq \min\left\{\mu_{\tilde{X}}(t_1), \mu_{\tilde{X}}(t_2)\right\}, \ \forall \lambda \in [0,1].$$

Definition 3.3.7. *Let $\tilde{X}$ and $\tilde{Y}$ be two fuzzy numbers. Then,*

$$\tilde{X} \gtrsim \tilde{Y} \longleftrightarrow \tilde{X} \vee \tilde{Y} = \tilde{X}, \tag{3.36}$$

where $\vee$ represents the logical maximum generator.

The last stated definition may seem complete and robust, however, its applicability can cause problems of irresolution (Dubois et al., 2000). By considering the ideas of Tanaka (1984) and Inuiguchi et al. (1990), we can state a flexible version of such a definition to avoid that concern. First, we require a condition for joint fuzzy sets as in Ramík and ímánek (1985).

Lemma 3.3.1. *Let $\tilde{X}$ and $\tilde{Y}$ be two LR-fuzzy numbers. Then, $\tilde{X} \vee \tilde{Y} = \tilde{X}$ if, and only if, per each $0 \leq h \leq 1$ the two statements are satisfied:*

$$\inf\{s|\mu_{\tilde{X}}(s) \geq h\} \geq \inf\{t|\mu_{\tilde{Y}}(t) \geq h\},$$
$$\sup\{s|\mu_{\tilde{X}}(s) \geq h\} \geq \sup\{t|\mu_{\tilde{Y}}(t) \geq h\}. \tag{3.37}$$

In other words, we can rewrite the two expressions in Eq. 3.37 above as:

$$x^L - L^*(h)\alpha^L \geq y^L - L'^*(h)\beta^L \quad \forall h \in [0,1],$$
$$x^R + R^*(h)\alpha^R \geq y^R - R'^*(h)\beta^R \quad \forall h \in [0,1], \tag{3.38}$$

where $\tilde{X} = (x^L, x^R, \alpha^L, \alpha^R)_{L,R}$ and $\tilde{Y} = (y^L, y^R, \beta^L, \beta^R)_{L',R'}$ and the new reference functions are:

$$L^*(h) = \sup\{t|L(t) \geq h\}, \quad L'^*(h) = \sup\{t|L'(t) \geq h\},$$
$$R^*(h) = \sup\{t|R(t) \geq h\}, \quad R'^*(h) = \sup\{t|R'(t) \geq h\}.$$

In the particular case of similar reference functions, i.e. $L = L'$ and $R = R'$, we can reconvert the Eq. 3.38 to:

$$\begin{aligned} x^L \geq Y^L \quad &\text{and} \quad x^L - \alpha^L \geq Y^L - \beta^L, \\ X^R \geq Y^R \quad &\text{and} \quad x^R + \alpha^R \geq Y^R + \beta^R. \end{aligned} \tag{3.39}$$

Definition 3.3.8. *Let $\tilde{X}$ and $\tilde{Y}$ be two LR-fuzzy numbers and a real number $0 \leq h \leq 1$. Then, $\tilde{X} \gtrsim^h \tilde{Y} = \tilde{X}$ if, and only if, the two statements below hold true:*

$$\begin{aligned} \inf\{s|\mu_{\tilde{X}}(s) \geq k\} \geq \inf\{t|\mu_{\tilde{Y}}(t) \geq k\}, \\ \sup\{s|\mu_{\tilde{X}}(s) \geq k\} \geq \sup\{t|\mu_{\tilde{Y}}(t) \geq k\}. \end{aligned} \tag{3.40}$$

per each $k \in [h, 1]$.

In practice, when working with LR-fuzzy numbers with bounded support, we could modify the Definition 3.3.8 and maintain that $\tilde{X}$ is h-greater than $\tilde{Y}$ whenever:

$$\begin{aligned} x^L - L^*(k)\alpha^L \geq y^L - L'^*(k)\beta^L \quad \forall h \in [0,1], \\ x^R + R^*(k)\alpha^R \geq y^R - R'^*(k)\beta^R \quad \forall h \in [0,1]. \end{aligned} \tag{3.41}$$

In such a case, note that we have formulated a less restrictive condition in comparison with the statements of Lemma 3.3.1. Therefore, we have a more flexible strategy to compare two LR-fuzzy numbers. See León et al. (2003), for instance.

When combining fuzzy logic and MCDM, the problem is performed by determining the fuzzy decision matrix $\tilde{X} = [\tilde{x}_{ij}]$, so that $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$. Now, we have a trapezoidal LR-fuzzy representation per each $\tilde{x}_{ij} = (x_{ij}^L, x_{ij}^R, \alpha_{ij}^L, \alpha_{ij}^R)_{L_{ij}R_{ij}}$, in which it is represented by the lower and upper spread of each magnitude. Regarding the set of weights, we can determine a fuzzy-weighting scheme $\tilde{\Omega}$, so that if $(w^L, w^R, \beta^L, \beta^R)_{L'R'} \in \tilde{\Omega}$ then:

$$\beta^L(r) \geq 0 \quad \text{and} \quad \beta^R(r) \leq 1, \text{ per each } r \in \mathbb{R}^+. \tag{3.42}$$

3.3.1 *Fuzzy-WMM*

The Fuzzy Weighted Mean Models (FWMM) is the WMM version to deal with fuzzy numbers (Triantaphyllou, 2000). By means of the definitions and propositions previously stated, we can convert the operations of the $\mathcal{M}^p$ functions into a reshaped case for functional purposes.

Fuzzy-WMM method:

Step 1 Determine the fuzzy decision matrix $[\tilde{x}_{ij}]$ so that $\tilde{x}_{ij} = (x_{ij}^L, x_{ij}^R, \alpha_{ij}^L, \alpha_{ij}^R)_{L_{ij}R_{ij}}$ and define the fuzzy weights $\tilde{w}_j = (w_j^L, w_j^R, \beta_j^L, \beta_j^R)_{L'_jR'_j}$, per each $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$.

Step 2 Normalize the fuzzy decision matrix as $[\tilde{r}_{ij}]$, per each $i \in \{1,\dots,N\}$ and $j \in \{1,\dots,M\}$.

Step 3 Calculate the weighted p-mean of each alternative:

$$\tilde{\mathcal{M}}_i^p = \mathcal{M}^p(\tilde{r}_{ij},\tilde{w}) = \left[\sum_{j=1}^{M} \tilde{w}_j \otimes \tilde{r}_{ij}^p\right]^{\frac{1}{p}}, \quad 1 \le i \le N, \tag{3.43}$$

in which the product $\tilde{w}_j \otimes \tilde{r}_{ij}^p$ is computed as $(w_j^L x_{ij}^L,\ w_j^R x_{ij}^R,\ \beta_j^L \alpha_{ij}^L,\ \beta_j^R \alpha_{ij}^R)_{L'L,R'R}$ and the summation is conducted with the $\oplus$ operator.

Step 4 By means of a fuzzy order relationship, make a ranking of the alternatives in descending order of the values of $\{\tilde{\mathcal{M}}_i^p\}_{i=1}^N$.

For further extensions and applications see, for instance, Triantaphyllou (2000).

3.3.2 *Fuzzy-TOPSIS*

The fuzzy version of TOPSIS (FTOPSIS) was presented in Chen and Hwang (1992). Its different approaches and implementations have been widely studied (Chen, 2000; Chen and Lee, 2010; Chen and Tsao, 2008) due to the large popularity of this ranking technique (Triantaphyllou, 2000). The development of this technique involves the following steps.

Fuzzy-TOPSIS method:

Step 1 Determine the fuzzy decision matrix $[\tilde{x}_{ij}]$ so that $\tilde{x}_{ij} = (x_{ij}^L, x_{ij}^R, \alpha_{ij}^L, \alpha_{ij}^R)_{L_{ij}R_{ij}}$ and define the fuzzy weights $\tilde{w}_j = (w_j^L, w_j^R, \beta_j^L, \beta_j^R)_{L'_jR'_j}$, per each $i \in \{1,\dots,N\}$ and $j \in \{1,\dots,M\}$.

Step 2 Normalize the fuzzy decision matrix as $[\tilde{r}_{ij}]$, for each $i \in \{1,\dots,N\}$ and $j \in \{1,\dots,M\}$.

Step 3 Calculate the fuzzy weighted normalized decision matrix $[\tilde{v}_{ij}]$ according to the set of fuzzy weights $\{\tilde{w}_j\}_{j=1}^M$. Then, we have $\tilde{v}_{ij} = \tilde{w}_j \otimes \tilde{r}_{ij}$ for each $i \in \{1,\dots,N\}$ and $j \in \{1,\dots,M\}$.

Step 4 Positive ideal and negative ideal are determined as $\widetilde{PIS} = (\tilde{v}_1^+,\dots,\tilde{v}_M^+)$ and $\widetilde{NIS} = (\tilde{v}_1^-,\dots,\tilde{v}_M^-)$ where

$$\tilde{v}_j^+ = \begin{cases} \widetilde{\max}_{1\le i\le n} \{\tilde{v}_{ij}\} & \text{if} \quad j \in J_{max} \\ \widetilde{\min}_{1\le i\le n} \{\tilde{v}_{ij}\} & \text{if} \quad j \in J_{min} \end{cases} \qquad 1 \le j \le M,$$

$$\tilde{v}_j^- = \begin{cases} \widetilde{\min}_{1\le i\le n} \{\tilde{v}_{ij}\} & \text{if} \quad j \in J_{max} \\ \widetilde{\max}_{1\le i\le n} \{\tilde{v}_{ij}\} & \text{if} \quad j \in J_{min} \end{cases} \qquad 1 \le j \le M,$$

where J_{max}, J_{min} are the sets of criteria to be maximized/minimized and $\widetilde{\max}, \widetilde{\min}$ stands for the $\max, \min$ operators per fuzzy component.

Step 5 Calculate the separation measures with regard to $\widetilde{PIS}$ and $\widetilde{NIS}$.

$$\tilde{D}_i^+ = \left[\sum_{j=1}^{M}(\tilde{v}_{ij} \ominus \tilde{v}_j^+)^p\right]^{\frac{1}{p}} \text{ and } \tilde{D}_i^- = \left[\sum_{j=1}^{M}(\tilde{v}_{ij} \ominus \tilde{v}_j^-)^p\right]^{\frac{1}{p}}, \ 1 \leq i \leq N. \tag{3.44}$$

Originally, the distance selected is the Euclidean ($p = 2$) but it may vary depending on the problem.

Step 6 Calculate the relative proximity to the ideal solutions using the relative index:

$$\tilde{R}_i = \frac{\tilde{D}_i^-}{\tilde{D}_i^+ \oplus \tilde{D}_i^-}, \ 1 \leq i \leq N. \tag{3.45}$$

Step 7 By means of a fuzzy order relationship, make a ranking of the alternatives in descending order of the values of $\{\tilde{R}_i\}_{i=1}^N$.

For further extensions and applications see, for instance, Nădăban et al. (2016) or Salih et al. (2019).

3.3.3 *Fuzzy-VIKOR*

The fuzzy version of the VIKOR method (FVIKOR) was presented in Opricovic (2011), although it was first studied by Sanayei et al. (2010). The main purpose was to determine the compromise solution of a fuzzy multicriteria problem. The procedure to perform the ranking algorithm has the following steps.

Fuzzy-VIKOR method:

Step 1 Determine the fuzzy decision matrix $[\tilde{x}_{ij}]$ so that $\tilde{x}_{ij} = (x_{ij}^L, x_{ij}^R, \alpha_{ij}^L, \alpha_{ij}^R)_{L_{ij}R_{ij}}$ and define the fuzzy weights $\tilde{w}_j = (w_j^L, w_j^R, \beta_j^L, \beta_j^R)_{L_j'R_j'}$, per each $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$.

Step 2 Determine the ideal solutions $\widetilde{PIS} = (\tilde{x}_1^+, \ldots, \tilde{x}_M^+)$ and $\widetilde{NIS} = (\tilde{x}_1^-, \ldots, \tilde{x}_M^-)$ per each criterion $j \in \{1, \ldots, M\}$ where:

$$\tilde{x}_j^+ = \begin{cases} \widetilde{\max}_{1\leq i\leq n} \{\tilde{x}_{ij}\} & \text{if } j \in J_{max} \\ \widetilde{\min}_{1\leq i\leq n} \{\tilde{x}_{ij}\} & \text{if } j \in J_{min} \end{cases} \quad 1 \leq j \leq M,$$

$$\tilde{x}_j^- = \begin{cases} \widetilde{\min}_{1\leq i\leq n} \{\tilde{x}_{ij}\} & \text{if } j \in J_{max} \\ \widetilde{\max}_{1\leq i\leq n} \{\tilde{x}_{ij}\} & \text{if } j \in J_{min} \end{cases} \quad 1 \leq j \leq M,$$

where J_{max}, J_{min} are the sets of criteria to be maximized/minimized and $\widetilde{\max}, \widetilde{\min}$ stands for the $\max, \min$ operators per fuzzy component.

Step 3 Normalize the decision matrix as $[\tilde{r}_{ij}]$per each $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$, where:

$$\tilde{r}_{ij} = \frac{\tilde{x}^+ \ominus \tilde{x}_{ij}}{\tilde{x}^+ \ominus \tilde{x}^-}. \tag{3.46}$$

Step 4 Compute the S and R scores per each alternative $i \in \{1, \dots, N\}$:

$$\tilde{S}_i = \sum_{j=1}^{M} \tilde{w}_j \otimes \tilde{r}_{ij} \quad \text{Utility measure.} \tag{3.47}$$

$$\tilde{R}_i = \widetilde{\max_{1 \le j \le M}} \{\tilde{w}_j \otimes \tilde{r}_{ij}\} \quad \text{Regret measure.} \tag{3.48}$$

Step 5 Calculate the $\tilde{Q}$-score as the convex combination according to the maximum group utility parameter $\nu \in [0, 1]$:

$$\tilde{Q}_i = \nu \cdot \frac{\tilde{S}_i \ominus \tilde{S}^-}{\tilde{S}^+ \ominus \tilde{S}^-} \oplus (1 - \nu) \cdot \frac{\tilde{R}_i \ominus \tilde{R}^-}{\tilde{R}^+ \ominus \tilde{R}^-} \ , \ 1 \le i \le N, \tag{3.49}$$

where:

$$\tilde{S}^- = \widetilde{\min_{1 \le i \le N}} \{\tilde{S}_i\} \ , \quad \tilde{R}^- = \widetilde{\min_{1 \le i \le N}} \{\tilde{R}_i\},$$
$$\tilde{S}^+ = \widetilde{\max_{1 \le i \le N}} \{\tilde{S}_i\} \ , \quad \tilde{R}^+ = \widetilde{\max_{1 \le i \le N}} \{\tilde{R}_i\}.$$

Step 6 Sort the vectors $(\tilde{S}, \tilde{R}, \tilde{Q})$ in ascending fuzzy order to get the arranged arguments per each score by means of a defuzzification method (Leekwijck and Kerre, 1999) to get (S, R, Q). We consider the sorting arguments as $\{\sigma_S, \sigma_R, \sigma_Q\}$ respectively. Then, we can say that the alternative $a_{\sigma_Q(1)}$ is a compromise solution if satisfies the following statements:

a) Acceptable advantage:

$$\frac{Q_{\sigma_Q(2)} - Q_{\sigma_Q(1)}}{Q_{\sigma_Q(M)} - Q_{\sigma_Q(1)}} \ge \frac{1}{M-1} \quad \text{Opricovic (2011) version.} \tag{3.50}$$

$$Q_{\sigma_Q(2)} - Q_{\sigma_Q(1)} \ge \frac{1}{M-1} \quad \text{Sanayei et al. (2010) version.} \tag{3.51}$$

b) Acceptable stability: $\sigma_Q(1) = \sigma_S(1)$ or $\sigma_Q(1) = \sigma_R(1)$.

When one of the conditions is not satisfied, we can propose a new scheme of compromise solutions. If the solution has no stability (Step 6b is false), we need to select k alternatives so that $Q_{\sigma_Q(k)} - Q_{\sigma_Q(1)} \ge \frac{1}{M-1}$, then $(a_{\sigma_Q(1)}, \dots, a_{\sigma_Q(k)})$ would be compromise solution. Conversely, if we have no advantage (Step 6a is false), we can select $\left(a_{\sigma_Q(1)}, a_{\sigma_Q(2)}\right)$ as compromise solution.

For further extensions and applications see, for instance, Gul et al. (2016).

3.4 UNWEIGHTED APPROACHES

As we have mentioned in the last two sections, multi-attribute optimization applied for aiding in the decision-making processes allows us to sort a set of alternatives. As a result, we are ready to select or exclude (or maybe both) some number of alternatives, which is very useful and supportive within prescriptive analytics. So far, we just have to decide which method is the best suited for our case study, thus accepting whether the implementation is fuzzy-based or not, and determining the weighting scheme that fits the behavior of our event. The first step may seem very complicated due to the mathematical background required to understand the arrangement strategy, nonetheless, the result is strictly based on such statements. Then, we can agree that the selection of the method is rather objective since it is complemented by theoretical properties. When referring to the selection of weights, it is not easy to determine their values and avoid conflict among decision-makers. DMs usually add their bias to the MCDM procedure, basically because they are expected to be experts in the field. In any case, non-objective ratings may lead to unfair resolutions.

For the last comment concerned, it is known that one of the major shortcomings of the MCDM is the choice of weights. They define the relative importance of each criterion in the decision phase of the algorithm. As we emphasized at the beginning of the section, for decision-makers, it is a hard task to select a scheme that leads to a common agreement. Each individual choice should be studied meticulously. However, an objective selection could not meet the requirements of the DM. But, in any case, the output of any set of weights does not give us meaningful properties and conditions of the optimal of each alternative over the criteria predetermined. It is easy to notice that their selection directly alters the resultant ranking, hence the weighted scheme has to be not only balanced but as objective as possible.

Let $\Omega_{lu} = \{w \in [0,1]^M | \sum_{j=1}^{M} w_j = 1,\ l_j \leq w_j \leq u_j\}$ be the weighting set. We can define the optimization problem associated with an MCDM method ranking score function (R). Then, by considering R as our objective function, we process the minimal and maximal values over the feasible region. It can be executed as follows:

$$\begin{array}{lll} min/max & R(w) & \\ s.t. & \sum_{j=1}^{M} w_j = 1, & \\ & l_j \leq w_j \leq u_j & 1 \leq j \leq M, \\ & l_j, u_j \geq 0 & 1 \leq j \leq M, \\ & l_j, u_j \leq 1 & 1 \leq j \leq M, \end{array}$$

where (l_j, u_j) are the selected lower and upper bounds that limit the criteria vanishing or saturation. By definition $l_j, u_j \in [0,1]$ with $1 \leq j \leq M$, but they are expected to be more fitted depending on the needs of the decision-makers. In this way, we would get the optimal values inherent in each optimization problem R^L and R^U as lower and upper scores, with their optimal points W^L and W^U associated with minimization and maximization respectively. Under this assumption, we can operate in the field of unweighted Multiple-Criteria Decision Making or UW-MCDM.

3.4.1 *Unweighted Mean Models*

The Unweighted Mean Models (UW-MM) are a non-weighted version of the WMM techniques. Given that we have removed the weights, the only step before the score optimization is the normalization stage. The main purpose of the unweighted approach is to tackle the weaknesses of the p-mean values obtained for WMM. Moreover, it is very simple to adult the resultant ranking by setting the values of the weighting scheme, which is basically the major concern of the MCDM theory. In Triantaphyllou (2000), we can find some cases in which it is exposed the main drawbacks for WSM and WPM cases.

Unweighted Mean Models method:

Step 1 Normalize the decision matrix as $[r_{ij}]$, per each $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, M\}$.

Step 2 Given $w \in \Omega_{lu}$, we consider the function $\mathcal{M}_i^p : \Omega_{lu} \to [0,1]$ per each $i \in \{1, \dots, N\}$ as:

$$\mathcal{M}_i^p(w) = \quad \mathcal{M}^p(r_{ij}, w) = \left[\sum_{j=1}^{M} w_j r_{ij}^p\right]^{\frac{1}{p}}. \tag{3.52}$$

Step 3 Compute the mathematical optimization problem by considering the p-mean (Eq. 3.52) as the cost function, per each alternative $1 \leq i \leq N$:

$$\mathcal{M}_i^{p,L} = \min\left\{\mathcal{M}_i^p(r_{ij}, w) : w \in \Omega_{lu}\right\}. \tag{3.53}$$

$$\mathcal{M}_i^{p,U} = \max\left\{\mathcal{M}_i^p(r_{ij}, w) : w \in \Omega_{lu}\right\}. \tag{3.54}$$

An interesting point to highlight is that the formulation of the problem in Eq. 3.53 and Eq. 3.54, matches with a linear programming system as long as $p = 1$. In addition, the Ω_{uw} set is clearly convex. Then, we can obtain the optimal solutions by means of classic solvers. The mathematical formulation would be the following:

$$\begin{array}{lll} \text{Min/Max} & \sum_{j=1}^{M} w_j r_{ij}, & \\ \text{s.t} & \sum_{j=1}^{M} w_j = 1, & \\ & l_j \leq w_j \leq u_j & 1 \leq j \leq M, \\ & l_j, u_j \geq 0 & 1 \leq j \leq M, \\ & l_j, u_j \leq 1 & 1 \leq j \leq M. \end{array}$$

Thus, it is easy to note that the last statement holds true whenever the UW-MM technique is shifted to the unweighted version of the WSM.

Proposition 3.4.1. *Given an Unweighted Mean Model framework in which the following statements hold true:*

(a) The p-mean utilized in Eq. 3.52 is the weighted mean, i. e. $p = 1$.

(b) The bounds selected for Step 2 verify $l_j = u_j = w_j^0$ per each $j \in \{1, \dots, M\}$.

Then, the UW-MM result coincides with the classical WSM of Fishburn (1967).

Proof. Considering the decision matrix as $[r_{ij}]$, with the (a) statement, we can rewrite the score function as $\mathcal{M}_i^1(w) = \mathcal{M}^1(r_{ij}, w) = \sum_{j=1}^M w_j r_{ij}$. In other words, we have the same weighted aggregation as the original. In addition, the (b) statement converts the Ω_{lu} into the standard Ω set as in § 3.2. Therefore, the values of each $M_i^{p,L}$ and $M_i^{p,U}$ coincides with M_i^1 per each $1 \leq i \leq N$, thus returning a classic WSM method. $\square$

The pseudo-code of the Unweighted Mean Model algorithm is presented in Algorithm 3.1, where it is included the incorporation of custom normalization, choice of p-mean, selection of (l_j, u_j)-bounds, and initial guess for the optimal weights.

Algorithm 3.1 Unweighted Mean Model algorithm implemented as in our GitHub repository López-García (2022a).

```
Require: Input decision matrix $X = [x_{ij}]$
Require: Normalization $\Phi$
Require: Generalized mean $p$
Require: Weight bounds $L$ and $U$
Require: Initial guess $W_0$
1: $\Omega = \{w | \sum_j w_j = 1, l_j \leq w_l \leq u_j\}$                Define the weight space
2: $[r_{ij}] = \Phi([x_{ij}])$                                     Normalize the decision matrix
3: for $i = 1, \dots, N$ do
4:     $\mathcal{M}_i^{p,L}, W_i^L$ =minimize($\mathcal{M}_i^p, [r_{ij}], \Omega, p, W_0$)        Minimize $\mathcal{M}_i^p$
5:     $\mathcal{M}_i^{p,U}, W_i^U$ =minimize($-\mathcal{M}_i^p, [r_{ij}], \Omega, p, W_0$)       Maximize $\mathcal{M}_i^p$
6: end for
7: return $\{M_i^{p,L}, M_i^{p.U}\}_{i=1}^N$, $\{W_i^L, W_i^U\}_{i=1}^N$
```

Remark. *For the last Proposition 3.4.1, we can also formalize the WPM method if we modify the (a) statement for the value $p = 0$. The proof would be trivial since $\mathcal{M}_i^0(w) = \mathcal{M}^0(r_{ij}, w) = \prod_{j=1}^M w_j r_{ij}$.*

Remark. *For the last Proposition 3.4.1, we can also generalize such property for any value of p in the (a) statement. The proof would be similar given that the (b) statement converts the Ω set into a single vector, i. e. null for Lebesgue measure, with no spread in $[0, 1]^M$.*

Despite the fact that we have not mentioned the normalization step, it is important to highlight that such transformation is essential in MCDA. In the particular case of WSM and WPM, the approach relies on the additive utility assumption. Nonetheless, both methods should be utilized with a defined normalization for Step 1 to avoid the aggregation of different units of measure Triantaphyllou et al. (1997).

3.4.2 *Unweighted TOPSIS*

The Unweighted TOPSIS (uwTOPSIS) technique Liern and Pérez-Gladish (2020) ranks decision alternatives based on the classical TOPSIS approach before mentioned in § 3.2.2. As a result of working without weights, the method solves the optimization problems that involve the relative proximity function (Eq. 3.27) in Step 5. The output gives us information about both minimal and maximal possible rank values per each alternative. Unlike the weighted approach, now the algorithm can be conducted by following the next six steps, in which the Step 2 is removed.

Unweighted TOPSIS method:

Step 1 Normalize the decision matrix as $[r_{ij}]$, where each $r_{ij} \in [0,1]$ for each $i \in \{1,\dots,N\}$ and $j \in \{1,\dots,M\}$.

Step 2 Positive ideal and negative ideal are determined as $PIS = (v_1^+,\dots,v_M^+)$ and $NIS = (v_1^-,\dots,v_M^-)$ where

$$v_j^+ = \begin{cases} \max\limits_{1\le i\le n} r_{ij} & \text{if} \quad j \in J_{max} \\ \min\limits_{1\le i\le n} r_{ij} & \text{if} \quad j \in J_{min} \end{cases} \qquad 1 \le j \le M,$$

$$v_j^- = \begin{cases} \min\limits_{1\le i\le n} r_{ij} & \text{if} \quad j \in J_{max} \\ \max\limits_{1\le i\le n} r_{ij} & \text{if} \quad j \in J_{min} \end{cases} \qquad 1 \le j \le M,$$

where J_{max} is the set of criteria to be maximized and J_{min} is the set of criteria to be minimized.

Step 3 Given Ω_{lu} and d a distance defined in $[0,1]^M \times [0,1]^M$, we consider the separating functions $D_i^+, D_i^- : \Omega_{lu} \to [0,1]$ for each of $i \in \{1,\dots,N\}$ as

$$D_i^+(w) = d\Big((w_1 r_{i1},\dots,w_M r_{iM}),(w_1 v_1^+,\dots,w_M v_M^+)\Big). \tag{3.55}$$

$$D_i^-(w) = d\Big((w_1 r_{i1},\dots,w_M r_{iM}),(w_1 v_1^-,\dots,w_M v_M^-)\Big). \tag{3.56}$$

Step 4 The relative proximity function to the ideal solutions is defined as $R_i : \Omega_{lu} \to [0,1]$ with:

$$R_i(w) = \frac{D_i^-(w)}{D_i^+(w) + D_i^-(w)}, 1 \le i \le N. \tag{3.57}$$

Step 5 For each $i \in \{1,\dots,N\}$, the values R_i^L and R_i^U are calculated by solving the mathematical programming problems over R_i considering the set of weights as the problem variables

$$R_i^L = \min\{R_i(w) : w \in \Omega_{lu}\}, \tag{3.58}$$

$$R_i^U = \max\left\{R_i(w) : w \in \Omega_{lu}\right\}, \tag{3.59}$$

where l_j lower bound and u_j upper bound of each w_j, $\forall j \in \{1, \ldots, M\}$.

The main consequence of the absence of weights is that the Eq. 3.26 and Eq. 3.27 in the classical TOPSIS are converted into functions that depend on the $w \in \Omega_{lu}$ as in Eq. 3.55, Eq. 3.56, and Eq. 3.57. Hence, the output is not only a decision interval $[R_i^L, R_i^U]$ but a set of optimal weights $\{w_i^{*L}, w_i^{*U}\}$ which give us information about separation boundaries and relative importance of the criteria. It is noteworthy to mention that uwTOPSIS does not rank alternatives per se, in case our goal is to sort a number of alternatives, we would need to define a rank function that involves the values of R_i^L and R_i^U.

Proposition 3.4.2. *Given a uwTOPSIS framework in which the following statements hold true:*

(a) The normalization in Step 1 is the vector ℓ^2 normalization (Table 3.1).

(b) The distance d utilized in Step 3 is the Euclidean distance.

(c) The bounds selected for Step 5 verify $l_j = u_j = w_j^0$ per each $j \in \{1, \ldots, M\}$.

Then, the uwTOPSIS result coincides with the classical TOPSIS of Hwang and Yoon (1981).

Proof. The demonstration of this proof is based on the original form extracted from Liern and Pérez-Gladish (2020). According to (a), the normalized matrix has the shape:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{N} x_{ij}^2}}. \tag{3.60}$$

For (b) we have set the same distance function than Hwang and Yoon (1981), so the relative proximity function R can be written as:

$$R_i(w) = \frac{\left[\sum_{j=1}^{M}(w_j r_{ij} - w_j v_j^-)^2\right]^{\frac{1}{2}}}{\left[\sum_{j=1}^{M}(w_j r_{ij} - w_j v_j^+)^2\right]^{\frac{1}{2}} + \left[\sum_{j=1}^{M}(w_j r_{ij} - w_j v_j^-)^2\right]^{\frac{1}{2}}} \tag{3.61}$$

Given that (c) implies that Ω is a null set, we can represent $\Omega = \{w_1^0, \ldots, w_M^0\}$. As a result, it is trivial that the domain of the function is a single element, transforming it into a vector representation. Then, $R_i(\Omega_{lu}) = R_i^L = R_i^U = R_i$ so uwTOPSIS and TOPSIS coincide. □

The pseudo-code of the Unweighted TOPSIS algorithm is presented in Algorithm 3.2, where it is included the incorporation of optimal directions, custom normalization, distance function, selection of (l_j, u_j)-bounds, the initial guess for the optimal weights, and the option of whether to force ideal solutions.

In the first proposal of Liern and Pérez-Gladish (2020), uwTOPSIS was applied to compare its performance with classical TOPSIS, in which the mathematical programming problem was solved by using the LINGO software. The examples implied solving a real car selection problem based on the dataset of Kao (2010) and rank factoring companies with

Algorithm 3.2 Unweighted TOPSIS algorithm implemented as in our GitHub repository López-García (2021a).

```
Require: Input decision matrix $X = [x_{ij}]$
Require: Optimal directions $v$
Require: Normalization $\Phi$
Require: Distance function $D$
Require: Weight bounds $L$ and $U$
Require: Initial guess $W_0$
Require: Whether to force ideal solutions $fis$
1: $\Omega = \{w \mid \sum_j w_j = 1,\ l_j \leq w_l \leq u_j\}$        Define the weight space
2: $[r_{ij}] = \Phi([x_{ij}])$        Normalize the decision matrix
3: $PIS,\ NIS = \texttt{get\_ideals}(v,\ fis)$        Get ideal solutions
4: for $i = 1, \dots, N$ do
5:     $R_i^L, W_i^L = \texttt{minimize}\{R_i,\ [r_{ij}],\ \Omega,\ p,\ W_0\}$        Minimize $R_i$
6:     $R_i^U, W_i^U = \texttt{minimize}\{-R_i,\ [r_{ij}],\ \Omega,\ p,\ W_0\}$        Minimize $R_i$
7: end for
8: return $\{R_i^L,\ R_i^U\}_{i=1}^N,\ \{W_i^L,\ W_i^U\}_{i=1}^N$
```

the sane decision matrix as Alemi-Ardakani et al. (2016). With the aim of expanding the use of this method, Benítez and Liern (2021) developed an open source R package (Benitez and Liern, 2020) and showed the deployment of their program for sustainability analysis. In particular, they studied the classification of Green Star hotels in Istanbul for sustainable tourism and the Social Progress Index (SPI) indicator to analyze the European Union regions. In the same line, Blasco-Blasco et al. (2021) developed an open source Python library (López-García, 2021a) and gave an application to build composite indicators for academic purposes in the Industrial University of Santander. Other remarkable contributions of the Unweighted TOPSIS method are the building of composite indicators for gender diversity Liern and Pérez-Gladish (2021), sustainable tourism management in Spain after COVID-19 Vicens-Colom et al. (2021), and the ranking of companies based on their diversity and financial performance Bouslah et al. (2022).

3.4.3 *Unweighted VIKOR*

The Unweighted VIKOR (uwVIKOR) ranks decision alternatives based on the classical VIKOR approach before mentioned in § 3.2.3. As a consequence of working without weights, the method solves the optimization problems that involve the aggregation function (Eq. 3.31), but unlike uwTOPSIS, now we have three score functions that yield six different scores. Now, the uwVIKOR can be applied by following the next steps:

Unweighted VIKOR method:

Step 1 Determine the ideal solutions $PIS = (x_1^+, \dots, x_M^+)$ and $NIS = (x_1^-, \dots, x_M^-)$ per each criterion $j \in \{1, \dots, M\}$ as in Step 1 of VIKOR.

Step 2 Normalize the decision matrix as $[r_{ij}]$ as in Eq. 3.28 per each $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, M\}$.

Step 3 Given a $w \in \Omega_{lu}$, we define the functions $S_i, R_i : \Omega_{lu} \to [0,1]$ per each $i \in \{1,\dots,N\}$ as:

$$S_i(w) = \sum_{j=1}^{M} w_j r_{ij}. \quad \text{Utility function.} \tag{3.62}$$

$$R_i(w) = \max_{1 \leq j \leq M} \{w_j r_{ij}\} \quad \text{Regret function.} \tag{3.63}$$

Step 4 Given a utility parameter $\nu \in [0,1]$, we define the aggregation function $Q_i : \Omega_{lu} \to [0,1]$ per each $i \in \{1,\dots,N\}$ as the combination:

$$Q_i(w) = \nu \frac{S_i(w) - S^-}{S^+ - S^-} + (1-\nu) \frac{R_i(w) - R^-}{R^+ - R^-}, \tag{3.64}$$

where:

$$S^- = \min_{1 \leq i \leq N} S_i(w), \quad R^- = \min_{1 \leq i \leq N} R_i(w),$$
$$S^+ = \max_{1 \leq i \leq N} S_i(w), \quad R^+ = \max_{1 \leq i \leq N} R_i(w).$$

Step 5 For each $i \in \{1,\dots,N\}$, the values Q_i^L and Q_i^U are calculated by solving the mathematical programming problems over Q_i considering the set of weights as the problem variables:

$$Q_i^L = \min \{Q_i(w) : w \in \Omega_{lu}\}, \tag{3.65}$$

$$Q_i^U = \max \{Q_i(w) : w \in \Omega_{lu}\}, \tag{3.66}$$

where l_j is the lower bound and u_j is the upper bound of each w_j, $\forall j \in \{1,\dots,M\}$.

Once the procedure is conducted, we get the pairs (Q_i^L, Q_i^U) for every alternative with their optimal points (W_i^L, W_i^U) in Ω_{lu}. As a consequence, we can also get the utility and stability measures associated with such points by evaluating them with Eq. 3.62 and Eq. 3.63. Thus we get (R_i^L, R_i^U) and (S_i^L, S_i^U). The reader must take into account that such scores do not necessarily have to be their R or S optimal points, because in the Step 5 just the Q_i-score is optimized.

One of the major contributions of the VIKOR is the introduction of compromise solutions by means of the Q-score that defines the final ranking. In the unweighted approach, we cannot sort the alternatives since there is no direct binary relation that defines a partial order for pairs. Therefore, we present here some possible solutions to determine whether the resultant output of uwVIKOR has a compromise solution.

Option 1 Given $0 < \lambda_S, \lambda_R, \lambda_Q < 1$ and $p \in \mathbb{R}$, we define:

$$S_i = \left(\lambda_S (S_i^L)^p + (1-\lambda_S)(S_i^U)^p\right)^{\frac{1}{p}},$$

$$R_i = \left(\lambda_R (R_i^L)^p + (1-\lambda_R)(R_i^U)^p\right)^{\frac{1}{p}},$$

$$Q_i = \left(\lambda_Q (Q_i^L)^p + (1-\lambda_Q)(Q_i^U)^p\right)^{\frac{1}{p}}. \tag{3.67}$$

So we basically operate with the generalization of the $\mathcal{M}^p$ per each score to aggregate the results obtained.

Option 2 Given $0 < \nu_S, \nu_R, \nu_Q < 1$, we calculate the values of:

$$\begin{aligned}
S_-^L &= \min_{1\le i\le N}\{S_i^L\}, \quad S_-^U = \min_{1\le i\le N}\{S_i^U\}, \quad S_+^L = \max_{1\le i\le N}\{S_i^L\}, \quad S_+^U = \max_{1\le i\le N}\{S_i^U\},\\
R_-^L &= \min_{1\le i\le N}\{R_i^L\}, \quad R_-^U = \min_{1\le i\le N}\{R_i^U\}, \quad R_+^L = \max_{1\le i\le N}\{R_i^L\}, \quad R_+^U = \max_{1\le i\le N}\{R_i^U\},\\
Q_-^L &= \min_{1\le i\le N}\{Q_i^L\}, \quad Q_-^U = \min_{1\le i\le N}\{Q_i^U\}, \quad Q_+^L = \max_{1\le i\le N}\{Q_i^L\}, \quad Q_+^U = \max_{1\le i\le N}\{Q_i^U\},
\end{aligned}$$

to propose:

$$\begin{aligned}
S_i &= \nu_S \frac{S_i^L - S_-^L}{S_+^L - S_-^L} + (1-\nu_S)\frac{S_i^U - S_-^U}{S_+^U - S_-^U}.\\
R_i &= \nu_R \frac{R_i^L - R_-^L}{R_+^L - R_-^L} + (1-\nu_R)\frac{R_i^U - R_-^U}{R_+^U - R_-^U},\\
Q_i &= \nu_Q \frac{Q_i^L - Q_-^L}{Q_+^L - Q_-^L} + (1-\nu_Q)\frac{Q_i^U - Q_-^U}{Q_+^U - Q_-^U}.
\end{aligned} \tag{3.68}$$

In both cases, we would proceed with the final scores in the same way that Step 5 in VIKOR works with (R_i, S_i, Q_i) to define a new version of the compromise solution for the non-weighted approach.

The pseudo-code of the Unweighted VIKOR algorithm is presented in Algorithm 3.3, where it is included the incorporation of optimal directions, utility parameter, selection of (l_j, u_j)-bounds, and initial guess for the optimal weights.

Algorithm 3.3 Unweighted VIKOR algorithm implemented as in our GitHub repository López-García (2021b).

```
Require: Input decision matrix $X = [x_{ij}]$
Require: Optimal directions $v$
Require: Utility parameter $\nu$
Require: Weight bounds $L$ and $U$
Require: Initial guess $W_0$
 1: $\Omega = \{w | \sum_j w_j = 1, l_j \le w_l \le u_j\}$                Define the weight space
 2: for $j = 1, \dots, M$ do
 3:     if $v_j == "max"$ then
 4:         $r_{ij} = (\max_{1\le i\le N}\{x_{ij}\} - x_{ij})/(\max_{1\le i\le N}\{x_{ij}\} - \min_{1\le i\le N}\{x_{ij}\})$    Normalize Max-direction
 5:     else
 6:         $r_{ij} = (x_{ij} - \min_{1\le i\le N}\{x_{ij}\})/(\max_{1\le i\le N}\{x_{ij}\} - \min_{1\le i\le N}\{x_{ij}\})$    Normalize Min-direction
 7:     end if
 8: end for
 9: for $i = 1, \dots, N$ do
10:     $Q_i^L, S_i^L, R_i^L, W_i^L$ =minimize$(Q_i, [r_{ij}], \Omega, v, \nu, W_0)$    Minimize $Q_i$
11:     $Q_i^U, S_i^U, R_i^U, W_i^U$ =minimize$(-Q_i, [r_{ij}], \Omega, v, \nu, W_0)$    Maximize $Q_i$
12: end for
13: return $\{S_i^L, S_i^U, R_i^L, R_i^U, Q_i^L, Q_i^U\}_{i=1}^N$, $\{W_i^L, W_i^U\}_{i=1}^N$
```

Proposition 3.4.3. *Given an uwVIKOR framework in which the bounds selected for Step 3 verify $l_j = u_j = w_j^0$ per each $j \in \{1, \ldots, M\}$. Then, the uwVIKOR result coincides with the classical VIKOR of Opricovic (1998).*

Proof. The boundary limitation of $l_j = u_j$ per each $j \in \{1, \ldots, M\}$ means that the Ω_{lu} set has null measure in $\mathbb{R}^M$. As a result, the w values per each function S, R, Q have no spread. Therefore, it returns a single value per each weighting scheme, giving us the same $Q_i(\Omega) = Q_i^L = Q_i^R = Q_i$ as in classic VIKOR. $\square$

Unlike in Proposition 3.4.1 and 3.4.2, the VIKOR method has clearly stated the kind of normalization (Eq. 3.28) and score functions (utility Eq. 3.29 and regret Eq. 3.30). Then, it makes it easier to transform from the unweighted scheme to the classic one.

3.4.4 *Problems attached to the Unweighted MCDM*

As long as we assign numerical values to a weighting scheme, the decision-makers involved will add their personal bias to the selection of alternatives. Even though the methodology is mathematically well-design, we also have to pay attention when selecting the relative importance of each criterion within the MCDM procedure. As a manner to solve such concerns, we have developed a set of methods that guarantees a free-bias system for member selection. The methodology is called Unweighted Multiple-Criteria Decision Making or UW-MCDM in short. By addressing a set of boundaries to each component of the weighting scheme we can not only remove the bias attached to each weight but also check how susceptible each alternative is in regard to the criteria selected. Hence, decision-makers can operate considering intervals instead of single values, in which we have more information about the behavior among alternatives for each MCDM framework.

The first problem that we encounter when we perform an unweighted model is the definition of an order relationship ($\preceq$) so that we can generate a cardinal ranking from the R^L and R^U scores. Even though we have recommended some ways to generate a final score, it may not behave as we expect and it could not work for every single UW-MCDM environment.

The second problem is the assignment of bounds (l_j, u_j) in the Ω_{lu} set for $j \in \{1, \ldots, M\}$. Although it makes a more flexible approach, we still have to decide their values and justify whether different criteria have associated different range limits. Then, it is not an easy task because we can note that an unweighted methodology with no bounds, i.e. $l_j = 0$ and $u_l = 0$ per each $j \in \{1, \ldots, M\}$, would underestimate the non-significant attributes per each alternative, thus exploding the most predominant criterion. In such a case, the result would not be accepted since it would not consider all the set of criteria.

3.5 FUZZY UNWEIGHTED APPROACH

With the aim of combining the best of fuzzy and unweighted approaches, we present an outline of how they should be defined with their stages and functionality. The objective is to include uncertainty together with non-biased schemes to generate an adaptive system that prevents most part of the common drawbacks of the MCDM techniques. Fig. 3.8 illustrates the process to carry out a decision-making problem by means of the fuzzy unweighted strategy.

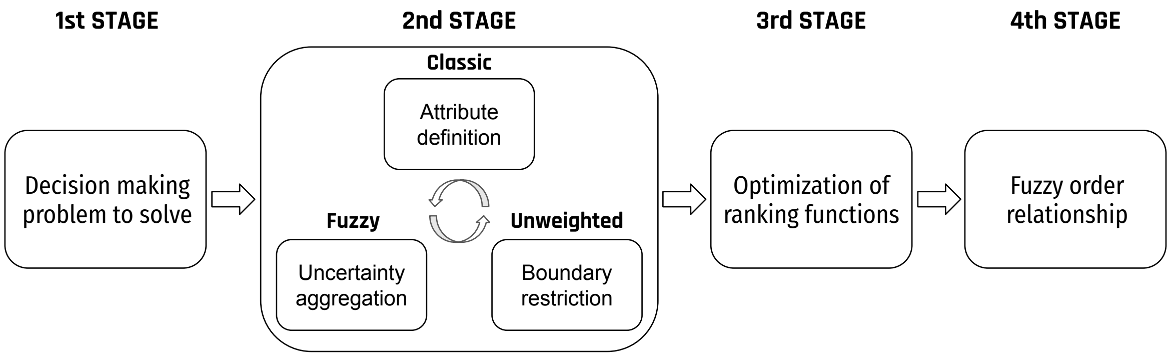


Figure 3.8: Scheme applied when solving an MCDM problem via Fuzzy Unweighted approach. Source: Own elaboration.

In the main, the technique is the same as § 3.3, although unweighted schemes add high complexity in terms of mathematical approach and computational calculus. Despite the fact that fuzzy operations are clearly stated at the beginning of § 3.3, it is not an easy task to perform the optimization of such ranking functions. The main problem is that the set of fuzzy numbers is not a totally ordered group, hence it requires an order relationship $\preceq$ so that we can compare the fuzzy results per each alternative.

For this thesis, the field of FUW-MCDM is introduced as a new methodology for helping decision-makers by means of the combination of fuzzy sets with non-static weighting schemes. Our proposal is to define the fuzzy unweighted version of WMM presented in § 3.2.1, as the Fuzzy Unweighted Mean Models (FUW-MM) method.

Fuzzy Unweighted Mean Models method:

Step 1 Determine the fuzzy decision matrix $[\tilde{x}_{ij}]$ so that $\tilde{x}_{ij} = (x^L_{ij}, x^R_{ij}, \alpha^L_{ij}, \alpha^R_{ij})_{L_{ij}R_{ij}}$ and define the fuzzy weights $\tilde{w}_j = (w^L_j, w^R_j, \beta^L_j, \beta^R_j)_{L'_jR'_j}$, per each $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$.

Step 2 Determine a fuzzy order relationship $\preceq$ to make pairwise comparisons between alternatives.

Step 3 Define the bounded weight space as $\tilde{\Omega}_{lu} = \{\tilde{w} | \tilde{l}_j \preceq \tilde{w}_j \preceq \tilde{u}_j\}$, where $\tilde{l}_j = (l^L_j, l^R_j, \alpha^L_j, \alpha^R_j)_{L_jR_j}$ and $\tilde{u}_j = (u^L_j, u^R_j, \alpha^L_j, \alpha^R_j)_{L_jR_j}$.

Step 4 Normalize the fuzzy decision matrix as $[\tilde{r}_{ij}]$, per each $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$.

Step 5 Given $\tilde{w} \in \tilde{\Omega}_{lu}$ and $p \in \mathbb{R}$, we consider the fuzzy function $\tilde{\mathcal{M}}^p_i : \tilde{\Omega}_{lu} \to [0, 1]$ per each $i \in \{1, \ldots, N\}$ as in Eq. 3.52.

Step 6 Compute the mathematical optimization problem by considering $\tilde{\mathcal{M}}^p_i$ as cost function, per each alternative $1 \leq i \leq N$:

$$\tilde{\mathcal{M}}_i^{p,L} = \min\left\{\tilde{\mathcal{M}}_i^p(\tilde{r}_{ij},\tilde{w}) : \tilde{w} \in \tilde{\Omega}_{lu}\right\}. \tag{3.69}$$

$$\tilde{\mathcal{M}}_i^{p,U} = \max\left\{\tilde{\mathcal{M}}_i^p(\tilde{r}_{ij},\tilde{w}) : \tilde{w} \in \tilde{\Omega}_{lu}\right\}. \tag{3.70}$$

Step 7 Perform a defuzzification method (Leekwijck and Kerre, 1999) that turns $\tilde{\mathcal{M}}_i^{p,L}$ and $\tilde{\mathcal{M}}_i^{p,U}$ into numerical scores $\mathcal{M}_i^{p,L}$ and $\mathcal{M}_i^{p,U}$.

The pseudo-code of the FUW-MM algorithm is presented in Algorithm 3.4.

Algorithm 3.4 Fuzzy Unweighted Mean Model algorithm implemented as in our GitHub repository López-García (2022a).

```
Require: Input fuzzy decision matrix $\tilde{X} = [\tilde{x}_{ij}]$
Require: Set the $(L, R)$ reference functions as trapezoidal
Require: Normalization $\Phi$
Require: Generalized mean $p$
Require: Fuzzy weight bounds $\tilde{l}$ and $\tilde{u}$
Require: Fuzzy order relationship $\preceq$
Require: Initial guess $W_0$
 1: $\tilde{x}_{ij} = (x_{ij}^L, x_{ij}^R, \alpha_{ij}^L, \alpha_{ij}^R)_{L_{ij}R_{ij}}$        Define the fuzzy structure
 2: $\tilde{w}_j = (w_j^L, w_j^R, \alpha_j^L, \alpha_j^R)_{L_jR_j}$        Define the fuzzy weights
 3: $\tilde{l}_j = (l_j^L, l_j^R, \alpha_j^L, \alpha_j^R)_{L_jR_j}$        Define the fuzzy lower bounds
 4: $\tilde{u}_j = (u_j^L, u_j^R, \alpha_j^L, \alpha_j^R)_{L_jR_j}$        Define the fuzzy upper bounds
 5: $\tilde{\Omega} = \{\tilde{w} \mid \tilde{l}_j \preceq \tilde{w}_l \preceq \tilde{u}_j\}$        Define the fuzzy weight space
 6: $[\tilde{r}_{ij}] = \Phi([\tilde{x}_{ij}])$        Normalize the fuzzy decision matrix
 7: for $i = 1, \dots, N$ do
 8:     $\tilde{\mathcal{M}}_i^{p,L}, \tilde{W}_i^L$ = minimize $(\tilde{\mathcal{M}}_i^p, [\tilde{r}_{ij}], \tilde{\Omega}, p, \tilde{W}_0, \preceq)$        Minimize $\tilde{\mathcal{M}}_i^p$
 9:     $\tilde{\mathcal{M}}_i^{p,U}, \tilde{W}_i^U$ = minimize $(-\tilde{\mathcal{M}}_i^p, [\tilde{r}_{ij}], \tilde{\Omega}, p, \tilde{W}_0, \preceq)$        Maximize $\tilde{\mathcal{M}}_i^p$
10: end for
11: return $\{\tilde{\mathcal{M}}_i^{p,L}, \tilde{\mathcal{M}}_i^{p.U}\}_{i=1}^N$, $\{\tilde{W}_i^L, \tilde{W}_i^U\}_{i=1}^N$
```

Unless otherwise stated, the order relationship defined in Step 2 of the FUW-MM algorithm is given by the distance of an LR-fuzzy number to the fuzzy zero $(0,0,0,0)$. Then, a fuzzy alternative is greater than another if and only if its distance to is higher. In this manner, we can perform the optimization problem stated in Step 6 by calculating the longer/shorter distance with respect to the LR-fuzzy zero.

In contrast with the procedure detailed in § 3.3.1, now it is addressed mathematical programming together with an order relationship $(\preceq)$. The incorporation of both concepts transforms a fuzzy outcome (subjected to a defuzzification stage) into a fuzzy scheme in which the components of the fuzzy numbers are composed of interval solutions. Hence, this range of fuzzy values will be responsible for generating the ranking system. When comparing Algorithms 3.4 and 3.1, the incorporation of LR-fuzzy numbers considerably complicates the optimization problem of the ranking functions. Not only for the mathematical approach but also for the computational cost involved in its implementation.

In the same line as § 3.4.1, we have analyzed some properties of the FUW-MM algorithms. The motivation of the following proposition is to show the link among the four different kinds of MCDA algorithms presented in Chapter 3.

Proposition 3.5.1. *Given a Fuzzy Unweighted Mean Model framework to solve a decision-making problem, then the following properties hold true:*

(a) *If $\tilde{l}_j = \tilde{u}_j = \tilde{w}_j^0$ per each $j \in \{1, ..., M\}$, then the FUW-MM result coincides with the FWMM problem described in § 3.3.1 for the weighting scheme $\{\tilde{w}_j^0\}_{j=1}^M$.*

(b) *If every fuzzy set involved in the problem $\tilde{x}_{ij}$, $\tilde{w}_i$, $\tilde{l}_j$ and $\tilde{u}_j$ have crisp-number shape, then the FUW-MM resultant ranking coincides with the ranking of a UW-MM problem described in § 3.4.1 for the defuzzification of all the fuzzy sets.*

(c) *If every fuzzy set involved in the problem $\tilde{x}_{ij}$, $\tilde{w}_i$, $\tilde{l}_j$ and $\tilde{u}_j$ have crisp-number shape and $\tilde{l}_j = \tilde{u}_j = \tilde{w}_j^0$ per each $j \in \{1, ..., M\}$, then the FUW-MM resultant ranking coincides with the ranking of a WMM problem described in § 3.2.1 for the defuzzification of all the fuzzy sets with weighting scheme $\{w_j^0\}_{j=1}^M$.*

Proof. For the case (a), we proceed equivalently than in the proof (and two remarks) of the Proposition 3.4.1. It is easy to note that $\tilde{\Omega}_{lu}$ now is composed by a single vector of fuzzy number $\tilde{w}_j^0$, then the fuzzy score $\tilde{\mathcal{M}}_i^p$ as a unique image over $\tilde{\Omega}_{lu}$. Hence, the resultant score per each alternative matches with the FWMM score described in § 3.3.1.

For the case (b), it is essential to take into account that the reference functions of crisp-numbers (L, R) are zero-valued out of the fuzzy core and such core consists of a single value, so $x_{ij}^L = x_{ij}^R = x_{ij}^0$ per each i, j pairs. This is also applicable for $\tilde{w}_i$, $\tilde{l}_j$ and $\tilde{u}_j$. As a result, any operation out of the fuzzy cores is null, thus limiting the fuzzy properties presented throughout § 3.3. Then, all the operations described for the core of the FUW-MM problem are the same as the methodology presented in UW-MM. Therefore, the score result of the FUW-MM after the defuzzification step matches with the UW-MM for the conditions described.

For the case (c), we just have to combine the notions detailed for the proofs of (a) and (b). Along the same line, the score obtained in an FUW-MM framework will be the same as a WMM problem.

□

3.6 SUMMARY

In this chapter, we have focused on how Multiple-Criteria Decision Analysis and Decision-Making operate and how helpful and profitable this subject is for decision-makers when facing a real problem with conflicting criteria. We have indicated different aspects and approaches regarding the kind of application desired or the environment selected. Within the family of outranking algorithms in MCDM, we have presented three families of algorithms with three different methodologies: classic, fuzzy, and unweighted. Apart from describing how they were designed and modeled, we have defined various versions of such algorithms, especially for the unweighted approach. Our main objective has been to generate a complete scheme for applicability purposes when outranking methods are required in a decision-making problem. Finally, a new area of study is presented concerning the Multiple-Criteria Decision Analysis field under the name of Fuzzy-Unweighted Multiple-Criteria Decision Making. Some interesting properties are presented to understand the FUW-MCDM area as a generalization of the three methodologies previously presented.

4 ARTIFICIAL INTELLIGENCE

From 1760 to 1840, the world experienced a massive change due to the Industrial Revolution. Great Britain was the first place where the economic, social, and technological transition occurred. Still, it did not take so long to expand to the European continent and the United States (Wrigley, 2018). The transformation implemented led to the standardization of new manufacturing processes thanks to the rise of mechanized factory systems. The use of machine tools and the power sources of coal, steam, and water provided humans with advanced methods that increased productivity, then achieving economies of scale. As a result, industrialization is considered one of the significant turning points in human history. Nowadays, the way we perceive societies is highly marked by the impact of the Industrial Revolution (Clark, 2014).

Over time, there have been four substantive stages related to transitions in the industry. As technologies advanced and their use was relevant at work, their impact has been marked throughout history. The following items summarize an overview of the different phases (de Vries, 1994).

I Industrial Revolution: The first human transition from hand production to machinery and supply chain use. Between the 18th and 19th centuries, the textile manufacturing and iron industry experienced rapid growth at the time. It made a difference in the societal status with the rise of the middle class.

II Industrial Revolution: It is also known as the Technological Revolution. In the late 19th century, there was a rapid spread of information due to the incorporation of the telegraph, which connected people around the world. In a physical context, the development of railroad networks enhanced the transportation of raw materials and finished products. Then, it is understood as the first step towards globalization. For societies, it meant a significant improvement in the standard of living due to the incorporation of gas and water supply along sewage systems in most regions of each country. The end of this period was due to the outbreak of the First World War in Europe.

III Industrial Revolution: It is also known as the Digital Revolution. It started at the half of the 20th century, with a transition marked by the emergent development of the Second World War. The key point for this stage was the shift from mechanical and analog systems to digital electronics and electric technologies. Since their incorporation into the industry, the world changed to the so-called Information Age, and it was due to the proliferation of digital computers and digital communications. The use of integrated circuits and transistors meant a great advance compared to the second industrial revolution. Due to the underlying technological change, mass production was a new standard for capitalist societies.

IV Industrial Revolution: It is an ongoing revolution called Industry 4.0 or 4IR. At the end of the 20th century, there was rapid change due to the smart automation of processes. Most of these enhancements involve using artificial intelligence techniques, including robotics and multi-agent systems. Moreover, the cybernetic tools allowed humans to perform decentralized and interconnected economies. The rise of the Internet of Things (IoT) through digitization provided an enormous advantage regarding the last industrial revolutions.

A summarized visualization of the last mentioned is illustrated in Fig. 4.1.

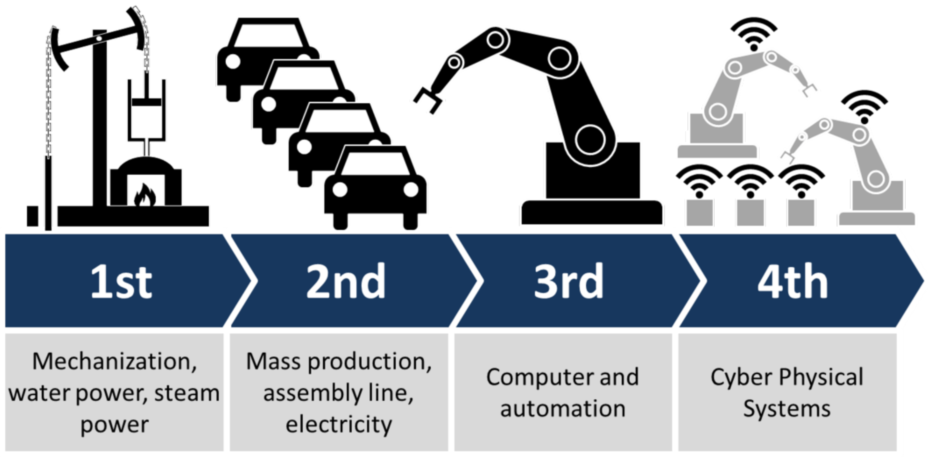

1st
2nd
3rd
4th
Mechanization, water power, steam power
Mass production, assembly line, electricity
Computer and automation
Cyber Physical Systems

Figure 4.1: Transitions made in the four stages of the industrial revolutions. Source: LinkedIn

The systems of digitalization have represented a breakthrough in the way we perceive the business world (Gamil et al., 2020). The continuous scanning and tracking of information is performed during the entire year and exchanged via Internet communication networks worldwide. The hardware responsible for collecting, processing, and storing some input information is diverse and specific depending on the task. Thus, all the physical devices and the software that enhances both digitalization and transmission of data are known as the Internet of Things (Miraz et al., 2015). Some usual examples are wireless sensors, control systems, cameras, and microphones. The rise of embedding systems has been the main responsible for developing intelligent systems and the industrial automation of processes. The main idea is to monitor the activity of most parts of a business sequence to control and evaluate its performance and yield.

The applicability of IoT is not only focused on machinery, it is also implemented for the customer, government, infrastructure, digital marketing, operational, and security applications (Ramson et al., 2020). As long as some routine tasks can be digitalized, the use of these sets of software can give us valuable information about the current situation.

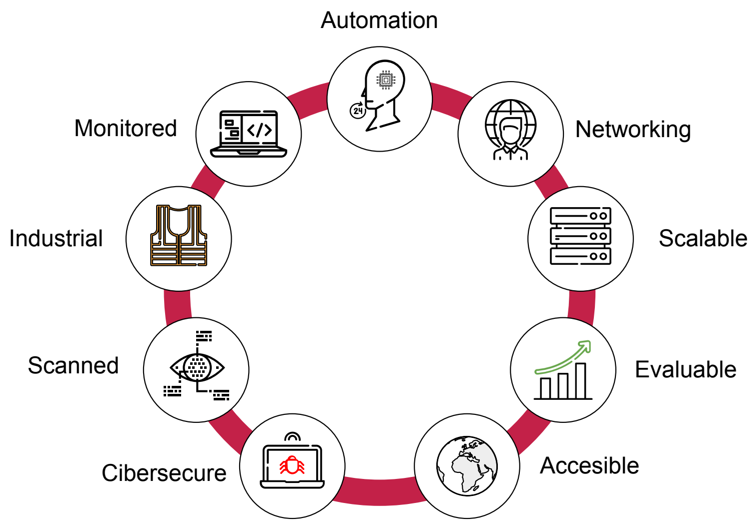


Figure 4.2: Activities, standards, and processes of the Internet of Things ecosystem.
Source: Own elaboration

So far, industries made a big effort to cover the automation processes. Since the first Industrial Revolution, the goal was to reduce human intervention, substituting it with machine activity. Over the next stages, the industries attempted to increase their productivity in order to achieve economies of scale and guarantee efficiency in their sector. However, it took a long time until people realized that he could mimic human behavior and respond intelligently to a given environment and conditions. By following that line, Artificial Intelligence (AI) emerged as a field of computer science that replicates the natural intelligence that characterizes humans to include it in algorithms or models displayed in real-life cases (Doumpos and Grigoroudis, 2013; Hastie et al., 2009). It utilizes the so-called intelligent agents to capture the event studied, and the system takes actions or decisions by means of the information given by such agents.

Although considering an intelligent machine may seem a very innovative notion, in 1950, Alan Turing already discussed whether or not it is possible for machinery to show intelligent behavior and whether machines could think by themselves (Cooper and van Leeuwen, 2013). So far, we have created software that simulates smart activity. However, some blatant limitations point out their inferiority against humans (LeCun et al., 2015). First, the machines can learn as much as the owner has decided. Second, the independence of machines is only considered for the task that they carry out. Last, regardless of the performance shown by intelligent systems, humans are responsible for taking the next step and deciding whether the system is ready to be launched.

In the previous paragraphs, we discussed that AI tries to replicate a human perception of reality to give an appropriate solution to some problem. Although it is correct, we all agree that the range of daily problems in our lives cannot be considered equally complex. Then, the action to execute will determine the level of involvement by machine interactions. Depending on the needs of the situation approached, we can divide the technique for learning the task into two different classes (Hastie et al., 2009).

- Intelligent Systems: Under some specific circumstances well defined and measurable, the model returns an exact decision to follow. Hence, human knowledge is conveyed to the machine.
- Statistical Learning: For the event that we are analyzing, the past information gives us the knowledge to perform an accurate response. Then, the model learns how to solve an uncertain scenario by making use of mathematical techniques.

Actually, AI can be understood as a division of three subfields that depends on how the algorithm has learned to respond to some particular action and how it processes the information given. In addition, the structure of the algorithm describes the architecture of the intelligent model, as well as the scope of the response that will be returned. Thus, the difficulty associated with the studied event and the data utilized in the learning phase requires a concrete architecture, which may be an Intelligent System, Machine Learning, or Deep Learning, as shown in Fig. 4.3.

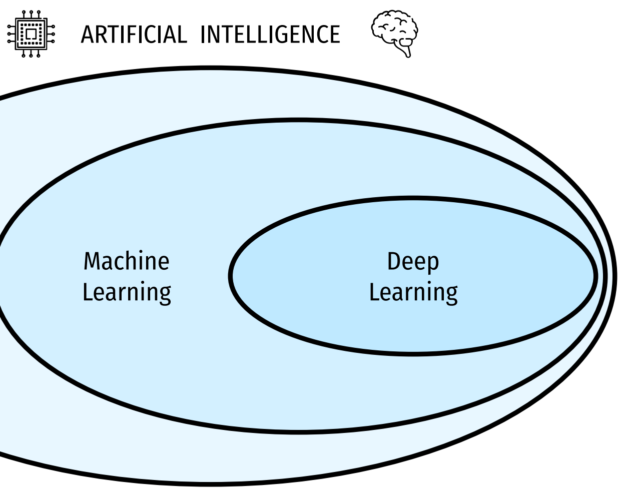


Figure 4.3: Main subfields of the Artificial Intelligence.
Source: Own elaboration

Thanks to the contributions made by mathematical analysis, operations research, statistics, and computation, nowadays, interdisciplinary teams develop robust and powerful artificial intelligence systems with incredible applications for real-life solutions. However, it took multiple scientists a long time and effort until they got the notion of what we today know as artificial intelligence (Doumpos and Grigoroudis, 2013).

The beginning of AI was world-renowned in the 1950s due to the significant contributions that arose at the time. Two of the main recognized theorists were Alan Turing, known as the father of theoretical computer science (Cooper and van Leeuwen, 2013), and Claude Shannon, known as the father of information theory (James, 2009).

4.1 MACHINE LEARNING

In the second half of the XX century, the field of Machine Learning (ML), also known as Statistical Learning, was a mathematical pure field with a theoretical perspective of functional analysis, operations research, probability, and statistics. This science aims to approximate, estimate, or recognize some response or patterns from a given dataset. For this purpose, researchers paid special attention to the construction of intelligent machines that automatically generate human-like interactions. Even though it seemed such a hard task in those days, the development of modern computers made it feasible.

The ML field includes advanced statistics, pure mathematics, modern computation techniques, and programming languages. Over the years, the literature related to this discipline has exploded due to the multiple applications that emerged for Industry 4.0 and Big Data. Nonetheless, their limitations are also a problem to face when implementing statistical techniques in the business. It is broadly known that three main points limit machine learning algorithms.

- Data: The amount of data utilized when dealing with real-life problems allows us to generalize and find common patterns in the studied events. We can also learn from large datasets and discriminate non-significant characteristics from a dataset. In such a way, the law of large numbers and the central limit theorem allow us to generalize our data. Moreover, the quality of the data is an important element to consider. We not only need to gather as much data as possible but also make sure that our dataset satisfies some basic properties that characterize the subjects of study.

- Complexity: An in-depth analysis can lead us to reach our goals. However, in the field of statistical learning, it is not always a necessary condition. The human bias can be a counterpart for computers since clear facts for humans are not as evident to detect with algorithms. Then, the computational part of the problem must be as simple as possible. It is the same point as Ockham's razor fallacy states, we should model a straightforward solution.

- Computation: Even though modern computer science allows us to easily perform our algorithms, computation is still a concern when dealing with large datasets or heavy input data. Moreover, almost every artificial intelligence problem is NP-hard, so the computational cost implies an exponential time.

The implementation of machine learning models requires considering some individual terms. Given a dataset $\mathcal{D} = \{X_i, y_i\}_{i=1}^N$ of N random independent identically distributed (i.i.d.) samples, the goal of ML models is to solve the mathematical programming problem of minimizing the empirical risk associated with some loss function $\mathcal{L}$ over the dataset (Vapnik, 1992). That is to say, we can formulate it by considering a functional decision set $\mathcal{F} = \{f : X \to Y\}$ so that $\mathcal{D} = \{X_i, y_i\}_{i=1}^N \subset X \times Y$, we can define a loss function as the cost function defined over $Y \times Y$ such that it returns positive values that indicates the amount of lost information by our model. Then, $\mathcal{L} : f(X) \times Y \to \mathbb{R}^+$ contrasts the response and the actual result. The key role of this function is to operate over the image of some function $f \in \mathcal{F}$ and to determine how suited the function is for the model over the data samples.

Finally, our objective is to obtain the best possible model, so we define the risk associated with the decision function in $\mathcal{F}$ as:

$$\text{Risk:}\quad R(f) = \mathbb{E}_{X,y}\left[\mathcal{L}(f(X),Y)\right] = \int_{X\times Y} \mathcal{L}(f(X),y)d\mathcal{P}(X,y), \quad \forall f \in \mathcal{F}. \tag{4.1}$$

When dealing with real problems, the dataset is extracted from a sample, which is usually discrete. Hence, the equation 4.1 can be converted to:

$$\text{Empirical risk:}\quad R_{\mathcal{D}}(f) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(f(X_i),y_i), \quad \forall f \in \mathcal{F}. \tag{4.2}$$

Once we have determined the dataset, the set of models, the loss function, and the empirical risk associated, now we can define the optimization problem behind machine learning:

$$\begin{aligned} \text{Minimize} \quad & R_{\mathcal{D}}(f), \\ \text{st.} \quad & f \in \mathcal{F}. \end{aligned}$$

Then, the best model is the one that satisfies to be $\underset{f\in\mathcal{F}}{\operatorname{argmin}} R_{\mathcal{D}}(f)$.

Given that the role of the loss function ($\mathcal{L}$) is to determine how good is the output of the model, it is crucial to select a function that appropriately interprets our data and punishes the errors propagated. Depending on the objective of the problem and the input data selected, we can decide on multiple losses. Considering $\{\hat{y}_i\}_{i=1}^{N}$ as the set of response elements, some of the common cost functions $\mathcal{L}(\hat{y}_i, y_i)$ are:

$$0-1\text{ loss:}\quad \mathbf{1}_{\{\hat{y}_i \neq y_i\}} = \begin{cases} 1 & \text{if } \hat{y}_i \neq y_i, \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

$$\text{Quadratric loss:}\quad \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2. \tag{4.4}$$

$$\text{Cross Entropy loss:}\quad -\sum_{i=1}^{N} y_i \log \hat{y}_i. \tag{4.5}$$

A great part of the classification approaches makes use of the Kullback-Leibler divergence, also known as relative entropy, as a loss function (Joyce, 2011). It is commonly extended since our predictions will represent the membership probability to some set. It can be formulated for two probability distributions P and Q as follows:

$$D_{KL}(P|Q) = \sum_{x\in X} P(x)\log\left(\frac{P(x)}{Q(x)}\right), \quad \text{discrete version.} \tag{4.6}$$

$$D_{KL}(P|Q) = \int_X p(x)\log\left(\frac{p(x)}{q(x)}\right)dx, \quad \text{continuous version.} \tag{4.7}$$

It is easy to note that D_{KL} is an asymmetric distance that compares the difference in distribution between the two probability samples. In other words, it is the expectation of the logarithmic coefficient between the two random samples. The relative entropy does not satisfy the triangle inequality either. It happens because it is a divergence in terms of information geometry. Its representation within machine learning is to measure the information gain of a distribution P in regard to an actual distribution Q.

As far as the learning process is concerned, the objective of the machine is to generalize from a given dataset and export the acquired knowledge to further data. How the data is structured will define the approach when learning. Machine learning algorithms may be divided into four different approaches (Suthaharan, 2016).

Supervised learning: The dataset contains both the input and the output of the event. The objective is to simulate the behavior of the event studied so that it fits the actual response in the future. Common types of supervised techniques include classification and regression.

Unsupervised learning: The dataset just has input samples with no additional information. This approach aims to discover the underlying structure of data. It is usually based on pattern recognition by means of statistical estimators. Two well-known applications are clustering and auto-encoders to classify or reconstruct input variables.

Semi-supervised learning: This approach is a mixture of supervised and unsupervised techniques. This problem arises when our dataset just has outputs for some fraction of the samples. It may occur when dealing with uncertainty or when combining different sources of information. Then, the implementation requires some assumptions that relate to the given samples. Some applications are speech analysis or the automatic text classifier.

Reinforcement learning: This approach attempts to mimic the behavior of how humans or intelligent systems take their actions in a given environment. Then, most of the algorithms implemented make use of dynamic programming techniques to solve such problems. Most of these models are implemented to simulate human behavior against some complex activity such as autonomous driving or competing in board games.

The models from each family have different structures, but they share the use of weights or parameters that through approximation techniques achieve the better fitting possible (Vapnik, 1999). Previously, we have described a functional decision set $\mathcal{F}$ which basically represents the family of models that can tackle some tasks. Then, per each $f \in \mathcal{F}$, we can attach to the model their combination of weights to process the desired set of data, i. e. $f(x, W)$ for $W \in \Omega_{\mathcal{F}}$. The main objective of machine learning models is to fit the input data given by assigning an adequate value of W.

When we refer to fitting a model, the underlying idea is the optimization stage utilized for minimizing the empirical risk associated (Vapnik, 1992). To execute this process, the input data is usually divided into batches that contain a certain number of elements, so the number of iterations is a multiple of the ceiling function of such a division. Finally, all these batches are iteratively introduced to the model f as many times as required, which is named epochs. Then, the entire dataset is passed forward through the model as many times as epochs are determined. The idea of utilizing the same set of samples multiple times may seem a little confusing and inefficient, however, we have to keep in mind that the data is serialized and computed in batches to transform the fitting phase into a feasible problem in terms of computational cost.

In the framework needed to implement the methodology of machine learning, we basically have defined a functional decision set and a loss function. The functional model f contains the weights to be optimized and the loss function ascertains the objective function for the optimization process (Hastie et al., 2009; Vapnik, 1992). In order to perform such a task, we have to make use of the so-called optimizers, which are a set of algorithms that optimizes the weights of the function when learning from data. Assuming differentiability for the loss function in the neighborhood of an initial point $W_0 \in \Omega_{\mathcal{F}}$, we can define the following optimizers for a dataset $\mathcal{D} = \{X_i, y_i\}_{i=1}^{N} = (X, y)$ and $F(W|X, y) = \mathcal{L}(f(X, W), y)$. In order to ease the mathematical notation applied, we assume that $\nabla_W F(W|X, y) = \nabla F(W|X, y)$ because it is well known that we are optimizing with regard to the W parameters. Detailed information for the implementation and formulation of the following optimization algorithms can be found in Ruder (2016).

1. Gradient Descent (GD): It is a first-order optimization algorithm. The strategy is to move along the steepest direction of the function, i. e. the inverse direction of the gradient. It can be computed in an iterative way as:

$$W_{k+1} = \quad W_k + \eta \nabla F(W_k | X, y), \quad \text{with } k \geq 0. \tag{4.8}$$

   The η parameter is a positive value known as the learning rate. Sometimes it can also be sequenced as:

$$\eta_{k+1} = \quad \frac{\left|[W_{k+1} - W_k]^\top [\nabla F(W_{k+1}) - \nabla F(W_k)]\right|}{||\nabla F(W_{k+1}) - \nabla F(W_k)||^2}, \quad \text{with } k \geq 0. \tag{4.9}$$

2. Stochastic Gradient Descent (SGD): Unlike the GD algorithm, now an updating is performed per each sample. The idea is to reduce the high redundancy through one-by-one iterations. It is also much faster because the update of the weights is evaluated with a single pair (X_i, y_i), which is computationally advantageous for limited machines.

$$W_{k+1}^i = \quad W_k^i + \eta \nabla F(W_k^i | X_i, y_i), \quad \text{with } k \geq 0. \tag{4.10}$$

3. Mini-batch Gradient Descent: Given a fixed length $0 < b < N$, we can compute the SGD by batches of samples instead of individual sampling. With the batch strategy, we are reducing the individual variance during the learning phase which may lead us to better local minima.

$$W_{k+1}^i = \quad W_k^i + \eta \nabla F(W_k^i | X_{(i:i+b)}, y_{(i:i+b)}), \quad \text{with } k \geq 0. \tag{4.11}$$

Where the $(i : i+b)$ subindex indicates the batch sequence from the i element to the $i+b$. An implementation of the algorithm is described in 5.

4. Momentum Stochastic Gradient Descent: In order to control the variation rate of the updated weights in the SGD algorithm, we can incorporate the concept of momentum by computing ΔW_k per iteration. Such a concept allows the algorithm to control steep changes among the elements of W. We can set a decay rate $0 < \alpha < 1$ to determine the amount of information passed to ΔW_k. It accelerates the convergence to an optimal solution since we are considering the direction, and so, reducing the oscillations across iterations.

$$\begin{aligned} v_k^i &= \quad \eta \nabla F(W_k^i | X_i, y_i) + \alpha \Delta W_k \quad , \\ W_{k+1}^i &= \qquad W_k^i + v_k^i, \qquad \text{with } k \geq 0. \end{aligned} \tag{4.12}$$

We can note that now we have an additional parameter to set, which adds more complexity when establishing the initial conditions of the optimizer. However, it is commonly set as $\alpha = 0.9$.

5. Nesterov Accelerated Gradient (NAG): The incorporation of direction vector quantity with moment makes the algorithm more consistent. Nonetheless, we are still omitting the practical variation of ∇F regarding its last position of the weights W_{k+1}. Then, we can guide the cost function in each update by modifying the evaluation of our loss.

$$\begin{aligned} v_k^i &= \quad \eta \nabla F(W_k^i - \alpha \Delta W_k | X_i, y_i) + \alpha \Delta W_k \quad , \\ W_{k+1}^i &= \qquad W_k^i + v_k^i, \qquad \text{with } k \geq 0. \end{aligned} \tag{4.13}$$

The change in the image per $W_k^i - \alpha \Delta W_k$ controls the accumulated gradient vector, so the NAG optimizer prevents fast results to non-satisfactory local optima.

6. Adaptive Gradient (AdaGrad): It is an adaptive-gradient method that tunes the learning rate to the parameters, which is highly recommendable when dealing with sparse data (Zeiler, 2012). Then, the updates rely on the gains obtained when evaluating the next step. We first define the gradient of the objective function at step k and component $j \in \{1, \dots, J\}$:

$$g_{k,j} = \quad \left[\nabla F(W_k | X, y)_j\right]_{j=1}^{J} = \partial_j F(W_k | X, y). \tag{4.14}$$

Then, we compute a similar weight update than SGD but now the learning rate is now modified by step and iteration using the sum of squares of each gradient component $\partial_j F(W_k^i | X_i, y_i)$:

$$W_{k+1,j} = \quad W_{k,j} - \frac{\eta}{\sqrt{G_{k,jj} + \epsilon}} g_{k,j}. \tag{4.15}$$

Where G_k is the diagonal matrix with the j, j components are the sum of squares of the j-component of $\nabla F(W_k|X, y)$ up to the step k. The ϵ constant is the smoothing term, which avoids the division by zero, and it is a very small value like 10^{-8}. One of the main benefits of AdaGrad is that the learning rate is iteratively tuned, so the choice of the initial value is not as rigorous as in SGD. In practice, it is usually set as 0.01 to control the gradient decay.

7. AdaDelta: It is an extension of AdaGrad with a different adaptive system for the learning rate parameter (Zeiler, 2012). The principal drawback of the AdaGrad algorithm is the aggressive monotonic reduction of the learning rate. As a manner to solve this problem, AdaDelta incorporates an accumulated window of w past iterations and Hessian function approximations for second-order methods. With the same representation of the gradients g_k, the square sum is averaged as an expectation.

$$\mathbb{E}[g^2]_k = \quad \rho\mathbb{E}[g^2]_{k-1} + (1-\rho)g_k^2,$$

where ρ is the controlling decay parameter usually fixed equal to η. We must bear in mind that now, the average just sums the past w elements.

Then, the weight update could be conducted with learning rate adaption over $\mathbb{E}[g^2]_{k+1}$ as:

$$\begin{aligned} W_{k+1,j} = &\quad W_{k,j} - \frac{\eta}{\sqrt{\mathbb{E}[g^2]_k + \epsilon}} g_k \\ = &\quad W_{k,j} - \frac{\eta}{RMS[g]_k} g_k. \end{aligned} \tag{4.16}$$

When studying the updates, we can note that the units do not match. Then, we can define an exponentially decaying average instead.

$$\mathbb{E}[\Delta W^2]_k = \quad \rho\mathbb{E}[\Delta W^2]_{k-1} + (1-\rho)\Delta W^2, \tag{4.17}$$

In such a way, the iteration is computed by:

$$RMS[\Delta W]_k = \quad \sqrt{\mathbb{E}[\Delta W^2]_k + \epsilon}. \tag{4.18}$$

Now, we can approximate the value of $RMS[\Delta W]_k$ with their update until the previous step. Finally, we replace the learning rate η in 4.16 by $RMS[\Delta W]_{k-1}$ to give the formulation:

$$W_{k+1,j} = \quad W_{k,j} - \frac{RMS[\Delta W]_{k-1}}{RMS[g]_k} g_k. \tag{4.19}$$

A great development for optimizing is that the learning rate has been suppressed in the AdaDelta algorithm. Another interesting advance in both computational and analytical areas is the limitation in the gradient accumulation by w length.

8. RMS propagation (RMSprop): It is a particular case of the AdaDelta algorithm, but is very extended in the Artificial Intelligence sector. In this algorithm, it is suggested the use of $\rho = 0.9$ for the squared sum of gradients, so that it leads to the following representation:

$$\begin{aligned} \mathbb{E}[g^2]_k &= 0.9\mathbb{E}[g^2]_{k-1} + 0.1g_k^2, \\ W_{k+1,j} &= W_{k,j} - \frac{\eta}{\sqrt{\mathbb{E}[g^2]_k + \epsilon}} g_k. \end{aligned} \tag{4.20}$$

9. Adaptive Moment Estimation (ADAM): It is an algorithm that computes adaptive learning rate as well (Kingma and Lei Ba, 2014). Its advanced implementation stores an exponentially decaying average of both past gradients m_k and past squared gradients v_k.

$$\begin{aligned} m_k &= \beta_1 m_{k-1} + (1-\beta_1)g_k, \\ v_k &= \beta_2 v_{k-1} + (1-\beta_2)g_k^2. \end{aligned} \tag{4.21}$$

Where they were usually initialized as 0-vectors. Thus, Adam estimates the first and second moment of gradients starting by g_k and g_k^2 respectively. It has been proven that this strategy is biased by 0, then we can counteract such concern by computing the following correctors:

$$\begin{aligned} \hat{m}_k &= \frac{m_k}{1-\beta_1}, \\ \hat{v}_k &= \frac{v_k}{1-\beta_2}. \end{aligned} \tag{4.22}$$

To conclude, Adam takes a similar implementation to AdaDelta in Eq. 4.16 but with the following difference:

$$W_{k+1,j} = W_{k,j} - \frac{\eta}{\sqrt{\hat{v}_k} + \epsilon} \hat{m}_k. \tag{4.23}$$

The authors recommended the use of $\beta_1 = 0.9$, $\beta_2 = 0.999$. For the numerical stability is used $\epsilon = 10^{-8}$. An implementation of the algorithm is described in 5.

10. AdaMax: It is an infinite norm generalization of the Adam algorithm. The update strategy for Adam is to scale the gradient of the loss by means of an ℓ^2 vector norm, as we can see in Eq. 4.21. Then, we could generalize such an idea to ℓ^p spaces as:

$$\begin{aligned} v_k &= \beta_2^p v_{k-1} + (1-\beta_2^p)|g_k|^p \\ &= (1-\beta_2^p)\sum_{l=1}^{k} \beta_2^{p(k-l)} |g_l|^p. \end{aligned} \tag{4.24}$$

For the AdaMax algorithm, we consider the $p = +\infty$ case for Eq. 4.24, so we can get the mathematical formulation by taking the limit:

$$\begin{aligned} u_k = & \lim_{p \to +\infty} [v_k]^{\frac{1}{p}} \\ = & \lim_{p \to +\infty} \left[(1 - \beta_2^p) \sum_{l=1}^{k} \beta_2^{p(k-l)} |g_l|^p \right]^{\frac{1}{p}} \\ = & \lim_{p \to +\infty} [1 - \beta_2^p]^{\frac{1}{p}} \left[\sum_{l=1}^{k} \beta_2^{p(k-l)} |g_l|^p \right]^{\frac{1}{p}} \\ = & \lim_{p \to +\infty} \left[\sum_{l=1}^{k} \left[\beta_2^{(k-l)} |g_l| \right]^p \right]^{\frac{1}{p}} \\ = & \max_{1 \le l \le k} \left\{ \beta_2^{(k-l)} |g_l| \right\}. \end{aligned} \tag{4.25}$$

The procedure to get to the last equity is by means of the same reasoning as Proposition 3.2.1. So we can finally write the formula:

$$u_k = \max \{ \beta_2 u_{k-1}, |g_k| \}. \tag{4.26}$$

This algorithm allows us to know the bounds of the weight updates. The main profit is that we do not need to correct the initialization bias.

We have seen how stochastic optimization is performed. In particular, we have shown the main implementation of the steepest descendent, which is the Stochastic Gradient, and how it has been developed until current optimizers for neural networks. An illustration of the training cost implicated in a classification task is shown in Fig. 4.4.

The procedure where our model learns from a given dataset $\mathcal{D}$, is broadly known as the fitting phase (Suthaharan, 2016). In general, both the procedure and the dataset are split into three phases, which are called training, validation, and testing. During the training, the model sequentially makes experiments with the train data to give the first response and store their batch empirical loss. Afterward, the weights are fixed, and then the validation data are handled likewise. The point is to compare the losses obtained per each batch and per each group (train vs validation) and then check how our model performs by comparing it along the number of epochs within the fitting. Once training and validation are conducted, the test phase evaluates how well is our model generalizing, since the test data is not known by our model when fitting their weights. Then, the test stage is the final proof that analyzes the quality of the predicted output until the model is eventually launched. Even though these steps might seem confusing, their proper implementation guarantees a good praxis in the data analytics field. Among many advantages, we would like to emphasize the following fit cases (Bashir et al., 2020):

1. Under-fitting: The resultant model had not learned significant information from the dataset. Therefore their applicability to reality remains far.

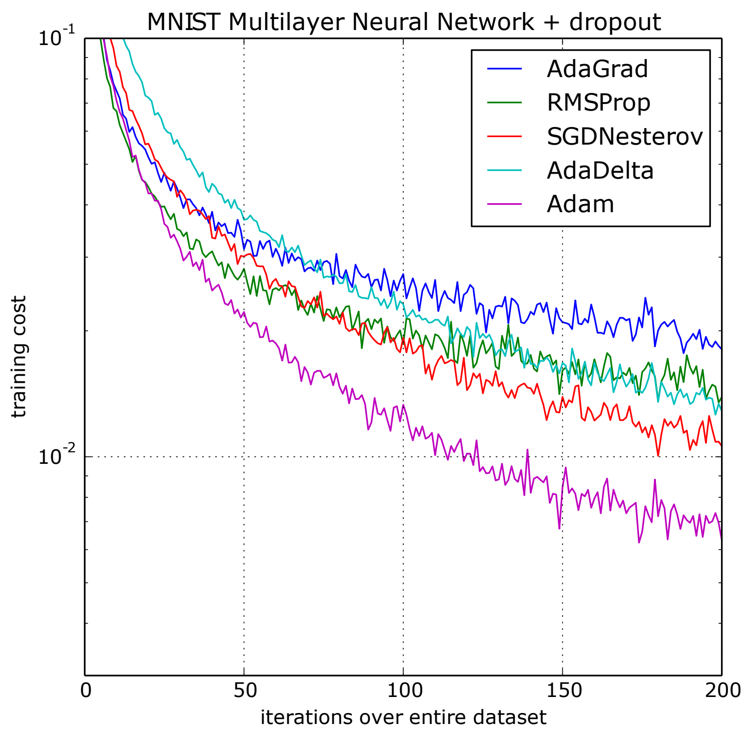


Figure 4.4: Performance of five different gradient-based optimization algorithms in the classification task of MNIST dataset.
Source: Kingma and Lei Ba (2014).

2. Good-fitting: The model has learned to generalize the studied event. It has shown an acceptable performance during the training and validation phases and, in turn, in the test phase.

3. Over-fitting: The resultant model just "memorized" the input data, that is to say, its applicability is limited to our dataset. As a consequence, our model will not be able to work out of the experimental set.

In Fig. 4.5, it is illustrated the three mentioned cases for a particular regression problem defined in $\mathbb{R}^2$.

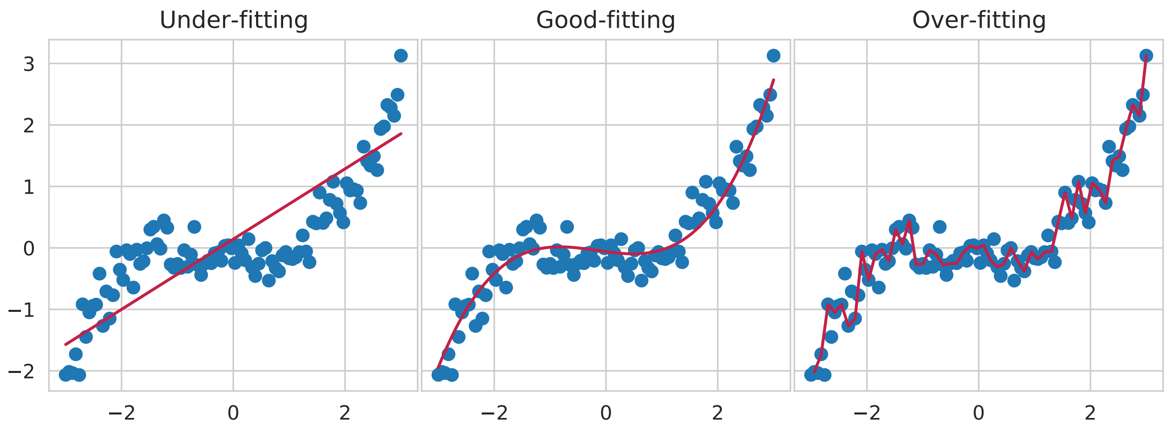


Figure 4.5: Possible fitting cases when facing a regression problem.
Source: Own elaboration.

Apart from the fitting structure presented in the predictions, we can also represent the loss curves presented during the training phase. In other words, we can successively store the values of $\mathcal{L}$ in both the training and validation sets and check how these successions converge. It is easy to realize that a balanced learning phase with acceptable fitting has to have similar loss values in the involved sets. A visible representation of the three mentioned cases of the fitting is depicted in Fig. 4.6, for a learning instance of 100 epochs.

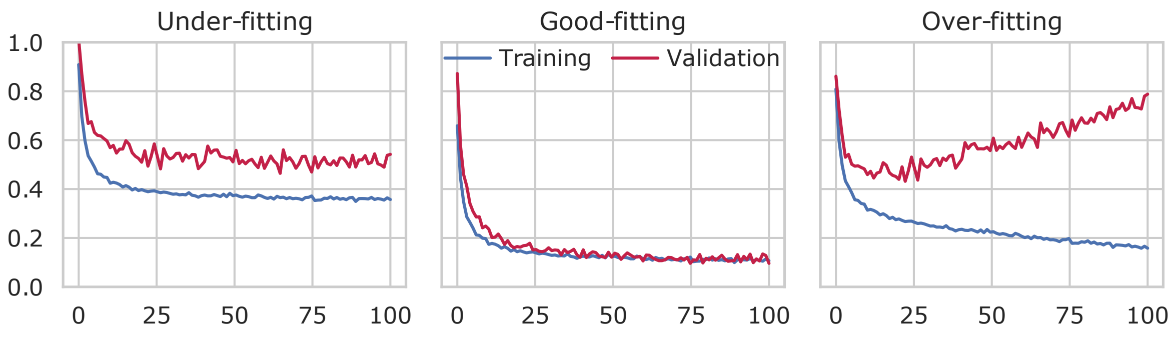


Figure 4.6: Loss curves for the three different cases in a learning procedure.
Source: Own elaboration.

We would like to mention that there exist some other cases related to a learning procedure. We already mentioned at the beginning of this section that the limitations of machine learning are subjected to data, complexity, and computation boundaries. Then, we can obtain multiple different results when facing hard tasks or when handling data under uncertainty. Nonetheless, we are assuming basic conditions that would lead to standard behavior in terms of learning.

The performance of the fitted model has to be evaluated in the testing phase, and so with the proper test set that was put aside. So far, we have described the fitting procedure and the protocol implemented over the train and validation sets, however, we have not indicated the utility of the test set. We first need to emphasize that the loss operator computed over the batches and epochs shows the global situation of the model's execution. Thus, the empirical risk is iteratively minimized to fit the dataset behavior. When we obtain the argmin $R_{\mathcal{D}}(f)$ by modifying the weights, we could evaluate the model with the same used sets, nonetheless, it would imply an inappropriate. In the first case, this is because the fitting has been conducted in order to reproduce the best possible response. Second, it may occur that our model only "memorizes" the data instead of learning the event's behavior. Hence, an external set (test set) is utilized to overcome that hypothetical case. It is important to mention that we could also process the data to study the attributed loss, but it is essentially studied some performance measures for the predictions returned by the model.

When studying a binary case, our model is expected to respond $f(X_i) = \hat{y}_i \in \{0, 1\}$ indicating whether the case study occurs. As a result, we can face four different cases depending on the two possible values of y_i and $\hat{y}_i$. They are defined as follows:

Hit	True positive	TP	$y_i = 1$ and $\hat{y}_i = 1$
Correct rejection	True negative	TN	$y_i = 0$ and $\hat{y}_i = 0$
Type I error	False positive	FP	$y_i = 0$ and $\hat{y}_i = 1$
Type II error	False negative	FN	$y_i = 1$ and $\hat{y}_i = 0$

In a matrix form, we can categorize the result given by our model as an actual-predicted representation. It is important not only for the formalization of the AI model evaluation but for the easy visualization that is generated. Table 4.1 contains a decision matrix example for a binary classification problem.

Table 4.1: Confusion matrix for a binary classification problem.

		Predicted label	
		Positive	Negative
Actual label	Positive	TP	FN
	Negative	FP	TN

It is easy to note that $TP + TN + FP + FP = N$ since the predictions can be obtained by using the entire dataset (although it is broadly extended only by the use of the testing set). Then, we can obtain the associated ratios to check the performance of our AI model in the testing phase (or any other), because the absolute may mislead us when comparing them. The largely used ratios to evaluate diagnostic systems (Bradley, 1997; Fawcett, 2006; Power et al., 2018) are:

$$\text{True Positive Rate or Recall} \quad \frac{TP}{TP+FN}, \tag{4.27}$$

$$\text{True Negative Rate or Specificity} \quad \frac{TN}{TN+FP}, \tag{4.28}$$

$$\text{False Positive Rate or Fall-out} \quad \frac{FP}{TP+FN}, \tag{4.29}$$

$$\text{False Negative Rate or Miss rate} \quad \frac{FN}{TN+FP}, \tag{4.30}$$

$$\text{Positive Predictive Value or Precision} \quad \frac{TP}{TP+FP}, \tag{4.31}$$

$$\text{Negative Predictive Value} \quad \frac{TN}{TN+FN}, \tag{4.32}$$

$$\text{False Discovery Rate} \qquad \frac{TP}{TP+FP}, \tag{4.33}$$

$$\text{False Omission Rate} \qquad \frac{FN}{TN+FN}, \tag{4.34}$$

$$\text{Accuracy} \qquad \frac{TP+TN}{TP+TN+FP+FP}, \tag{4.35}$$

$$\text{Balanced Accuracy} \qquad \frac{TPR+TNR}{2}. \tag{4.36}$$

In general, we reference them by using the acronym made by their initials. By means of the combination of some of the evaluation ratios previously described, we can construct more specific and robust synthetic evaluation metrics.

$$\text{Positive Likelihood Ratio} \qquad \frac{TPR}{FPR}, \tag{4.37}$$

$$\text{Negative Likelihood Ratio} \qquad \frac{FNR}{TNR}, \tag{4.38}$$

$$F_\beta\text{-Score} \qquad (1+\beta^2)\frac{P \cdot R}{\beta^2 P + R}, \tag{4.39}$$

$$\text{Brier score} \qquad \sum_{i=1}^{N} (y_i - \hat{y}_i)^2, \tag{4.40}$$

$$\text{Jaccard score} \qquad \frac{TP}{TP+FP+FN}, \tag{4.41}$$

$$\text{Informedness or J-statistic} \qquad TPR + TNR - 1, \tag{4.42}$$

$$\text{Markedness or } \Delta_p \qquad PPV + NPV - 1, \tag{4.43}$$

$$\text{Fowlkes–Mallows Index} \qquad \sqrt{PPV \cdot TPR}, \tag{4.44}$$

$$\kappa\text{-Coefficient} \qquad \frac{2(TP \cdot TN - FP \cdot FN)}{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}, \tag{4.45}$$

$$\phi\text{-Coefficient or MCC} \qquad \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}. \tag{4.46}$$

Before we use the metrics above defined, it is essential to give a better explanation of their need and correct implementation. The likelihood ratios estimate the probability of cross-hit-error cases for dichotomous outcomes. For that reason, it is widely used for validating medicine diagnostics (Harrell et al., 1982) together with precision and recall. The F_β-score (Sasaki, 2007), with $\beta \geq 0$, considers the relevance of recall when evaluating β-times than precision. The traditional form is with $\beta = 1$, so it is the harmonic mean of precision and recall. The Brier score (Brier, 1950) is the only mentioned formulation that evaluates the classifier by taking into account the predicted class probabilities via mean squared error. The Jaccard score (Jaccard, 1912) determines the similarity between the asymmetric binary attributes of the sample. The informedness (Youden, 1950) indicates the information of a predictor for some condition and the markedness (Power et al., 2018) shows how marked a predictor is for some condition, both based on all the predictions. The Fowlkes–Mallows Index (Fowlkes and Mallows, 1983) is widely used for evaluating clustering techniques because it determines the similarity between groups. The higher this index, the higher similarity is presented. Finally, the ϕ-coefficient (Matthews, 1975), known as the Matthews correlation coefficient, is a robust measure to evaluate a classification task. Owing to their formulation, it considers every single case assumed in the confusion matrix. Then its use is highly recommended, particularly for binary classification evaluation (Chicco and Jurman, 2020; Chicco et al., 2021), being more informative than the before mentioned. It is noteworthy to mention that Informedness, Markerdness, Cohen's κ-coefficient (Cohen, 1960), and MCC ϕ-coefficient values are bounded by -1 and 1, therefore a 0 value is equivalent to an average random prediction.

Despite the fact that the rate metrics (Eq. 4.27 to Eq. 4.36) combined with the confusion matrix give us meaningful information about the model performance, we also have to mind the misclassification errors that may occur (Bradley, 1997). This is the main reason that explains the need for additional metrics (Eq. 4.37 to Eq. 4.46) with complementary interpretation. In order to cover every point that relates to the diagnostic evaluation, we can make use of the Receiver Operating Characteristic (ROC) curve. The ROC curve illustrates the true positive rate (sensitivity) against the false positive rate (specificity) as the decision threshold is varied (Marzban, 2004). The area under the ROC curve, known as AUC, is a simple measure to summarize such trade-off as a single value that allows us to visualize the accuracy of the diagnostic test (Swets, 1988). By accumulating every slope of the ROC points when the decision threshold is varied, we can define a trapezoidal integration curve of the FP-rate (a_i) and TP-rate $(1 - b_i)$ series per each value of $i \in \{2, \ldots, N\}$, as a threshold invariant score:

$$\text{AUC:} \sum_{1=2}^{N} \left[(1 - b_i)(a_i - a_{i-1}) + \frac{1}{2}(b_{i-1} - b_i)(a_i - a_{i-1}) \right] . \tag{4.47}$$

By Eq. 4.47, we can see that the TPR and FPR distributions are simultaneously considered for calculating the AUC score. Then, the graphical representation allows us to interpret the performance of our classifier when we modify the decision threshold. In order to give an in-depth analysis of the ROC curve interpretation, Fig. 4.7 illustrates the variations on the curve depending on two main aspects: the decision threshold and the discriminability index returned from the model performance.

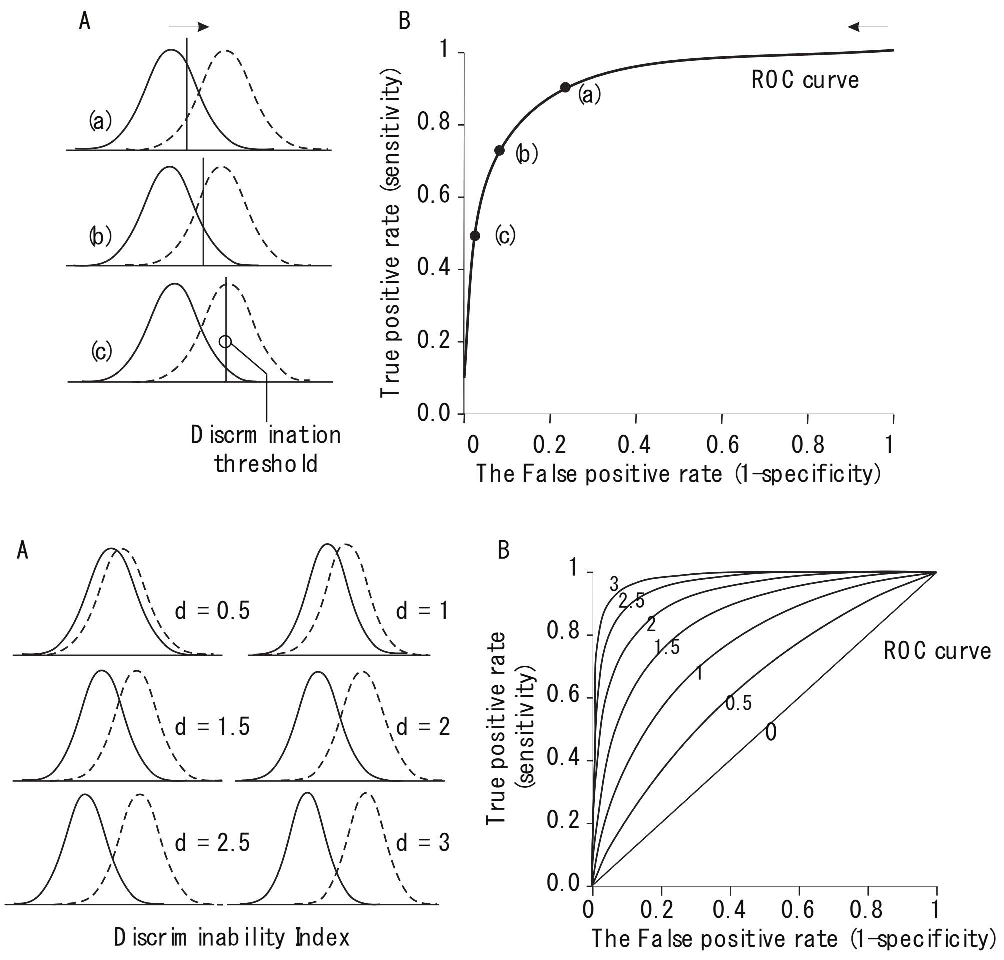


Figure 4.7: Behaviour of the ROC curve when the discrimination threshold (A-top) and the discriminability index (A-bottom) vary over the observations.
Source: Christensen (2009).

The last paragraphs have been dedicated to evaluating the performance of classification models. However, applying such metrics would not make sense in regression tasks. Owing to the nature of the regression problems, the motivation is to approximate the behavior of a known event. Then, the actual value (y_i) and the prediction ($\hat{y}_i$) have to be compared by measuring the closeness of the results because a direct comparison between numbers would be misleading. The usual manner to estimate the model's accuracy is through pairwise comparisons. Some of the evaluation functions used are mentioned above.

$$\text{Mean Absolute Error} \quad \text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{4.48}$$

$$\text{Mean Square Error} \quad \text{MSE} = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{4.49}$$

$$\text{Root Mean Square Error} \quad \text{RMSE} = \sqrt{\text{MSE}} \tag{4.50}$$

Mean Square Log-Error
$$\text{MSLE} = \frac{1}{N}\sum_{i=1}^{N}\left[\log(1+y_i) - \log(1+\hat{y}_i)\right]^2 \tag{4.51}$$

Mean Absolute Percentage Error
$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{|y_i|} \tag{4.52}$$

Weighted-MAPE
$$\text{WMAPE} = \frac{\sum_{i=1}^{N}|y_i - \hat{y}_i|}{\sum_{i=1}^{N}|y_i|} \tag{4.53}$$

Median Absolute Error
$$\text{MedAE} = \text{median}\{|y_i - \hat{y}_i|\}_{i=1}^{N} \tag{4.54}$$

Max Error
$$\text{MaxE} = \max\{|y_i - \hat{y}_i|\}_{i=1}^{N} \tag{4.55}$$

Relative Absolute Error
$$\text{RAE} = \frac{\sum_{i=1}^{N}|y_i - \hat{y}_i|}{\sum_{i=1}^{N}|y_i - \mu_y|} \tag{4.56}$$

Total Error
$$\text{TE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i) \tag{4.57}$$

Coefficient of determination
$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \mu_y)^2} \tag{4.58}$$

Adjusted R^2
$$R^2_{\text{adj}} = 1 - \frac{(1-R^2)(N-1)}{N-k} \tag{4.59}$$

Explained Variance
$$\text{EV} = 1 - \frac{\text{Var}[y - \hat{y}]}{\text{Var}[y]} \tag{4.60}$$

λ – Quantile Loss
$$\lambda\text{Q} = \frac{1}{N}\left[\mathbf{1}_{\{y-\hat{y}\}} - \mathbf{1}_{\{\hat{y}-y\}}\right] \tag{4.61}$$

Tweedie (1984) Deviance
$$\frac{2}{N}\sum_{i=1}^{N}\left(\frac{\max\{y_i, 0\}^{2-p}}{(1-p)(2-p)} - \frac{y_i\hat{y}_i^{1-p}}{1-p} + \frac{\hat{y}_i^{2-p}}{2-p}\right) \tag{4.62}$$

Huber (1964) loss
$$L_\delta = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{if } y_i - \hat{y}_i \leq \delta, \\ \delta\left(|y_i - \hat{y}_i| - \frac{1}{2}\delta\right) & \text{otherwise.} \end{cases} \tag{4.63}$$

4.1.1 *Probabilistic and statistical notions*

In mathematics, we say a system is deterministic as long as such a system is not affected or involved by random variables. Hence, the behavior and the conditions of the system studied will not be altered by randomness in any of their states. A common study approached by deterministic models is the physical solution of differential equations because the initial conditions and development are assumed to be given. On the contrary, there exists systems and processes defined by means of random variables. They are called stochastic or random systems, and they are widely used in mathematical modeling to describe random phenomena.

Since most of the Machine Learning models depend on random variables that affect the state of the process studied, we focus the thesis on developing probabilistic models that help us build more complex intelligent systems.

One of the most basic models for regression analysis is linear regression (LR). It basically predicts some targets through a weighted sum of the input data. The interpretation is quite easy since it learns the linear relationship between variables. When it is tackled as one input for one output, it is named simple linear regression. On the contrary, a multivariate linear regression maps multiple input data to one output. In any case, the input variables are called explanatory variables. LR has various applications, but it is broadly used in two cases. First, when making predictions or forecasting, LR learns from a dataset, and then the linear error is evaluated in the resultant fitting. Second, we can analyze and quantify the linear relationship among different terms when explaining the variation between actions and responses. For the standard implementation of linear regression from a dataset $\mathcal{D} = \{X_i, y_i\}_{i=1}^{N}$, we have to assume weak exogeneity, normality, linear correlation (X over y), homoscedasticity, and error independence. The mathematical formulation is presented as:

$$\begin{aligned} y &= X\beta + \beta_0 + \varepsilon. \\ y_i &= X_i^\top \beta + \beta_0 + \varepsilon_i \\ &= \beta_0 + \beta_1 x_{i1} + \cdots + \beta_n x_{in} + \varepsilon_i \quad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{4.64}$$

The homoscedasticity implies that each error $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$. As we can see, the β-vector determines the set of coefficients to fit during the training, where the β_0 term is called intercept, and it marks the vertical step of the line. The most common way to fit the weighted sum is via the ordinary least squares method, i. e. a loss function equal to Eq. 4.4. Then:

$$\begin{aligned} \beta^* = & \operatorname*{argmin}_{\beta} (y - X\beta)^\top (y - X\beta) = \\ & \operatorname*{argmin}_{\beta_0, \ldots, \beta_n} \sum_{i=1}^{N} \left[y_i - \beta_0 - \sum_{k=1}^{n} \beta_k X_k \right]^2 . \end{aligned} \tag{4.65}$$

By means of the coefficients of β, we can draw conclusions about the weighted value of each instance in the output. Thus, we can get information about which of the features have major relevance when predicting because larger values affect the outcome more. It is very easy to notice due to the linear representation of LR. That is why linear models are very used in most applied sciences. Besides, each β-weight has an associated confidence interval, which is so advantageous when understanding the impact of the feature variation in the response. Another way to get the feature importance of our data is to perform the t-statistic as:

$$t_{\beta_k^*} = \frac{\beta_k^*}{SE(\beta_k^*)}, \tag{4.66}$$

being $SE(\cdot)$ the standard error operator. We can interpret it as a penalized version of the coefficient, so the more variance has, the less impact the feature has.

Sometimes, the dataset we are trying to analyze has leverage to some features or some set of outliers regarding their distribution. In such a case, we conveniently add the so-called regularization to prevent non-significant results caused by the attached bias. As we have mentioned, we can measure the impact of an explanatory variable through their β_k-coefficient associated, so we can force the model to limit the global values of the β vector. Both sparse data or outliers are usually regularized by using the loss functions:

$$\begin{aligned} \text{Lasso:} \quad & (y - X\beta)^\top (y - X\beta) + \lambda_1 ||\beta||_1, && (4.67) \\ \text{Ridge:} \quad & (y - X\beta)^\top (y - X\beta) + \lambda_2 ||\beta||_2, && (4.68) \\ \text{Elastic Net:} \quad & (y - X\beta)^\top (y - X\beta) + \lambda_1 ||\beta||_1 + \lambda_2 ||\beta||_2. && (4.69) \end{aligned}$$

Andrey Tikhonov was the first person to implement regression techniques. Nonetheless, he applied them in the field of integral equations as an ℓ^2 corrector. The incorporation of regularization to solve linear regression was proposed for Hoerl et al. (1962), where authors applied the Ridge analysis developed by Tikhonov. Tibshirani (1996) incorporated the Lasso (least absolute shrinkage and selection operator) estimator. Its application with sparse data was already implemented by Chen and Donoho (1994) for the basis pursuit technique for optimization problems. As a combination of both to overcome some weaknesses presented in Lasso and Ridge, Zou and Hastie (2005) presented Elastic Net as a manner to correct the double shrinkage problem. They generally control the bias-variance tradeoff usually presented in real datasets by means of ℓ^p norms (Li and Liny, 2010).

The performance of linear regression models in real-world applications is not very effective. The explanation is chiefly related to the assumptions stated previously. Firstly, LR formulation only describes linear relationships (Eq. 4.64) based on a weighted sum. In other words, every non-linearity presented among the explanatory variables will not be taken into consideration. Second, their predictive power is less than modern statistical techniques in regression or forecasting tasks. It is merely related to the simplification accepted via single linear functions that describe the dataset as unique events. Nonetheless, it is important to recognize that its easy interpretation of weights is very intuitive when analyzing the feature's effect in our model.

In addition to the least square method that is used as a loss function $\mathcal{L}$, we can also calculate the coefficient of determination, also known as R^2, to measure the proportion of variation explained by the set of dependent variables. The mathematical formulation is shown in Eq. 4.70.

$$R^2 = \quad 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{SS_{residual}}{SS_{total}}. \tag{4.70}$$

Thus, we can interpret the R^2 coefficient as a measure of goodness of the global fit obtained by our model. However, the coefficient is not defined to describe non-linear variations or collinearity nor to indicate the existence of omitted-variable bias. Moreover, neither R^2 nor least squares provide evidence about whether the independent variables are suitable for the regression task.

As long as we want to infer in regression problems, we can utilize LR models to describe some responses. But, if we want to ascertain whether an event y belongs to some class, we will not be able to perform it with LR formulation, since it does not output probabilities. The linear structure cannot behave as a probability function either, so we need to create new techniques for classification tasks. The binary logistic model is a statistical estimator of probability events by calculating the log odds by using independent variables as predictors. It was one of the first solutions for solving binary classification as $\{0, 1\}$ labels, i. e. it is assumed that $y \sim \text{Bernoulli}(p)$ for some probability p. The logistic model computes the membership probability of an event through the logistic function, which has the formal representation:

$$\text{logistic}(t) = \frac{1}{1 + e^{-t}}. \tag{4.71}$$

It is easy to note that $\text{logistic}(t) \in (0, 1)$ per each $t \in \mathbb{R}$. A point to emphasize is that 4.71 is differentiable, and so, its derivative as well.

For a given dataset $\mathcal{D} = \{X_i, y_i\}_{i=1}^{N}$, such that $y_i \in \{0, 1\}$ indicates whether or not the response belongs to the studied class, we combine the predictor variables as a weighted aggregation and map it over 4.71 to obtain:

$$\begin{aligned} P(y_i = 1) = & \ \text{logistic}\left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_n x_{in}\right) \\ = & \ \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_n x_{in})}}. \end{aligned} \tag{4.72}$$

Given that $P(y_i = 1)$ is bounded, we have to consider a decision threshold so that it separates the results as 0 or 1. It is usually set as 0.5 because it is the half value of the output space. The error associated with the predicted class is evaluated with the cross-entropy loss (4.5). By doing so, we are considering the averaged probability log error as a measure of goodness of fit, i. e. comparing $P(y_i = 1)$ against y_i. An important point to highlight is that it is impossible to obtain a zero loss per any prediction since $y_i \in \{0, 1\}$ and $0 < P(y_i = 1) < 1$. It has a contrary interpretation to LR models, given that linear approximations may return similar values for some points. We can also note that we can link $\{P(y_i = 1), P(y_i = 0)\}$ with $\{y_i, 1 - y_1\}$ because we have assumed that y is a Bernoulli-distributed variable.

We can see the difference in the approach of linear regression and logistic models in the illustration in Fig. 4.8. The blue dots are the dependent variable to describe, and the red line indicates the solution of the fitting.

For interpretability purposes, now the β-coefficient states the weights of each explanatory variable in a different manner.

$$odds = \log\left(\frac{P(y_i = 1)}{1 - P(y_i = 1)}\right) = \log\left(\frac{P(y_i = 1)}{P(y_i = 0)}\right) = e^{\beta_0 + \beta_1 x_{i1} + \cdots + \beta_n x_{in}}. \tag{4.73}$$

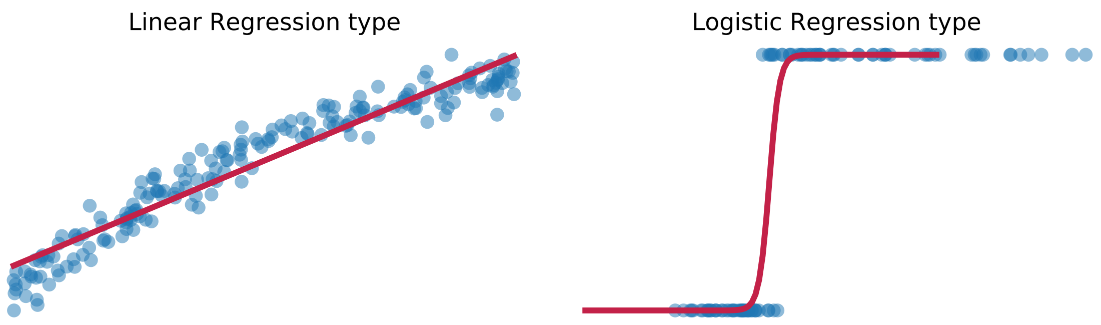


Figure 4.8: Difference between linear regression and logistic regression approaches. The red lines indicate the solution to the fitting.
Source: Own elaboration.

As a consequence, when we modify the value of the x_i variable to $x_i + \Delta_i$, we have an odds ratio of:

$$\frac{odds_{\Delta_i}}{odds} = e^{\beta_i \Delta_i} \tag{4.74}$$

Hence, the ratio variation of a change Δ_i is exponentially proportional to β_i.

4.1.2 *Principal tasks in Machine Learning*

In this section, we have mentioned the four different approaches that concern the development of ML models. Even though they represent distinguished fields, the tasks that could be executed can coincide.

4.1.2.1 *Classification and regression*

It is the assignment problem for a given dataset by supervised approach. In particular, the different values, categories, classes, groups, or types are assumed to be known and distinctive among the others. The main point is to generate a pattern recognition method by means of the known output (y_i). Hence, the classification algorithm is able to yield an accurate answer ($\hat{y}_i$) for a given input (X_i). The main difference between classification and regression tasks is that the first one aims to identify the relationship of a sample within a set of sub-populations (classes) by means of the so-called explanatory variables or feature vectors (Carrizosa and Romero Morales, 2013), and the second one, the main objective is to estimate the relationship between a set of dependent variables (outcome or response) and a set of independent variables (predictors) through a number of unknown parameters and the statistical error terms (Freund et al., 2006).

Owing to the nature of this thesis is mainly focused on presenting the supervised methodology, the list of algorithms, methods, and applications will be described throughout our work.

4.1.2.2 *Clustering*

Cluster analysis is considered one of the main pattern recognition techniques that concern unsupervised tasks (Diday and Simon, 1976), gaining a great deal of attention from the middle of the 20th century onwards (Blashfield and Aldenderfer, 1978). This task consists in finding the underlying structure in a collection of unlabeled data. Then, a cluster is a subset of elements of the original collection that contains objects with similar patterns among them (Madhulatha, 2012). It is important to remark that, while the goal of clustering analysis is descriptive, the goal of classification techniques is entirely predictive, so there should be no confusion between them. Therefore, the clustering's target is to discover a set of categories that remain unknown. Thus, we can recognize new groups that are of interest in themselves and their intrinsic assessment (Rokach and Maimon, 2005). As stated in Tryon and Bailey (1971), "*understanding our world requires conceptualizing the similarities and differences between the entities that compose it*".

According to Rokach and Maimon (2005), given a space $\mathbf{S}$, the clustering structure is represented as a set of subsets $C = \{C_1, \ldots, C_n\}$ of $\mathbf{S}$, such that,

$$\mathbf{S} = \bigcup_{i=1}^{n} C_i, \quad C_i \cap C_i = \emptyset, \quad \text{per each } i \neq j. \tag{4.75}$$

When solving a clustering problem, the task of finding a partitioning from a given data-set is usually performed by means of some optimality criterion (Duran and Odell, 2013). The purpose of the optimality criterion is to reflect the levels of the desirability of the various partitions. In general, clustering algorithms can be considered hierarchical or partitional. Hierarchical algorithms find successive clusters by taking into account previously established clusters, whereas partitional algorithms ascertain all clusters at time (Madhulatha, 2012). On the one hand, hierarchical algorithms rely on distance-based methods, in which any of the metrics defined in Table 3.2 may be considered to merge the different nested partitions and measure the linkage among clusters. An example of the application of this kind of algorithm is presented in Yim and Ramdeen (2015), where it is used for studying language variables for psychological purposes. On the other hand, partitional algorithms draw for the premise that k different clusters are considered beforehand. In turn, the algorithm searches for the best possible positions by iteratively reallocating reference points over the sample space. The search criterion is widely known as fitness measure (Nanda and Panda, 2014). One of the first partitional algorithms was the K-means (Lloyd (1982) but originally presented in 1957), which still remains popular due to its low-complexity attached (Jain, 2010). Over the years, various partitional algorithms have been implemented with different approaches. For instance, two well-known algorithms are the Partitioning Around Medoids (PAM) (Kaufman and Rousseeuw, 1990) or the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996).

4.1.2.3 *Dimensionality reduction*

It is the set of data transformation techniques from the actual N-dimensional space into a lower n-dimensional embed space. That task is usually performed when dealing with a large number of observations or large observational variables. Thus, we can limit the computational cost associated with processing all the information at once. Depending on the motivation of the problem, it is required to select or extract features from the original dataset, which divides the task into two subgroups.

This study is commonly known as manifold learning, due to the topological objects, and it relies on the assumption that the high data dimensionality representation is usually artificial because the features or parameters of each sample can be described by low dimension functions (Cayton, 2005). As a consequence, the implicit function theorem allows us to model the dataset as a low-dimensional manifold space embedded into the original one. Then, the set of algorithms for manifold learning is designed to extract the parameters that define the embedded representation.

A common example of dimensionality reduction is the principal component analysis (PCA) technique (Pearson, 1901), which is an unsupervised method for projection that maximizes the variance in the mapping space through matrix eigenvectors. PCA is also applied for exploratory data analysis to summarize the main features that compound the X_i input data. In a similar way, the linear discriminant analysis (LDA) technique (Fisher, 1936) seeks a linear combination that describes and/or distinguishes various classes. Hence, LDA can be applied for either dimensionality reduction (unsupervised) or classification (supervised) because it determines the effectiveness of the set of variables over the class membership. Finally, an example of artificial neural networks applied in this task is the Autoencoders, which reduce the feature space into a latent space by means of efficient codification of the input data. Their implementation is broadly used in examples as anomaly detection (Chen et al., 2018) and denoising techniques for clinical samples (Gondara, 2016), speech (Lu et al., 2013), and image (Majumdar, 2018). A widely recognized case of encoder-decoders is the transformers networks (Vaswani et al., 2017), being the state-of-the-art in natural language processing (Wolf et al., 2020).

4.1.3 *K-Nearest Neighbours*

Over the XX century, lots of algorithms and techniques have been deployed to solve classification a regression tasks. A major breakthrough occurred in the middle of the century when Fix and Hodges (1951) studied the consistency properties of non-parametric discrimination problems. For a random variable Z and two distributions F and G with known density functions f and g respectively, it is defined the likelihood ratio procedure over a threshold c for the quotient $f(z)/g(z)$. Motivated by the discrimination properties stated in Fix and Hodges (1951), Cover and Hart (1967) introduced the concept of K-Nearest Neighbours (KNN), a non-parametric method for estimation without making strong assumptions about the dataset. Further properties about the convergence of the algorithm were presented in Devroye (1978), and other basic concepts are explained in Altman (1992). The

KNN method is built by a distance function that determines the closeness over a sample in regard to the entire set of observations. The forecast is obtained by ranking such distances and the result is based on a voting system for the K neighbor elements. In Fig. 4.9 is illustrated the KNN functioning through three steps procedures: addition of a new unlabeled sample, calculation of the pair-wise distance comparison for the entire dataset, and voting system for the K nearest elements in the set.

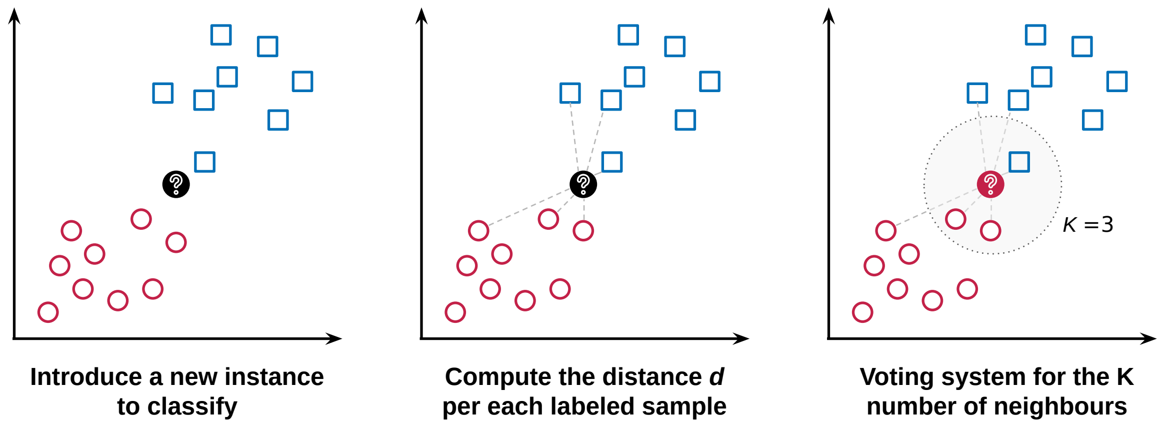

?
?
?
K =3
Introduce a new instance to classify
Compute the distance d per each labeled sample
Voting system for the K number of neighbours

Figure 4.9: Graphical interpretation of the KNN classification algorithm for binary classification (red/blue classes) with Euclidean distance and $K = 3$ neighbors.
Source: Own elaboration.

It is easy to note that the principal parameters that set the method are the number of neighbors $K \in \mathbb{N}$ and the distance function considered d to determine the pair-wise proximity to the resultant elements. The distance usually utilized for KNN is the Euclidean distance for continuous variables or the Hamming distance for discrete variables, although the metric selection can also be attached with supervised metric learning (the definition can be found in Wang and Sun (2015)). The choice of K also has a significant impact on the method because it determines the misclassification error (Hall et al., 2008). On the one hand, larger values of K lead to less chance of error but it is prone to high computational times when predicting. On the other hand, smaller values of K ease the decision process but they ascertain non-precise decision boundaries when there exist noisy features. In practice, it is commonly extended the use of heuristic techniques such as empirical study (Chomboon et al., 2015) or hyperparameter optimization (Feurer and Hutter, 2019). The last matter to emphasize is the voting system utilized to predict the class of an instance because the original KNN makes use of straightforward counting. This concern is harder to overcome since it cannot be handled by means of numerical experiments. However, it has been demonstrated that the use of weighting schemes via distance-to-sample to produce class probability estimation outperforms the default KNN strategy (Jiang et al., 2007).

Over the years, the structure of the KNN algorithms has considerably evolved, where the target data and procedure are responsible for the generalization of such techniques (Bhatia and Vandana, 2010). The KNN performance has also been a matter for study with efficiency improvement obtained by Sproull (1991) and Liu et al. (2003). With respect to the K-Nearest Neighbours applications, its predictive power has been shown in multiple tasks such as economic forecasting (Imandoust and Bolandraftar, 2013), television advertisement rating (Hariadhy et al., 2021), tumor diagnosis (Yuan et al., 2004), and text categorization (Guo et al., 2004).

4.1.4 *Support Vector Machine*

Within the family of constructive models, Boser et al. (1992) proposed a supervised algorithm for classification that relied on the marginal distance between classes in the direct space. The basis for implementation and empirical risk minimization was already studied by Vapnik (1992). The point was to obtain the best statistical generalization performance in a very high-dimensional feature space via separation properties (Vapnik, 1999). A widely known example is the Support Vector Machine (SVM) of Cortes and Vapnik (1995). SVMs are a set of robust predictors based on non-probabilistic methods. Their strategy relies on the search of a hyperplane $w^\top \cdot x + b = 0$ so that it maximizes the distance between the different class events. The w vector is the normal vector of the hyperplane and it is responsible to separate the feature space generating a margin of size $\frac{b}{||w||}$. In such a way, we can distinguish three hyperplanes.

$$w^\top \cdot x + b = \begin{cases} 1; & 1 - \text{class decision boundary}, \\ 0; & \text{Optimal hyperplane}, \\ -1; & -1 - \text{class decision boundary}. \end{cases} \tag{4.76}$$

In practice, it is just a technique via the geometric constructive procedure. The support vector machine models can be divided into linear and nonlinear implementations (Hastie et al., 2009). In the particular case when the data domain can be divided linearly to separate the classes, it is called linear-SVM. From a topology perspective, a data domain $\mathcal{D}$ is linearly separable by a support vector $w^\top \cdot x + b = 0$ if there exist two subdomains that make a disjoint union of $\mathcal{D}$.

$$\begin{aligned} D_1 &= \{X : w^\top \cdot x + b \leq 0\}, \\ D_2 &= \{X : w^\top \cdot x + b > 0\}. \end{aligned} \tag{4.77}$$

As a consequence, the subdomains D_1 and D_2 determine the different classes, as indicated in Eq. 4.76. The optimization strategy is basically a search of the parameters that maximize the marginal distance between classes (Suthaharan, 2016). Similarly, a multiclass support vector machine can be deployed as a combination of various binary-class support vectors together with an ensemble approach (Franc and Hlavac, 2002).

For the cases when the data is not linearly separable, some extensions transform the data domain into a high dimensional feature space to solve the prescriptive procedure called non-linear-SVM (Suthaharan, 2016). The mapping functions that convert the original space into the feature space are known as kernel functions (Scholkopf et al., 1999), and so the support vector can be rewritten as $w^\top \phi(x) + b$, where $\phi : \mathbb{R}^N \to \mathbb{R}^{N'}$ for $N' >> N$.

In Fig. 4.10 is depicted two simple 2-dimensional examples of linear and non-linear SVM problems.

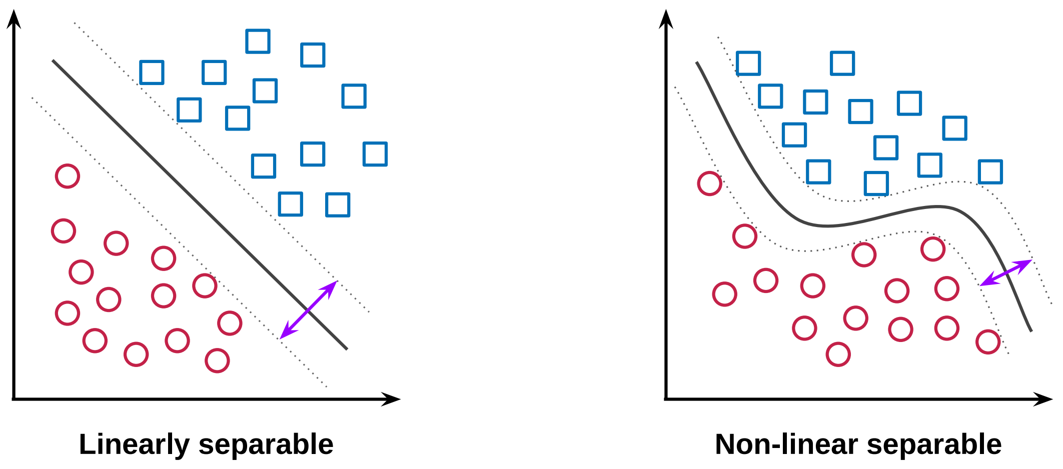


Figure 4.10: Maximum-margin hyperplane and class-margins for a support vector machine model in two binary classification tasks. The left plot depicts a problem in a linear separable space and the right plot shows the decision boundaries after a mapping over a kernel. Source: Own elaboration.

For both cases, the optimization problem associated with the fitting stage can be performed by means of the following expression of Bottou and Lin (2007):

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}||w||_2^2, \\ \text{s.t} \quad & S(w^\top \phi(x) + b) \geq 1, \\ & \forall i \in \{1, ..., N\}. \end{aligned} \tag{4.78}$$

where S is the response function for each X_i sample over the N'-hyperplane.

In the field of machine learning, support vector machines have been considered one of the most important algorithms due to their friendly usage and great performance (Pradhan, 2012). Their applicability has been demonstrated in various fields such as bioinformatics (Byvatov and Schneider, 2003) and chemistry (Chen et al., 2004). Among many interesting applications of SVM algorithms, it is worth mentioning the machine condition monitoring (Widodo and Yang, 2007) and web content classification (Sun et al., 2002). Other approaches can be found as the Lagrangian matrix expansion (Balasundaram and Kapil, 2010), quantum assisted (Rebentrost et al., 2014), fuzzy structured (Wang et al., 2005) or wavelet-based kernel (Zhang et al., 2004) types of support vector machines.

4.1.5 *Decision Trees*

In graph theory, a tree is a connected, acyclic, and undirected graph formerly defined by Arthur Cayley (1857). Within the field of ML, a decision tree (DT) is a tree-like structure defined as a graph $G = (V, E)$ with the set of nodes V and edges E. Its nodes can be considered as decision, chance, and terminals that define a disjoint collection as $V = \mathcal{D} \cup \mathcal{C} \cup \mathcal{T}$ respectively. A tree is built by a root node, leaf nodes that represent the classes, and internal nodes that represent the test conditions. In Fig. 4.11 is depicted as a graph structure of a decision tree.

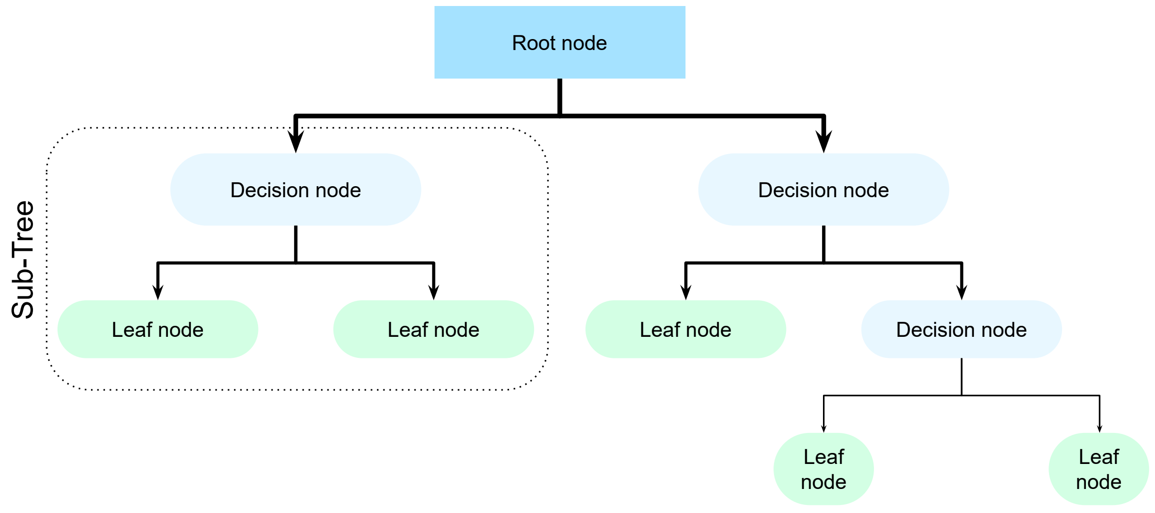


Figure 4.11: Decision tree graph representation with its nodes and vertices.
Source: Own elaboration.

Decision trees were designed to cover both regression and classification. The first regression instance was named automatic interaction detection (AID) by Morgan and Sonquist (1963) and the first classification instance was developed under the name θ-AID (Messenger and Mandell, 1972), however, it took a long time until classification and regression trees (CART) by Breiman et al. (1984) made a breakthrough in the DT field.

The vertices that define the paths are constructed via inductive rule, where the node split is defined by the association of such decision rules. For a given node t with two leaves t_L and t_R, we can mathematically represent the node "impurity" as a function that computes the deviation of the splitting, some examples are presented as follows (Loh, 2014):

$$\phi(t) = \sum_{i \in S(t)} (y_i - \hat{y}_t)^2, \tag{4.79}$$

$$\phi(t) = \sum_{i \in S(t)} \frac{(y_i - \hat{y}_t)^2}{y_i} \quad \chi\text{-squared}, \tag{4.80}$$

$$\phi(t) = -\sum_j p(j|t) \log p(j|t) \quad \text{Entropy}, \tag{4.81}$$

$$\phi(t) = 1 - \sum_j p(j|t)^2 \quad \text{Gini index}, \tag{4.82}$$

where $S(t)$ is the set of data after t and $p(j|t)$ stands for the observations of the j-class proportion. With the use of $p(j|t)$ we can also compute the classification error as:

$$\text{Classification error}: \quad CE(T) = 1 - \max_j p(j|t). \tag{4.83}$$

As a manner to control the tree complexity, it was included the search for the most appropriate descriptor per each split by calculating the best reduction in impurity between a decision node and its leaves (Deconinck et al., 2005). Tree pruning is an alternative to control both overfitting and outlier influence. This strategy compares the branches to find the least complex one, in which multiple versions can be defined (Esposito et al., 1997). The manner DTs classify input data is by means of the concept of information gain. The entropy (H) is originally used as the measure to count that gain because it quantifies the disorder or uncertainty of data. Then, the conditional entropy is just the j-entry of the summation and so the information gain associated with the tree hit as:

$$\text{Information gain}: \quad \nabla I(p,j) = H(p) - H(j|p). \tag{4.84}$$

Finally, we can use the gain ratio to quantify the quantity of information obtained in the splitting:

$$\text{Gain ratio}: \quad g(p,j) = \frac{\nabla I(p,j)}{H(p)}. \tag{4.85}$$

During the fitting, the gain ratio gives us information about the goodness of a split (Singh and Gupta, 2014). Additionally, it helps to reduce the favor given by entropy or the Gini index for outliers.

4.1.5.1 *Random Forest*

With the aim of preventing the limitations presented in DTs and improving their performance, Ho (1995) designed the Random Forest (RF) model. It consists of a tree-based algorithm that makes use of stochastic modeling to combine each subtree structure via i.i.d. random vectors. In its formal definition, Random Forest is an ensemble of a finite number of trees $\{h(X,\theta_j)\}_{j=1}^{n}$ where each θ_j is an independent identically distributed random vector and each $h(X,\theta_j)$ tree unit contributes equally to the final output (Breiman, 2001). The incorporation of the random component θ_j allows us to add randomness in two ways (Cutler et al., 2012). First, an independent bootstrap from the original data is used to fit the trees. Second, considering the inputs $X_i = (x_{i1}, \dots, x_{ik})$, it is selected a subsample of $l \leq k$ to search for the best split independently at each node.

Once we have computed the basic steps to insert the randomization to the learning trees and we have fitted the decision trees $\hat{h}_i$, we can predict each new input $F(x)$ as:

$$\text{Regression}: \quad \frac{1}{k}\sum_{j=1}^{k} \hat{h}_j(x) \tag{4.86}$$

$$\text{Classification}: \quad \underset{y}{\text{argmax}} \sum_{j=1}^{k} \mathbf{1}_{\{\hat{h}_j(x)=y\}}(x) \tag{4.87}$$

Given that the bootstrap sample is taken as a subset of the dataset $\mathcal{D} = \{X_i, y_i\}_{i=1}^{N}$, there will be another subset that contains the "out-of-bag" data as $\mathcal{I} = \{i : (X_i, y_i) \notin \mathcal{D}\}$. Hence, such a set is very useful for estimating generalization error and variable importance (Cutler et al., 2012).

Finally, it is essential to highlight that the generalization error for random forest converges to a bounded limit when the number of trees h_j tends to infinite (Breiman, 2001). Additionally, such error relies on the strength and correlation among trees.

4.1.5.2 *Gradient Boosting Machines*

Gradient Boosting Machines (GBM) is a set of algorithms that learns from data via weak learners through an additive approach (Friedman, 2001). The underlying idea relies on the iterative search of an approximation function that accurately maps the response of a given training data sample $\mathcal{D} = \{X_i, y_i\}_{i=1}^{N}$ of known values. Regarding a loss function $\mathcal{L}(y, \cdot)$, GBMs estimate the objective function of the minimization problem that implies the fitting of the joint distribution of (X_i, y_i)-pairs. In turn, the optimal function may be defined as in Eq. 4.88:

$$F^*(\mathbf{X}) = \underset{F}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{X},y}\mathcal{L}(y, F(\mathbf{X})). \tag{4.88}$$

Now, we build the additive expansion of an approximation function $F(\mathbf{X}; \mathbf{a})$ where $\mathbf{a}$ is the set of characteristic parameters. Thus, we can estimate the objective by following the ensemble technique of the Eq. 4.88, i. e. by a weighted accumulation sum of learners.

$$\begin{cases} F_m(\mathbf{X}) = F_{m-1}(\mathbf{X}) + \rho_m h_m(\mathbf{X}; \mathbf{a}). \\ F_0(\mathbf{X}) = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(y_i, \alpha). \end{cases} \tag{4.89}$$

Here, h_m are the weak learners, and ρ_m are the multipliers of the linear search. The iterative upload steps of the algorithm are described as the procedure described in Eq. 4.90. The underlay goal is to greedily improve the model performance by minimizing the loss function at each time (h_m) via linear search (ρ_m).

$$\begin{aligned} \tilde{y}_{i,m} &= -\left[\frac{\partial \mathcal{L}(y, F_{m-1}(X_i))}{\partial F_{m-1}(X_i)}\right]_{i,m}. \\ \mathbf{a}_m &= \underset{\mathbf{a},\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left[\tilde{y}_i - \beta h(X_i, \mathbf{a})\right]. \\ \rho_m &= \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(y_i, F_{m-1}(X_i) + \rho h(X_i; \mathbf{a}_m)). \\ F_m(\mathbf{X}) &= F_{m-1}(\mathbf{X}) + \rho_m h(\mathbf{X}; \mathbf{a}_m). \end{aligned} \tag{4.90}$$

We can use this procedure to minimize any differentiable loss $\mathcal{L}$ with forward stagewise additive modeling. It fits the learners $h(\mathbf{X}, \mathbf{a})$ to the responses of the approximated member of the pseudoresponses $\tilde{y}_i$ of the loss function in the steepest-descent strategy. For the particular case in which we consider decision trees as base learners h, then we are focusing on the Gradient Tree Boosting (GTB) methodology (Friedman, 2002).

4.1.5.3 *Extreme Gradient Boosting*

A particular case of GBs is the Extreme Gradient Boosting (XGBoost) defined in Chen and Guestrin (2016), which is a GTB ensemble with a scalable end-to-end system. The approximation method for the optimal search is based on a similar approach as the GBM additive procedure described in § 4.1.5.2. Nevertheless, XGBoost executes decision trees as weak learners, then the loss function is regularized in order to penalize the tree complexity. By doing so, we also avoid over-fitting. The representation is described in equation 4.91.

$$\mathcal{L}_{XGB}(y, F(\mathbf{X})) = \mathcal{L}(y, F(\mathbf{X})) + \sum_k \Omega(h_k), \tag{4.91}$$

where $\Omega(h_k) = \gamma T + \frac{\lambda}{2}||w_{h_k}||_2^2$.

In particular, considering T trees, Ω prevents over-fitting by smoothing the learned weights (w). The parameter γ regularizes the loss reduction gain and so determines the complexity of the tree base learners. The λ parameters control the impact of the weights over the loss. It is easy to note that, when $\gamma = \lambda = 0$ in equation Eq. 4.91, we have the same methodology as tree-GBMs.

During the fitting phase, the XGBoost loss can be controlled through the formal definition of the iterative process. Now, we can build the sequence described for GBMs (Eq. 4.89) as an iteration k over the loss equation Eq. 4.92.

$$\mathcal{L}_{XGB}(y, F_k(\mathbf{X})) = \sum_{i=1}^{N} \mathcal{L}\left(y_i, F_{k-1}(X_i) + h_k(X_i)\right) + \sum_k \Omega(h_k). \tag{4.92}$$

XGBoost is known to be one of the state-of-art techniques in machine learning. Its effectiveness and applicability have been proven in several challenging areas of classification (Ogunleye and Wang, 2020) and regression (Gumus and Kiran, 2017), proving an excellent performance with easy and scalable launching. Moreover, XGBoost is also a good classifier in imbalance tasks such as financial assessment (Chang et al., 2018), credit scoring (He et al., 2018), or face image manipulation (Dang et al., 2019).

4.2 DEEP LEARNING

Deep learning is a subfield of machine learning algorithms (as shown in Fig. 4.3) characterized by artificial structures (Shinde and Shah, 2018). It allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction (LeCun et al., 2015). In the same spirit as traditional ML, deep learning can solve any of the tasks described in § 4.1.2, although deep models have been mostly designed to tackle unsupervised problems (Goodfellow et al., 2016). The explanation of this is that data processing through multiple customized mathematical operators (Vidal et al., 2017) can extract high-level features from input data (Deng and Yu, 2014).

4.2.1 *Neural Networks*

The nervous system is responsible to conduct electric signals through the organism in order to send a response for a given stimulus, which will be turned into action in sensory information. A neuron is an electrically excitable cell that constitutes the basic component of the nervous system. It is presented in all animals but sponges and placozoa, in addition, plants and fungi have different structures. A typical neuron can be classified into three types: sensory neurons respond to stimulus, motor neurons coordinate the signal and the locomotive system, and interneurons connect one neuron to another making a neural circuit. In general, the structure of a neuron can have many different shapes and sizes, but it is commonly divided into the cell body, dendrites, and axons. Their basic function is to coordinate the information via complex and specialized feedback known as synapses, in which an electrical or chemical signal is transmitted from one to another neuron.

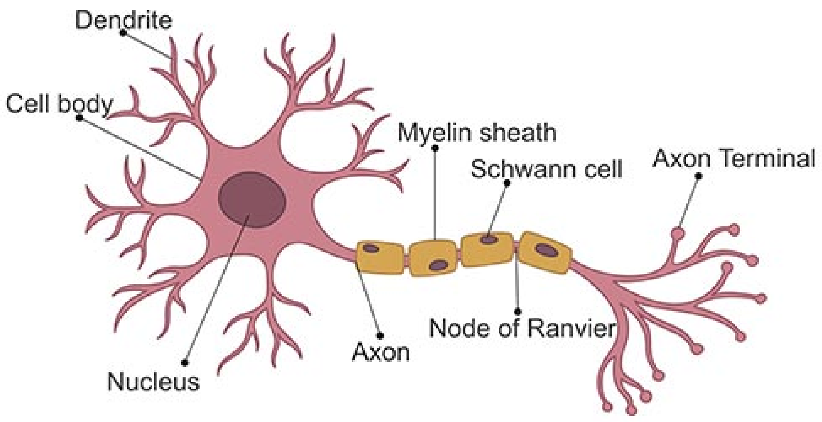


Figure 4.12: Structure of a biological neuron divided by its major regions.
Source: CUSABIO.

The biological study of neural circuits started in the 19th century. Santiago Ramón y Cajal proposed the behavior of neurons as basic functional elements of the nervous system. In his research, he stated that neurons perform as discrete cells that transmit information through electric connections. That pioneer idea is the basis of neuron doctrine and modern neuroscience. The contributions of Ramón y Cajal were recognized with the Nobel Prize in Medicine in 1906, shared with Camillo Golgi.

Once neuroscience advanced, the study of neural circuits and their anatomy became a consolidated field of biology. In parallel with the study of the nervous system of humans, scientists also started to analyze the behavior of synapses to develop intelligent systems. In 1943, McCulloch and Pitts (1943) developed the concept of artificial neurons as a computation unit that mimics the behavior of a biological neuron in a very simple formulation. The logical gate implemented processed the input and fitted the response by using a decision threshold. The idea of utilizing a threshold was to distinguish spatial sections under the assumption of linearly separable problems. It was the first attempt at an implementation of an artificial neuron system. We can represent it with formal mathematics as:

$$f_\theta(x) = \begin{cases} 1 & \text{if} \quad w^\top \cdot x \geq \theta, \\ 0 & \quad \text{otherwise.} \end{cases} \tag{4.93}$$

Their system was called the MCP neuron, but it is also known as the Threshold Logic Unit or Linear Threshold unit. The convex combination of weights $w^\top \cdot x = \sum_{i=1}^{m} w_i x_i$ determines the spatial regions, which are activated via step function. Such weighted product of vectors of m-dimension is fitted to the input to give the most accurate response for the given data. In other words, it is required to optimize the parameters $w_1, \ldots, w_m$ in order to fit the actual distribution of some event. They are called neurons or parameters of the gate function. This kind of artificial neuron operates over the logical identities based on the following Table 4.2.

Table 4.2: Basic truth tables for all binary logical combinations: P and Q.

P	Q	AND	OR	NAND	NOR	XOR	XNOR	$\to$	$\leftarrow$
T	T	T	T	F	F	F	T	T	T
T	F	F	T	T	F	T	F	F	T
F	T	F	T	T	F	T	F	T	F
F	F	F	F	T	T	F	T	T	T

In 1957, Rosenblatt made an impressive breakthrough with the implementation of an algorithm known as a perceptron, Rosenblatt (1957, 1958, 1963). Its simple implementation did not take long in becoming widely popular. Such an artificial neuron was designed to compute a binary classification by using step functions over a linear combination of features. The main idea of the function is to divide the input into regions and determine which of them are representative when making a decision. A simple mathematical formulation of a perceptron function is presented as follows:

$$\rho(x) = \begin{cases} 1 & \text{if} \quad \phi(w^\top \cdot x + b) \geq 0, \\ 0 & \quad \text{otherwise.} \end{cases} \tag{4.94}$$

For artificial neurons, the concept of weights defined by the perceptron functions is quite different than the idea of the weighting scheme discussed in Chapter 3. Additionally, we have the notion of bias, i. e. the parameters that control the spatial position of the features studied.

In the equation given above, the function maps the x input by combining it with a weighted aggregation. A major advance in difference with Eq. 4.93 is the use of customizing activation functions ϕ. For feed-forward networks, the activation used to transfer the information to the output is the logistic function (Eq. 4.71). Depending on the scope of the problem, they are expected to process the weighted sum ($w^\top \cdot x$) and adjust an adequate output. For instance, multi-classification problems require different activation functions than regression problems. Unlike the Linear Threshold unit, now we have a bias b which is a new parameter to fit when fitting the function. In any case, the Threshold Logic Unit

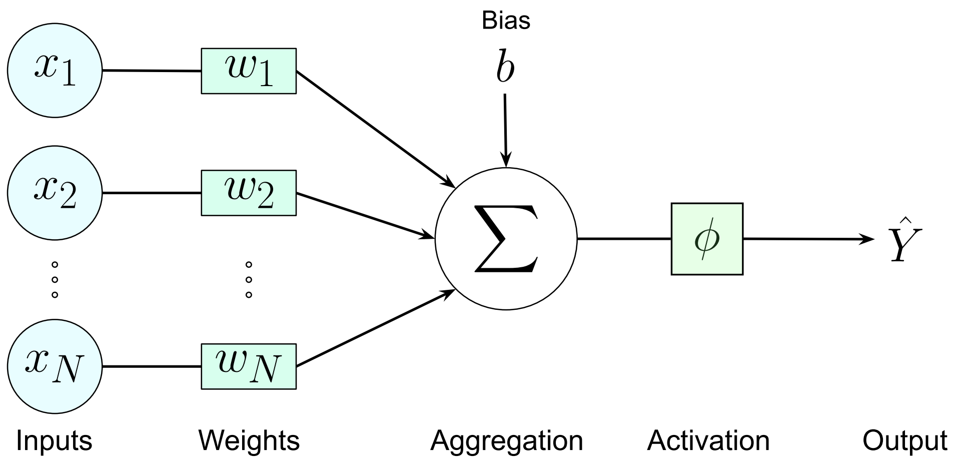


Figure 4.13: Implementation of the generalized perceptron function.
Source: Own elaboration.

can be understood as a particular case of the perceptron function of Rosenblatt. As described in Fig. 4.13, the artificial neuron of McCulloch and Pitts has a fixed activation function (Heaviside step) and a fixed bias parameter as the decision threshold.

In regard to activation functions, they are responsible to transmit the processed input data to the response. Then, an activation determines how the model propagates the information to the output (Sazli, 2006). Actually, sometimes the activation is also called a transfer function. When performing AI tasks, the choice of these functions relies on the objective of the problem. For example, some specific functions just belong to the family of regressors and others can only be applied when performing pattern recognition. In any case, a crucial requirement for the activation function is that it has to be bounded in their domain since the operation of FNN is based on the universal approximation theorem (Laudani et al., 2015). On the whole, we can distinguish three different categories of functions:

- Ridge: Actuates over linear combinations of the input variables.
- Radial: Actuates with a metric or distance established over some origin.
- Folding: Processes the input via high-order functions recursively.

In order to give a graphical representation of the commonly used activation functions, Fig. 4.14 depicts the behavior of ten different $\mathbb{R}$-valued activation functions defined in $[-3, 3]$. Thus, we can also analyze their image of them and their differential properties.

When referring to Deep Learning, a parallel concatenation of them is called neural network (NN), and it is considered the most basic feed-forward neural network (FNN) (Schmidhuber, 2015). It happens because the information only flows forward, in which its weights are responsible to learn from the input data. It was subsequently generalized to multiclass-perceptrons in order to tackle the simultaneous labeling problems. Thanks to the definition of perceptron and the construction NNs, nowadays we can build artificial neural networks (ANN) with millions of parameters. This architecture is popularly called as Multi-Layer Perceptron (MLP) (Hastie et al., 2009). In Fig. 4.15 is depicted the information flow over an FFN. The formal definition of each layer of the perceptron is a hidden

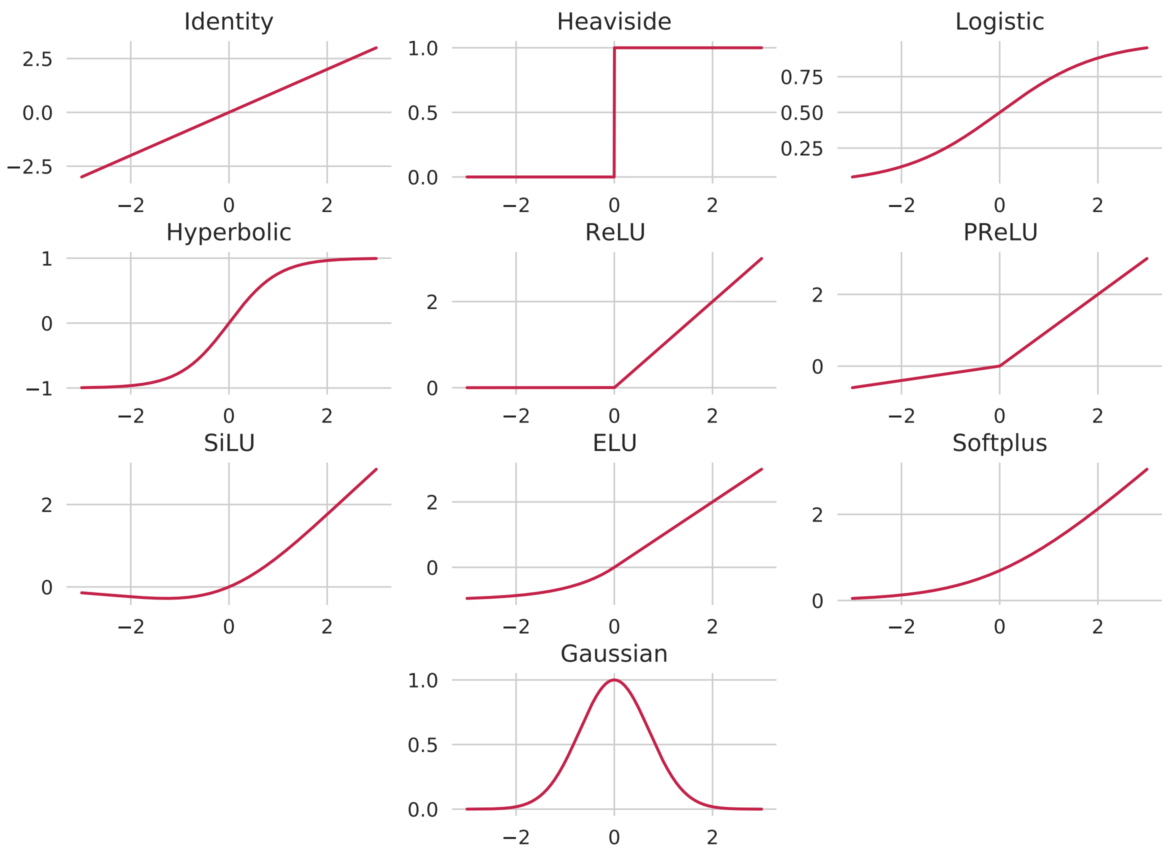


Figure 4.14: Ten different types of activation functions for neural networks.
Source: Own elaboration.

layer since the idea is that it is connected between the input in the output and it is sequentially applied through the Chain's rule. In general, the number of layer nodes is usually set as $N \geq l_1 \geq \cdots \geq l_n \geq k$, being N the size of the input, l_i the number of neurons in the ith layer, and k the output length. It is important to bear in mind that for each node of these hidden layers the workflow is the same as described in Fig. 4.13.

It is noteworthy to mention that deep neural structures may also have attached serious problems. A powerful system made by multiple concatenated neurons is able to learn features unnoticeable to the human eye, so the ANN can also learn non-representative elements from a given dataset, such as statistical noise (Bebis and Georgiopoulos, 1994). Hence, such models are prone to overfitting, causing null applicability of the network. As a manner to solve that concern, Srivastava et al. (2014) introduced a straightforward way to prevent it via intermediate layers that randomly drop units (along with their connections) from the neural network during training. These kinds of layers are called dropout regularization layers and they are very effective to prevent the excessive adaptation of the perceptron units. Fig. 4.16 depicts the application of a dropout within a neural network.

Over the years, deep structures of dense layers fully connected have shown great performance in both supervised (Murtagh, 1991) and unsupervised problems, demonstrating a state-of-the-art methodology (Cheng and Titterington, 1994; Delashmit and Manry, 2005). Among the great multitude of applications in multiple fields, we can highlight contributions to subjects such as industrial anomaly detection (Aldakheel et al., 2021), financial

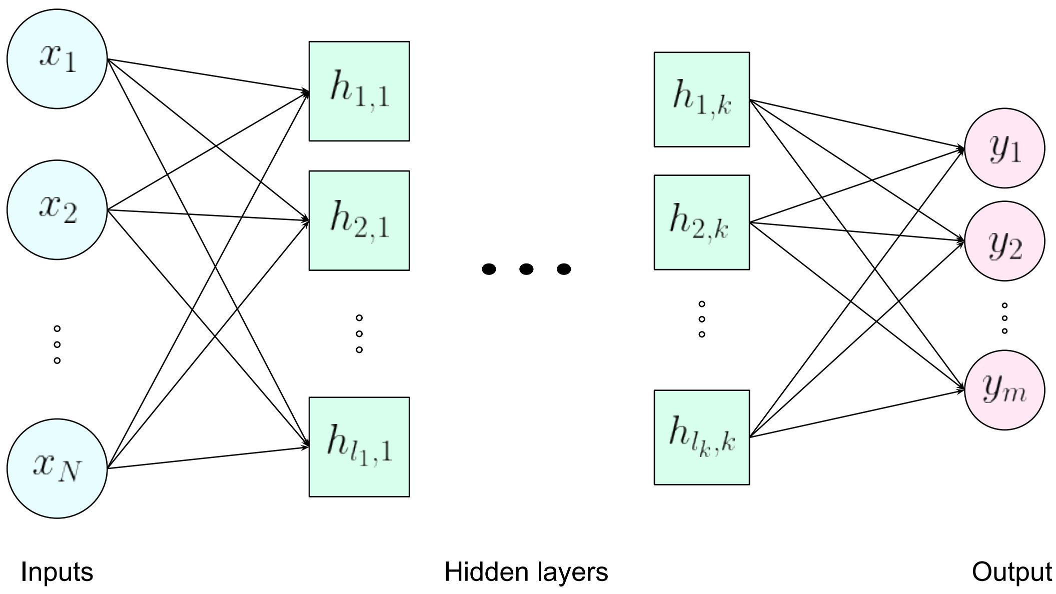


Figure 4.15: Architecture of a feed-forward model with k hidden layers.
Source: Own elaboration.

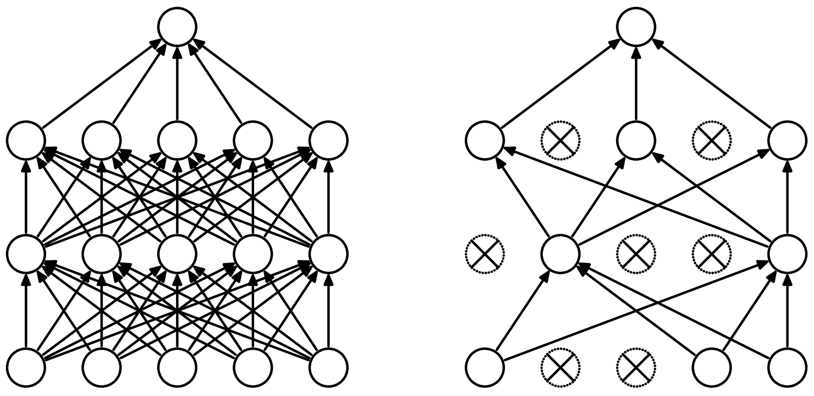


Figure 4.16: Neural net workflow trained with standard structure (left) and with dropout regularization after each layer (right).
Source: Srivastava et al. (2014).

forecasting (Corchado et al., 1998), chemistry pattern recognition (Burns and Whitesides, 1993), and survival analysis (Biganzoli et al., 1998). As a consequence, at the end of the XX century, there was a high interest in this subject (see Wasserman and Schwartz (1988) or Zhang (2000)). Apart from the structure and functionality of FNN, their performance is supported by Cybenko's theorem, which states that finite linear combinations of compositions of fixed univariate function and a set of affine functionals can uniformly approximate any continuous function of N real variables with support in the unit hypercube (Cybenko, 1989).

4.2.2 *Convolutional Neural Networks*

In functional analysis, convolution is a mathematical operator that is applied over two functions to produce a resultant one. This function is computed as the integral of the product over the domain in which the original functions interact. The mathematical formulation for two Lebesgue integrable scalar functions f and g in $\mathbb{R}^N$ may be written for element $x \in \mathbb{R}^N$ as:

$$(f * g)(x) = \int_{\mathbb{R}^N} f(\tau)g(x-\tau)d\tau, \quad \text{Continuous version.} \tag{4.95}$$

$$(f * g)(x) = \sum_{t \in T} f(t)g(x-t), \quad \text{Discrete version.} \tag{4.96}$$

It is easy to note that convolution is an operator on the linear space of integrable functions, which yields interesting properties such as commutativity, distributivity, and associativity (functional and scalar). In addition, when considering Lebesgue spaces the convolution operator measure is dominated by the used functions, i. e. whenever $f \in L^1(\mathbb{R}^N)$ and $g \in L^p(\mathbb{R}^N)$ with $p \geq 1$, then $f * g \in L^p(\mathbb{R}^N)$ and $||f * g||_p \leq ||f||_1||g||_p$. Owing to the special properties presented in that operator, convolution is widely utilized in many areas of mathematics, physics, and engineering.

Convolutional Neural Networks (CNN) is a type of ANN whose aim is to perform convolutional operators for the input data with a subsequent inner product. The key point is the kernel of the operation since it defines the dimensions (height, width, and depth) passed through a feature map followed by a number of strides that determines the spatial dimensionality to operate. Such kernels contain the weights, and so they are responsible to learn from data (O'Shea and Nash, 2015). Convolutional layers, similar to FCNs, also expand the inputs to extract patterns for data, although they are designed to extract spatial components by means of convolutional operators. As a result, each stage of the convolutional process for training AI models processes the input by means of local transformations, in turn, each model can learn from robust spatial features that characterize the dataset utilized (Albawi et al., 2017). Another particularity of CNN layers is that their neurons only convey the information via small regions of the preceding layer, which gives them a special behavior when handling data.

In regard to deep learning, CNNs are considered one of the most used types of ANNs (Li et al., 2021). CNNs were mainly defined to solve difficult tasks related to image-driven pattern recognition. Its particular manner to operate with input tensors makes them shift/space invariant thanks to their inner structure. Then, they are very useful to capture and extract spatial features that describe either the behavior or representation of some subject. Even though the convolution operator is the decisive factor of the CNNs, their architecture is also composed of more elements (Wu, 2017).

1. Kernels It is responsible to learn from the input samples, so they define how the model process the data. It is designed to work with small spatial dimensionality and spread it all over the depth of the network.

2. Pooling layer: It is responsible to reduce the dimension of feature maps via summarization techniques. It is usually performed over small portions of data with common maximum or average transformations.

3. Activation function: It is the same concept in every neural network and works the same way.

4. Normalization: It is responsible to normalize the output after the kernel process. Sometimes this element is suppressed, although it is statistically recommended to standardize the result before the following step.

5. Fully connected layer: It is responsible to connect the neurons of the last layer to the desired output. Sometimes, it is utilized a flattened layer (i. e. from tensor to 1D-vector) in order to perform a total combination of the neurons.

The combination of the convolution operator, kernel, activation, normalization, and pooling is called a convolutional block. In a mathematical 2D representation, assuming a filter size of $n \times m$ for a kernel with parameters W^l at the depth l, if our resultant tensor h^l has $N \times M$ size the block performs as:

$$h_{ij}^{l+1} = \sum_{a=0}^{n-1} \sum_{b=0}^{m-1} w_{ab}^{l} h_{i+a,j+b}^{l}. \tag{4.97}$$

Then, it is applied an activation ϕ, which is widely used ReLU (Fig. 4.14), and a normalization ψ, where the common one is the so-called batch normalization or statistical standardization (Table 3.1).

An instance of a CNN model is depicted in Fig. 4.17, where an image classification task motivates the use of such a network.

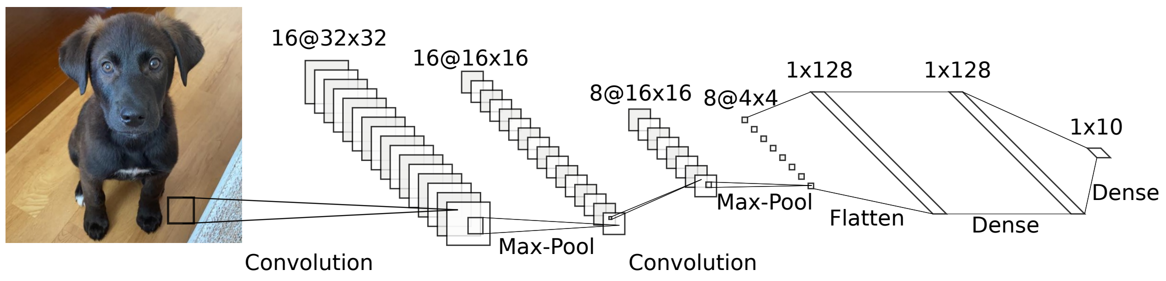


Figure 4.17: Topoogy of a CNN model designed to classify 10 different animals. In this case, it is processing a beautiful dog.
Source: Own elaboration built with NN-SVG.

Over the years, the complexity attached to the deep structures of CNNs has exploded. One of the earliest CNN in achieving global recognition was presented by LeCun et al. (1989) under the name of LeNet. It was subsequently improved, with great performance in handwritten digit recognition, and named LeNet-5 (LeCun et al., 1998). Afterward, it took time until Krizhevsky et al. (2012) presented AlexNet for image classification purposes, successfully competing in the ImageNet challenge. The developments in comparison with LeNet are remarkable in terms of input RGB format, deep structure, incorporation of dropout layers, and a high number of output responses among many others. Since 2012 until now, there have been considerable advances in the design of new convolutional architectures for general applications, although highly focused on image-related tasks. The major developments came from VGG-nets (Simonyan and Zisserman, 2014), residual networks (ResNets) (He et al., 2016), and MobileNets (Howard et al., 2017) (and its further versions Howard et al. (2019) and Sandler et al. (2018)).

4.2.3 *Recurrent Neural Networks*

Given that traditional neural networks cannot convey the temporal information of the times series, Recurrent Neural Networks (RNN) were designed in order to address this problem with their inner loop sequence. These networks are characterized by their infinite impulse response and the graph representation that they have. Its internal structure connects the data previously processed with the following input, as shown in Fig. 4.18, so they can learn the underlying temporal characteristics and apply them successively. This allows it to convey the dynamic patterns of the datasets as well as connect them recursively with a self-feedback. This forward direction described is responsible to analyze input data in the series-time domain.

Let h_t be the impulse response of an RNN, a simple approach of RNNs is the Elman networks (Elman, 1990), in which the activation function is applied over a copy of the hidden response, with a definition as follows:

$$\text{Hidden vector} \quad h_t = \sigma_h \left(W_h x_t + V_h h_{t-1} + b_h \right) \tag{4.98}$$

$$\text{Output vector} \quad \hat{y}_t = \sigma_y \left(W_y h_t + b_y \right) \tag{4.99}$$

In each equation, x_t represents the sequenced input at time t, $\hat{y}_t$ is the output generated, h_t is the impulse response of the RNN block, W and V the weights matrices, b the bias, and σ are activation functions. The σ function commonly used is the sigmoid function because the response ranges between 0 and 1, so the output eases the weight update. As stated in Eq. 4.98 and Eq. 4.99, the activation functions may be different depending on the aim of the application.

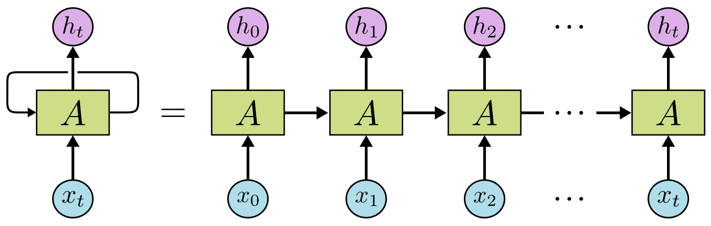


Figure 4.18: RNN layer architecture and its serialization over t steps.
Source: Own elaboration adapted from ResearchGate.

We have decided to emphasize the Elman approach because it is the recursive method that we have implemented for the RNN examples in 6.2. It is considered a Simple Recursive Network (SRN) for its blocks and its input-output representation. Nevertheless, there exist other kinds of SRNs such as Jordan networks (Jordan, 1997), Recursive Auto-Associative Memory (RAAM) (Pollack, 1990), Gated Recurrent Units (GRU) (Dey and Salemt, 2017), and Independently RNN (IndRNN) (Li et al., 2018) among many others.

Since its first release, many authors have utilized the recursive networks in many different branches. Actually, the efficacy of its application has been proven to solve various machine learning problems such as forecasting (Zhang and Xiao, 2000), prediction (Fan et al., 2017), anomaly detection (Huang et al., 2019), outlier detection (Williams et al., 2002), translation (Cho et al., 2014), handwriting recognition (Koutník et al., 2014), speech recognition (Miao et al., 2015), audio generation (Koutník et al., 2014), and sentiment analysis (Basiri et al., 2021).

The implementation of recurrent neural networks made a breakthrough in the area of machine learning. However, there are some limitations that simple recurrent blocks cannot overcome. Since the development of SRNs, it has been proved that these models are unable to learn long-time sequences. Another limitation that is commonly reported is the behavior of gradients caused by backpropagation through time because the sum of gradient errors tends to explode or vanish.

4.2.4 *Long-Short Term Memory*

A particular case of RNNs is the LSTM (Long Short-Term Memory) layers introduced in Hochreiter and Schmidhuber (1997). They are a gradient-based model that was developed to adapt the long-term dependencies through a cell state that joins the entire model. The merge of short and long properties of a series gives the LSTM layers more advantages when learning from data compared to simple RNN layers. Thus, they are able to connect larger time adequately lags with lower error. There is a limitation of the back-flow error since it is enforced to be limited by the use of constant error carousels (CECs). With the use of CECs, the conflict of weight updating is targeted. Apart from being more robust in terms of implementation, they can outcome both exploding and vanishing error problems. Hence, they have proven to bridge long-time lags within datasets better.

LSTM layers have a similar recurrent structure to Elman networks, but the data flow is transmitted for different blocks. The innovation attached to these layers is the combination of three gates called forget (Eq. 4.100), input (Eq. 4.101), and output (Eq. 4.104) that transmit the short and long terms over a memory cell (Eq. 4.103) before sending the output to the neurons through a hidden gate (Eq. 4.105). These three stages allow the network to keep or forget past components in data, merge them with recent information, and compute it to get a response. Moreover, the described memory cell is constantly uploaded as a combination of the last output (h_{t-1}) and the current input (x_t) (Eq. 4.102), indicating continued feedback that determines how relevant is the information received and whether or not include it in the model. The multiplicative gate units mentioned learn how to handle the constant error flow. Finally, the output is produced as a filtered result of the entire process combined with the cell state.

The definition of each gate is written from the equations (Eq. 4.100) to (Eq. 4.105), and the data flow is shown in Fig. 4.19:

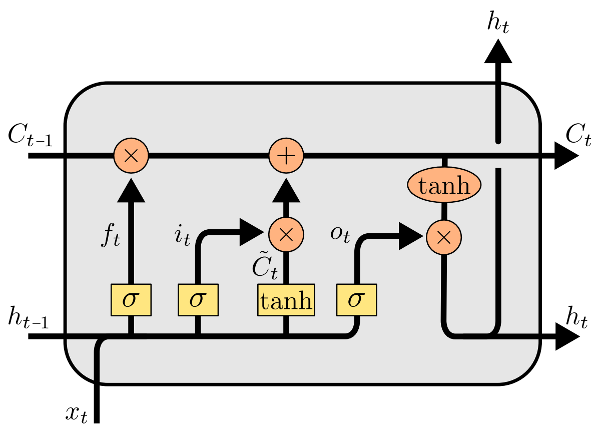


Figure 4.19: LSTM layer architecture with its workflow and different gates.
Source: Own elaboration adapted from Towards Data Science.

$$\text{Forget gate} \quad f_t = \sigma\left(W_f h_{t-1} + V_f x_t + b_f\right) \tag{4.100}$$

$$\text{Input gate} \quad i_t = \sigma\left(W_i h_{t-1} + V_i x_t + b_i\right) \tag{4.101}$$

$$\text{New memory} \quad \tilde{C}_t = \tanh\left(W_C h_{t-1} + V_C x_t + b_C\right) \tag{4.102}$$

$$\text{Memory cell} \quad C_t = f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t \tag{4.103}$$

$$\text{Output gate} \quad o_t = \sigma\left(W_o h_{t-1} + V_o x_t + b_o\right) \tag{4.104}$$

$$\text{Hidden gate} \quad h_t = o_t \otimes \tanh(\tilde{C}_t) \tag{4.105}$$

In each equation, x_t represents the sequenced input at time t, h_t is the impulse response of the LSTM layer, W and V the weights matrices, b the bias, $\otimes$ the Hadamard product and σ is the logistic function. We would like to highlight that the LSTM structure that we have presented matches its first definition (Fig. 4.19). However, multiple variants approach various fields of machine learning. It is worth mentioning that the depicted architecture refers to a single LSTM layer, but we can also combine them and get different models based on how we build them.

It is important to remember that the above-mentioned gates are not layers, despite the fact that they are responsible for fitting the weights and biases in the neural network. This innovation simultaneously avoids input weights conflicts and prevents information loss among their gates. Additionally, the different products applied open and close the access to the propagation error by keeping it constant (Hochreiter and Schmidhuber, 1997).

To conclude with general RNN blocks, there exists a manner to concatenate RNN layers via bidirectional structures. The underlying idea is to process the input forwards and backward so as to capture relevant features in both directions. They are called bidirectional RNN blocks. They are widely used in multiple applications. Most of them show better performance (Schuster and Paliwal, 1997). In Fig. 4.20, we have shown the topology of a bidirectional LSTM block.

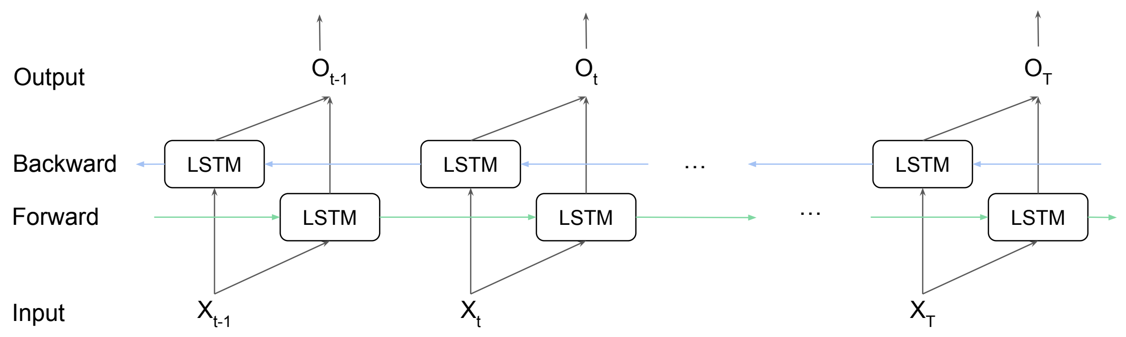


Figure 4.20: Architecture of a bidirectional LSTM block with its flow forward and backward. Source: Own elaboration.

4.3 EXPLAINABLE ARTIFICIAL INTELLIGENCE

Artificial Intelligence has achieved incredible success in many different fields of study. As we have explained during this section, the number of approaches and tasks that can be performed by means of machine learning is countless. Hence their incorporation for practical purposes is nowadays standard. We have remarked that evaluating AI models is a very important stage before their launch. However, another point to cover apart from accuracy and correct operation is the interpretation of the results.

As far as artificial intelligence is concerned, the devices and/or systems that allow their execution is supposed to interact autonomously since ML algorithms can cover the decision-making process and return the response to humans. Even though it is claimed that model explainability it is not required (Gunning et al., 2019), it is hard for a professional team to give credibility to a black box. Understanding the reasons behind the predictive analysis is crucial because trustworthiness must be a standard in the field of statistical learning (Ribeiro et al., 2016). For such reasons, transparent feedback is necessary, particularly when we let AI techniques solve problems with sensible data such as cybersecurity, defense, finance, medicine, self-driving, and so on. Nowadays, such concern is further aggravated due to the use of deep neural networks. Their implementation is widespread in multiple applications where they play a key role in regard to decision making. Nonetheless, there is a lack of knowledge and interpretation that researchers have attempted to solve.

Explainable Artificial Intelligence (XAI) is a recent field of AI composed of a set of tools, techniques, and algorithms that provides explanations, interpretation, and intuition in a human-like way to understand the functioning of ML models (Das and Rad, 2020). The main purpose of the XAI field is to convert a black box into an intelligible system whose behavior and operation are comprehensible for the final user (Gunning et al., 2019). The suitable way to proceed during the preparation of an AI system is to incorporate an additional step between the evaluation and project start-up as depicted in Fig. 4.21.

The importance of XAI techniques is rising owing to the great performance lately achieved by DNNs (Došilović et al., 2018). First, the computing systems are notably improving, allowing longer processing stages. Second, the availability of large datasets en-

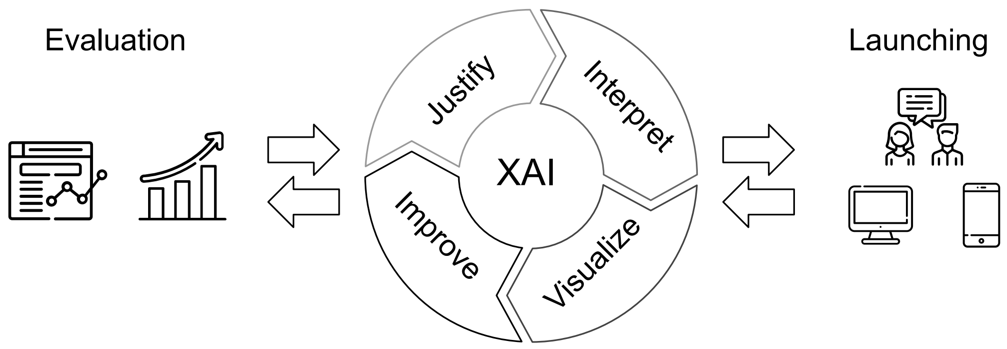


Figure 4.21: Visualization of the XAI step between evaluation and launching.
Source: Own elaboration.

ables high-level deep AI systems to perform considerably well. Consequently, it generated a framework in which human reasoning is far from understanding internal behavior.

One of the first known techniques for explainability was presented by Schetinin et al. (2007) as a seminar work motivated by the interpretation of results in clinical application. After this important contribution, we find several works approaching the XAI problem. For images, Bach et al. (2015) introduced Layer-wise Relevance Propagation (LRP), a pixel-wise decomposing technique that attributes relevance scores via backpropagation. In newer works, the Local Interpretable Model-Agnostic Explanations (LIME) of Ribeiro et al. (2016) presents an importance search via binary paths towards the output. Thus, LIME can estimate the most informative areas per each data input. Another recent contribution was the SHapley Additive exPlanations (SHAP) in Lundberg and Lee (2017). From a given input, SHAP attempts to find explainable predictions by calculating the feature contributions from the input to the output prediction. Even though multiple XAI techniques have been implemented over the years, we have just focused on introducing some of them in this thesis. For a detailed introduction to this topic, Das and Rad (2020) prepared a comprehensive summary of the different algorithms with their applications.

For the particular case of convolutional neural networks, there exists vast bibliography related to their output and XAI techniques. One of the first steps toward interpretability was the analysis of activation and saliency maps. The most representative approach is the class activation map (CAM) algorithm (Zhou et al., 2015), which highlights the discriminative zones of a given input. Although the CAM algorithm made a significant breakthrough in the field of explainability, the requirement of a global average pooling before the output layer meant a great limitation in terms of computational structure. As a manner to solve this issue, Selvaraju et al. (2017) presented Grad-CAM, an algorithm that extracts the spread of the gradients in the network graph to remark the most representative areas per each sample. By doing so, all the preceding layers play a part in the projected saliency map. Another contribution is that the gradient projection can be represented at any network depth. Intending to generalize the concept of Grad-CAM, Chattopadhay et al. (2018) defined Grad-CAM++. This new algorithm allows a better localization employing weighted incorporation of the CNN partial derivatives, thus covering bigger areas of the input object. Finally, Omeiza et al. (2019) presented a variation of Grad-CAM++ named

Smooth-Grad-CAM++. The difference is that now activation maps are generated with an addition of Gaussian noise to detect anomalous activations. The objective of Smooth-Grad-CAM++ is to identify such anomalies and generate sharper saliencies from the input space. An illustrated comparison of this family of XAI algorithms can be found in Pinciroli Vago et al. (2021), where it is applied in iconography artwork analysis.

4.4 SUMMARY

During this chapter, we have detailed how the field of Artificial Intelligence was founded, showing the motivation under this matter of study and the basic notions that define its functionality. After that, we focused our attention on the Machine Learning and Data Mining subjects, where a statistical and optimization background is given. Once we have presented the main tasks that concern this study, we have introduced three main ML algorithms: K-Nearest Neighbours, Support Vector Machines, and Decision Trees. Finally, we have presented the concepts of Deep Learning, a subfield of Machine Learning. Once we have clarified the definition of Deep Neural Network, we have introduced three families of Artificial Neural Network: Feed-forward Neural Network, Convolutional Neural Network, and Recurrent Neural Network.

Part II

EXPERIMENTAL STUDY

This part of the thesis is divided into three chapters as follows:

5 MULTIPLE-CRITERIA DECISION MAKING CASE STUDIES

5.1 MCDM-CASE 1: AN ACADEMIC PERFORMANCE INDICATOR USING FLEXIBLE MULTI-CRITERIA METHODS

This case study is based on our published research: Blasco-Blasco O, Liern-García M, López-García A, Parada-Rico SE "An Academic Performance Indicator Using Flexible Multi-Criteria Methods". *Mathematics*, 2021; 9(19):2396, DOI: 10.3390/math9192396, (Blasco-Blasco et al., 2021). Hence, all the information or results mentioned in this section have been already addressed in that paper.

5.1.1 *Experiment Setup*

Composite indicators are a very useful tool for summarizing global performance of institutions and enabling future decision-making. Ultimately, there is an increasing demand for performance measures that assess an ongoing implementation strategy. Due to the major challenges attached to such task, in this case study we address the three following points.

1. How to evaluate it at different points in time.
2. How to estimate the weighting scheme of the criteria.
3. How to normalize the data.

Our proposal is based on a recent method denominated uwTOPSIS extensively described in 3.4.2. It has been applied to data collected from 2975 students enrolled in the first year of technical careers (exact sciences and engineering), from the first semester of 2016 to the first semester of 2019 of the Industrial University of Santander of Colombia. In the section we show that our proposal makes it possible to measure and evaluate the academic performance of students at two points in time and this allows the University to know whether its student support policy has been successful and to what degree it has been effective. In regard with the computation of such amount of data, it has been managed mainly by using the version 3.8.5 of the Python programming language.

This case study is structured as follows. First, I indicate how to construct the academic performance indicator. Second, I describe the requirements to guarantee the development of a well-defined indicator. Third, I study the properties of the academic performance indicator. Finally, I present the study case applied to the Industrial University of Santander, or UIS in short for the Spanish translation of *Universidad Industrial de Santander*, with a discussion and conclusions of the work conducted.

5.1.2 *Construction of the academic performance indicator*

In this section, it will be constructed a composite indicator that allows us to relate the evaluation of the alternatives and assess performance over two time periods when only one of the criteria varies as in Parada et al. (2019). Steps will be followed to construct a composite indicator are as follows:

Step 1 Development of the conceptual framework or theoretical model, which will allow us to identify the phenomenon to be measured, the groups involved, and identify the variables.

Step 2 Selection of the decision criteria. It is essential to identify the variables or simple indicators. Determining how to obtain the data, studying whether there is missing data and how to resolve it, and analyzing the data structure is fundamental to constructing a quality.

Step 3 Comparison of alternatives. When a composite indicator is constructed, in most cases, the units in which the variables are measured and the nature of these are very diverse (Wang, 2014). To compare and aggregate variables, and simple indicators, it is necessary to express them on a similar scale, i. e. , normalize (Freudenberg, 2003). However, the normalization method must be carefully assessed because the results may vary depending on the method used. In this work, the method proposed by Liern and Pérez-Gladish (2020).

Step 4 Assignment of weights and aggregation. One of the problems, widely discussed in the literature is the weighting of variables according to their relative importance (Ouenniche et al., 2018; Saisana et al., 2005). Sometimes, the assignment is done based on expert judgment, but at other times, as we will develop in this paper, we will not assign the weights a priori but apply the uwTOPSIS method, in which it is not necessary to have the weights showing a priori the relative importance of the criteria.

Step 5 Fifth and final step. Comparison of the alternatives and sensitivity analysis of the composite indicator. In this case, the total and partial performance of each student will be evaluated using the indicator proposed in this work. When an indicator is proposed, it is necessary to test its usefulness and study whether it meets the properties of the indicators (European Commission. Joint Research Centre. and Organisation for Economic Co-operation and Development., 2008) and finally, the robustness of the indicator must be determined.

5.1.2.1 *Academic performance indicator*

Given the values of the m dimensions considered at time t, a multicriteria method is applied and the relative proximity interval is determined for each alternative, obtaining the performance range $R_i^I = [R_i^L, R_i^U]$ per each $i \in \{1, \dots, n\}$. If at time $t+1$, of all dimensions considered in the previous period, only one has varied the variables that compose it, so the score of the modified dimension is considered and the values obtained are normalized for each alternative using min-max normalization (Cables et al., 2016).

Once the score at time $t+1$ has been obtained for each alternative, it is compared with the score at time t. Thus, the results for each alternative are obtained at two points of time, which allows a priori-posteriori contrast to be carried out.

Suppose an individual has an initial score $[R^L, R^U]$ and a final score x previously normalized. When comparing x with $[R^L, R^U]$, a scalar $\lambda > 0$ would be expected to exist, so $x = R^L + \lambda R^U$, since its initial interval indicates its reference framework. However, it can be that x is outside this range, which would indicate that the individual has obtained a score lower or higher than the performance range. So the next indicator is defined.

Definition 5.1.1. *(Academic performance indicator) Given a set $\mathcal{D} = \{(x, R^L, R^U) \in [0,1]^3 : R^L \leq R^U\}$, the academic performance indicator is defined as $P_A : \mathcal{D} \to [0,1]$ such that*

$$P_A(x, R^L, R^U) = \begin{cases} 0 & \textit{if} \quad x \leq R^L \\ \dfrac{x - R^L}{R^U - R^L} & \textit{if} \quad R^L < x < R^U \\ 1 & \textit{if} \quad x \geq R^U \end{cases} \tag{5.1}$$

In the case $R^L = R^U$, the academic performance indicator is given by the characteristic function of the interval $[R^U, 1]$.

Proposition 5.1.1. *Given $0 \leq R^L < R^U \leq 1$, then per each $y \in [0,1]$ exists $x \in [0,1]$ such that $P_A(x, R^L, R^U) = y$.*

Proof. The proof is trivial by taking $x = R^L + (R^U - R^L)y$, where it exists and $0 \leq R^L + (R^U - R^L)y \leq R^L + (R^U - R^L) = R^U \leq 1$. □

From Proposition 5.1.1 it is concluded that the indicator P_A is surjective in $\mathcal{D} \backslash \Pi$, where $\Pi = \{(x, R^L, R^U) \in \mathcal{D} : R^L = R^U\}$.

5.1.2.2 *Determination of performance range values with uwTOPSIS*

To determine the performance range $R_i^I = [R_i^L, R_i^U]$ the MCDM technique uwTOPSIS is used, because unlike the TOPSIS algorithm, it considers the weights as variables. In order to avoid that data-dependence attached to the usual TOPSIS normalization, which is the $|| \cdot ||_2$ vector normalization, we have considered an alternative normalization functions that returns a rank-reversal decision making approach. As a result, the score for each alternative is given as an interval that describes the minimum and maximum possible score that could be obtained by varying the set of weights if they were calculated by applying the TOPSIS method. Then, the interval $R_i^I = [R_i^L, R_i^U]$ as our performance range, so that it represents the worst and the best possible score for each alternative $i \in \{1, \dots, n\}$. In particular, we have designed our uwTOPSIS technique with customized normalization (ϕ), usual separation function (D euclidean distance), proximity function (R) and the set of lower and upper bounds of the weights ($\Lambda = \{l_j, u_j\}_{j=1}^m$). Therefore, each R_i^I can be written as $R_i^I(X_i^0, \Lambda, \phi, D, R)$ due to the implicit function theorem, where X_i^0 is the vector with the dimensions of the i-th student a priori.

In order to generate a rank-reversal system, we have re-designed the unweighted version. First, the positive and negative ideals have been considered to be $A^+ = (1, \dots, 1)$ and $A^- = (0, \dots, 0)$ thus having a data independent comparative model. Second, the ϕ normalization used is the same than the proposed by Liern and Pérez-Gladish (2020), in which two normalization functions η and ξ are given by the following expressions:

$$\eta_{A,a,b,B;k_1,k_2}(x) = \begin{cases} \dfrac{1 - e^{k_1 \frac{x-A}{a-A}}}{1 - e^{k_1}} & \text{if} \quad A \le x < a \\ 1 & \text{if} \quad a < x < b \\ \dfrac{1 - e^{k_2 \frac{B-x}{B-b}}}{1 - e^{k_2}} & \text{if} \quad b < x \le B \\ 0 & \quad \text{otherwise} \end{cases} \tag{5.2}$$

$$\xi_{A,a,b,B}(x) = \begin{cases} \dfrac{x - A}{a - A} & \text{if} \quad A \le x < a \\ 1 & \text{if} \quad a < x < b \\ \dfrac{B - x}{B - b} & \text{if} \quad b < x \le B \\ 0 & \quad \text{otherwise} \end{cases} \tag{5.3}$$

In both normalization functions, the (A, a, b, B) array represents the trapezoidal fuzzy shape and the (k_1, k_2) coefficients are the left and right exponents that determines whether each the spread is convex or concave. For a better understanding of their functionality, Fig. 5.1 depicts how η and ξ transform the data.

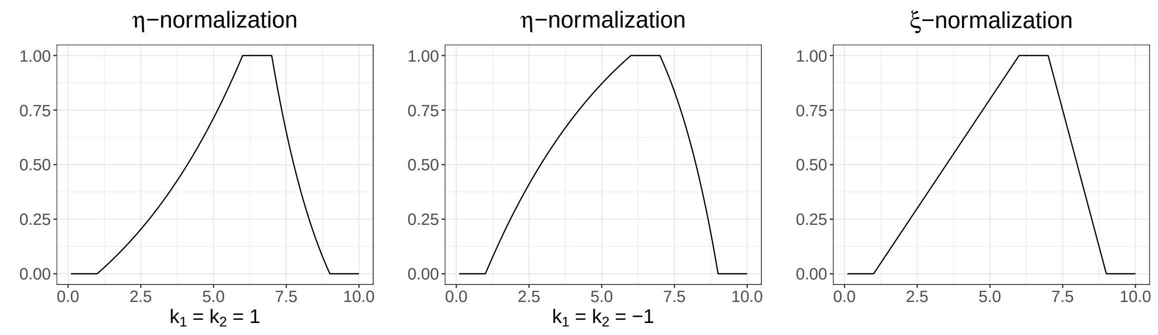


Figure 5.1: Particular case of the normalizations η and μ in which the values $(A, a, b, B) = (1, 6, 7, 9)$ have been set. The first graph is the normalization η with the values $k_1 = k_2 = 1$, the second is the normalization η with $k_1 = k_2 = -1$ and the third is the normalization ξ.
Source: Blasco-Blasco et al. (2021).

5.1.2.3 *Properties of academic performance indicator*

When constructing an indicator, it is necessary to make sure that it verifies properties that guarantee its usefulness (Ivanova et al., 1999; Parada et al., 2019; Zheng, 1993). In the following, the properties that verifies P_A will be checked.

1. *Existence and determination.* For every tern $(x, R^L, R^U) \in \mathcal{D}$ the function P_A is well-defined and the value of $P_A(x, R^L, R^U)$ exists due to the proposition 5.1.1.

2. *Uniqueness.* By the indicator's construction, each element of its domain returns a unique value.

3. *Monotonicity.* Given $(x, R^L, R^U) \in \mathcal{D}$, for the cases where $x \leq R^L$ and $x \geq R^U$ the indicator is monotonically being a constant. For the case that $R^L < x < R^U$, the gradient of the indicator can be defined as

$$\nabla P_A(x, R^L, R^U) = \left(\frac{1}{R^U - R^L}, -\frac{x - R^L}{(R^U - R^L)^2}, \frac{x - R^U}{(R^U - R^L)^2} \right).$$

   Therefore, the function is monotonic increasing with respect to x and monotonic decreasing with respect to R^L and R^U.

   Fig. 5.2 shows a particular case of the behavior of the excellence indicator. Three graphs are represented where one of the variables varies and the rest remain fixed.

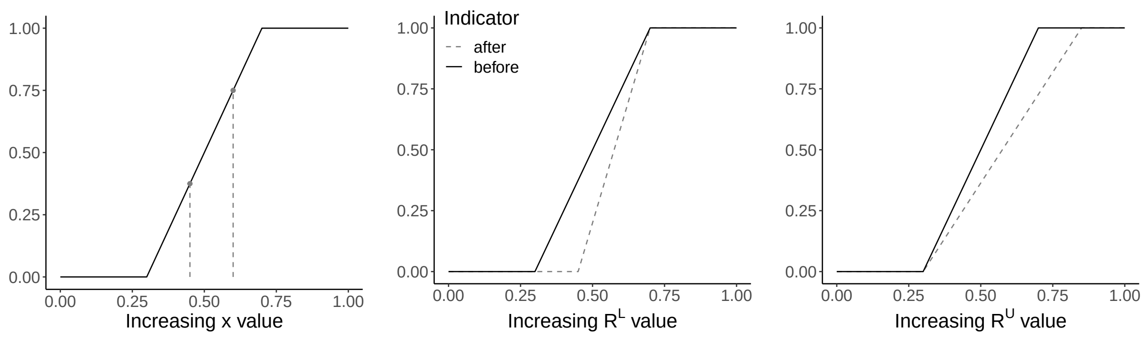


Figure 5.2: Special case of the behavior of the academic performance indicator P_A by increasing the value of each of its variables 0.15.

Source: Blasco-Blasco et al. (2021).

4. *Continuity.* The indicator is continuous except for the case where $R^L = R^U$. This is because, for each $(x, R^L, R^U) \in \mathcal{D}$, where $R^L < R^U$ is satisfied

$$\lim_{x \to (R^L)^+} P_A(x, R^L, R^U) = 0 \text{ and } \lim_{x \to (R^U)^-} P_A(x, R^L, R^U) = 1.$$

5. *Decomposability.* Given $\mathcal{D}_1, \mathcal{D}_2 \subset \mathcal{D}$ such that $\mathcal{D}_1 \cup \mathcal{D}_2 = \mathcal{D}$ and $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$, it is easy to see $P_A = P_A|_{\mathcal{D}_1} + P_A|_{\mathcal{D}_2}$. For the case that $R^L = R^U$ the proof is trivial. Now if in each $\mathcal{D}_1$ and $\mathcal{D}_2$ is satisfied $R^L < x < R^U$, we have the same decomposition since the function has no discontinuities in $\mathcal{D}$ as is well demonstrated by the property 4.

6. *Normality.* By definition, the indicator P_A takes values between 0 and 1, where in addition the same scale is guaranteed for each of the variables in $\mathcal{D}$.

7. *Scale invariance.* Given an element $(x, R^L, R^U) \in \mathcal{D}$ and a scalar $\lambda \in \mathbb{R}$ such that $(\lambda x, \lambda R^L, \lambda R^U) \in \mathcal{D}$, so that $P_A(\lambda x, \lambda R^L, \lambda R^U) = \frac{\lambda x - \lambda R^L}{\lambda R^U - \lambda R^L} = \frac{x - R^L}{R^U - R^L} = P_A(x, R^L, R^U)$.

8. *Translation invariance.* Given an element $(x, R^L, R^U) \in \mathcal{D}$ and a scalar $\lambda \in \mathbb{R}$ such that $(x + \lambda, R^L + \lambda, R^U + \lambda) \in \mathcal{D}$, so that $P_A(x + \lambda, R^L + \lambda, R^U + \lambda) = \frac{(x+\lambda)-(R^L+\lambda)}{(R^U+\lambda)-(R^L+\lambda)} = \frac{x - R^L}{R^U - R^L} = P_A(x, R^L, R^U)$. $\square$

Having verified that the academic performance indicator obeys the stated properties and in order to contrast the a priori and a posteriori results, the academic performance indicator for the set of ternaries $\{(X_i^1, R_i^L(X_i^0), R_i^U(X_i^0))\}_{1 \le i \le n}$ will be applied of each alternative. Where, X_i^0 and X_i^1 are the variables of the i-th alternative at time t and $t+1$ respectively.

5.1.3 *Application of the academic performance indicator to UIS students*

In this section, the academic performance indicator will be applied to data from the Industrial University of Santander, Bucaramanga campus, Colombia. For this purpose, we have obtained information from science and engineering students enrolled in the periods from the first semester of the academic year 2016 to the first semester of 2019.

First, a global study of the results obtained by applying the academic performance indicator have been carried out, thus determining the performance of students in their first semester at university. Subsequently, the results obtained are shown when grouped by gender and economic status. Finally, to analyze student performance using the academic performance indicator, a threshold of $\alpha = 0.6.$ is set. It is considered that if a student has a P_A score equal to or higher than 0.6, he/she presents an adequate academic level and meets the minimum required by the university.

5.1.3.1 *Dataset*

For this work, from the data provided by the SEA, 2975 students enrolled in the first year of science and engineering are selected and the data of the academic, cognitive, economic, health and social dimensions of students who have just joined the university have been obtained. The variables that compose each dimension are:

Academic dimension (A) This dimension is composed of three items: a diagnostic test of UIS Math (DTM); EFAI-4 numerical ability (NUA); and, 11-Math Knowledge Test (PSO).

Cognitive dimension (C) Five items are assessed: verbal reasoning (VR), numerical reasoning (NR), abstract reasoning (ABR), memory (MEM) and spatial attitude (SA).

Economic dimension (E) The variables that analyze this dimension are: Income from economic dependence, ED = wage/SMMLV (where SMMLV = current legal minimum monthly wages), the number of siblings (NS), the position between siblings (PS) and the payment of rent during the course (PRC).

Health dimension (H) The eight variables that compose this dimension are: Anxiety (ANX), Depression (DEP), Emotional Adjustment (EMA), Alcohol Dependence (ALD), Psychoactive Substance Abuse (PSA), Chronic Illnesses (CI), Disability (DI) and "Question 23" (P23), which refers to the tendency toward suicide.

Social dimension (S) This is determined by family dysfunction (FAD), through the "Family APGAR" (Smilkstein, 1978).

As the dimensions are measured on different scales (Table 5.1) the transformation proposed by Liern et al. (2020b) has been used, which allows us to work with data grouped by intervals and combine different types of variables.

Table 5.1: Dimensions, ranges, ideals and normalization functions in the UIS dataset.

Dimension	Original	Transf.	Ideal	Normalization
Academic	{VL, L, LM, M, MH, H, VH}	$[1,7]$	$[6,7]$	$\eta_{1,6,7,7;1,0}(x)$
Cognitive	{VL, L, LM, M, MH, H, VH}	$[1,7]$	$[6,7]$	$\eta_{1,6,7,7;-1,0}(x)$
Economic	$[0,1]$	$[0,1]$	$[0.8,1]$	$\xi_{0,0.8,1,1}(x)$
Health	$[0,0.65]$	$[0,0.65]$	0.65	$\xi_{0,0.65,0.65,0.65}(x)$
Social	$\{0.1,0.5,0.7,1\}$	$[0.1,1]$	$[0.7,1]$	$\xi_{0.1,0.7,1,1}(x)$

At the end of the semester, to study the variation in academic performance, the grades of each student in the subjects of calculus and algebra are considered and not the university entrance grades, which were considered in the academic dimension before the beginning of the semester. Since the grades are measured differently from the other dimensions, they will be normalized using the min-max normalization and calculate the mean.

Performance on admission to college will be determined by the interval $[R^L, R^U]$, which represents the worst and best possible score for each student on entering university and is the result of aggregating the five dimensions. The uwTOPSIS method will be applied to obtain the interval. At the end of the semester, the value of the academic dimension will be calculated, which will be given by the X^1 value, normalized with the minimum-maximum normalization. With the data obtained before and after the end of the first semester, the students will be classified depending on whether or not the academic dimension score is contained within the interval. The student can be considered to have had an expected performance if the final score is contained in the interval $[R^L, R^U]$, excellent if he/she has obtained a final score higher than R^U and, on the contrary, insufficient score if he/she has had a final score lower than R^L. From now on, the academic performance of students are distinguished as shown in Table 5.2.

Table 5.2: Type of academic performance from P_A.

	Indicator $P_A(X^1, R^L, R^U)$
Excellence	$P_A = 1$
Expected	$0 < P_A < 1$
Insufficient	$P_A = 0$

In the case study, the dimensions as criteria have been considered to be maximized. This will cause students with worse conditions to obtain a small value in R^L and, on the contrary, those with stable conditions will have a value close to 1 in R^U. We have considered the positive and negative ideals to be $A^+ = (1, \ldots, 1)$ and $A^- = (0, \ldots, 0)$ respectively. Finally, to prevent the method from eliminating certain criteria when optimizing, we have considered the minimum and maximum weights to be, respectively, 0.1 and 0.5.

Due to the large amount of data handled, both the data management and the model implementation has been done with Python (Van Rossum and Drake Jr, 1995). In particular, we have made use of our GitHub repository designed to apply the uwTOPSIS method (López-García, 2021a). For implementation purposes, 5.1.6 contains the Script 3, which is the Python implementation with both η and ξ normalization functions.

Remark. *The mathematical optimization associated to the problem has been computed with the optimize module of the SciPy library (Virtanen et al., 2020). We would like to highlight that this is not an unique option, although for our particular case it was enough. In cases when we face more complex problems, it would be convenient to develop a metaheuristic algorithm that fits with the behaviour of the objective function.*

5.1.3.2 *Classification of students at UIS Colombia according to gender and economic status.*

Taking into account the interval values obtained with the uwTOPSIS method, the performance of students in each semester of the academic years 2016 to 2019 was studied. The results obtained by applying the proposed academic performance indicator (5.1) on the data set offered by the SEA, show us that on average the academic performance is 0.6498 or higher in all the periods analyzed, except in those corresponding to the two semesters of 2018 that take values of 0.2177 and 0.2795 (Table 5.3).

Table 5.3: Average of the a priori, posteriori and academic performance indicator results by semester.

Course	R^L	R^U	X^1	P_A
2016_1	0.3718	0.8540	0.8346	0.8602
2016_2	0.4081	0.8533	0.7913	0.7734
2017_1	0.4225	0.8671	0.8172	0.7814
2017_2	0.5192	0.8879	0.7733	0.6498
2018_1	0.5753	0.8988	0.5868	0.2177
2018_2	0.5080	0.8835	0.5723	0.2795
2019_1	0.5334	0.8868	0.8299	0.7560

Thus, for example, while in the semester of 2016_1, on average they had an academic performance of 0.8602 or in 2017_1 the value of the indicator is 0.7814, in 2017_2 the value is reduced to 0.6498 due to the number of students who had low scores. In addition, in the two periods of 2018 there was an atypical behavior due to the fact that the students' scores when considering all dimensions were not good and also the width of the interquartile range in these periods was larger (Fig. 5.3). One possible explanation is the national university strike in Colombia in 2018. That strike forced the institution to delay classes. Some students had to cancel the academic semester and a majority failed many subjects, among them calculus and linear algebra.

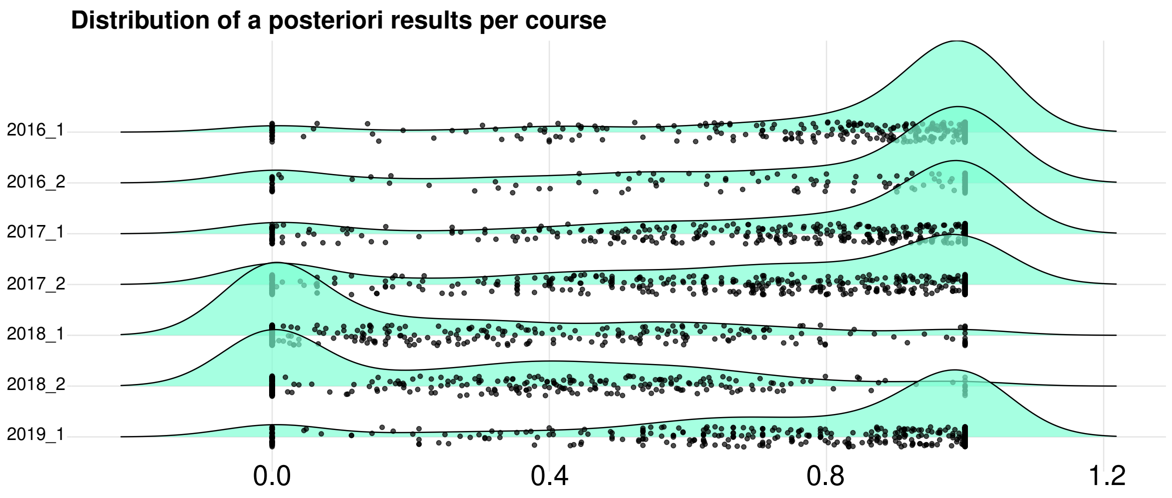


Figure 5.3: Distribution of students by academic semester.
Source: Blasco-Blasco et al. (2021).

If the academic performance according to gender for all periods is analyzed, information on 2040 male and 935 female science and engineering students is considered. The average academic performance for males is 0.6211 and for females 0.6632, therefore, it is seen that the values are similar when disaggregated by gender. (Table 5.4).

Table 5.4: Academic performance indicator P_A values broken down by gender.

Group	P_A	Var	N
M	0.6211	0.1555	2040
F	0.6632	0.1380	935

When breaking down the mean values by gender, for each semester, it can be seen that the highest scores occur in semester 2016_1, where the mean for men is 0.8571 and for women is 0.8667. The largest differences occur in semester 2017_2, where the mean for men is 0.6149 while for women in this same period is 0.7195. In this case, it can be seen that the variance in the men's group for this period is the highest with 0.1454. More detail can be seen in Table 5.5. In Fig. 5.4, it can be seen that there are no significant differences in the academic performance score by gender for the different periods studied. Therefore, it can be said that academic performance is not affected by gender for these semesters.

Table 5.5: Mean and variation of the academic performance indicator broken down by semester and gender.

Course	Group	Mean	Var
2016_1	M	0.8571	0.0681
	F	0.8667	0.0667
2016_2	M	0.7623	0.1175
	F	0.7949	0.0919
2017_1	M	0.7708	0.1012
	F	0.8046	0.0854
2017_2	M	0.6149	0.1454
	F	0.7195	0.0974
2018_1	M	0.2125	0.0839
	F	0.2297	0.0927
2018_2	M	0.2661	0.0856
	F	0.3075	0.0796
2019_1	M	0.7452	0.1095
	F	0.7848	0.0786

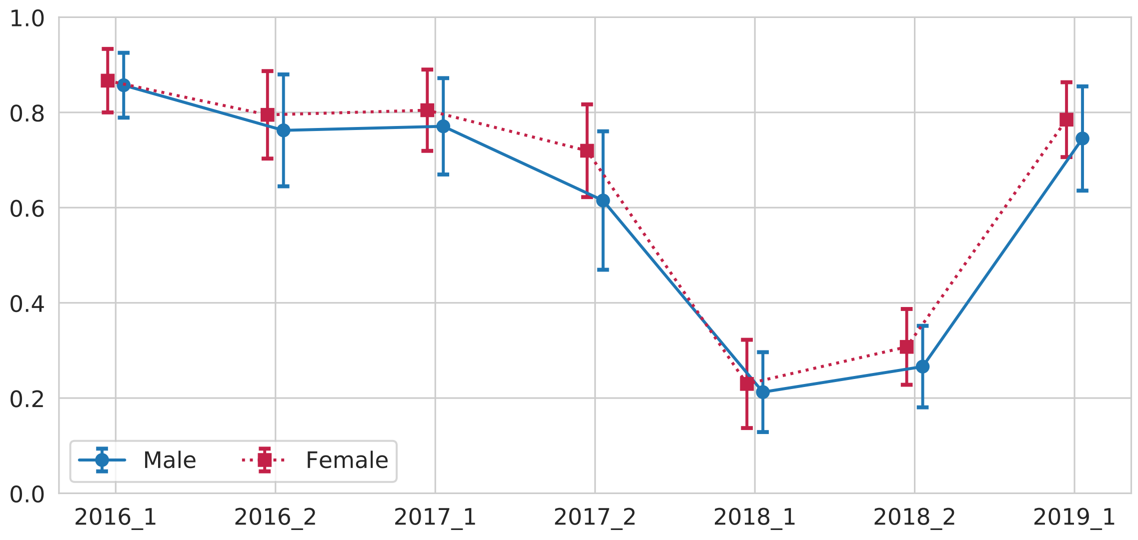


Figure 5.4: Value of the academic performance indicator by gender.
Source: Own elaboration.

Next, it is intended to study how the economic condition of students who access the UIS affects academic performance. The institution classifies students according to whether their economic risk is low, medium or high. However, for the periods analyzed, students are either low risk (ER_l) or high risk (ER_h). Thus, a total of 458 students are at low economic risk while a total of 2517 are at high economic risk, which is 84.6% (Table 5.6).

If we group by economic status and gender, it can be seen that 85.15% of men and 83.42% of women belong to the high economic risk group (Table 5.6). Therefore, most of the students accessing UIS have a hard economic situation. It is worth mentioning that the UIS is one of the most important official universities in the region and due to its public institution it serves students from the lowest socioeconomic strata, which is reflected in the data.

Table 5.6: Students by risk group and gender.

	Number of students		**Percentage of students**	
Gender	ER_l	ER_h	ER_l	ER_h
M	303	1737	14.85	85.15
F	155	780	16.58	83.42

If Table 5.7 is observed , where the average performance values per semester are shown, it can be seen that in all periods, except in 2016_1 the scores are higher for students with low economic risk. If in the 2016_2 and 2017_1 periods the scores between the two groups do not differ by more than 0.06 points. From this period onwards, the differences are significant, reaching the maximum difference in 2018_1 as reflected in Fig. 5.5.

Table 5.7: Academic performance indicator values broken down by semester per economic condition.

Course	Group	P_A	Var
2016_1	ER_l	0.8411	0.1000
	ER_h	0.8629	0.0631
2016_2	ER_l	0.8276	0.0855
	ER_h	0.7663	0.1116
2017_1	ER_l	0.8130	0.0748
	ER_h	0.7792	0.0979
2017_2	ER_l	0.7525	0.0767
	ER_h	0.6270	0.1412
2018_1	ER_l	0.4747	0.0739
	ER_h	0.1568	0.0702
2018_2	ER_l	0.4968	0.0399
	ER_h	0.2200	0.0795
2019_1	ER_l	0.9110	0.0360
	ER_h	0.7174	0.1101

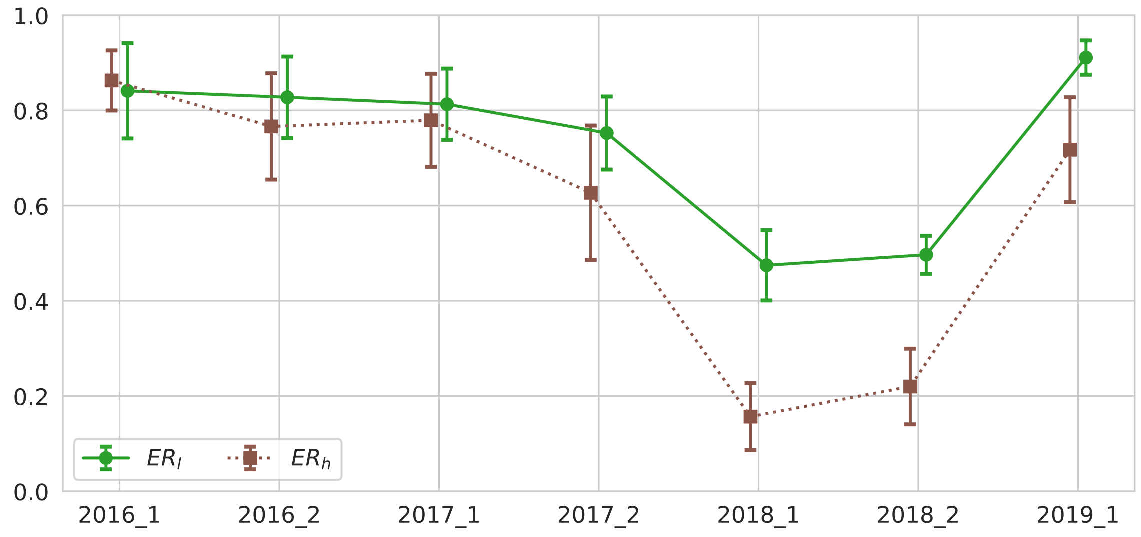


Figure 5.5: Value of the academic performance indicator by economic risk.
Source: Own elaboration.

The intention is to compare the results of academic performance with P_A taking into account whether they have received actions or not. If Table 5.8 is observed, it can be seen that in all semesters except in 2019_1 the mean value of the P_A indicator is higher for students who have not received complementary actions, i. e. those who had a high score in the academic dimension when they accessed the university. It can also be highlighted the differences that exist between the two groups of students in semester 2016_2, where the average academic performance for those who did not participate in any action was 0.9080 and those who participated in some action was 0.6924. In 2018_2, the mean academic performance for those who did not participate in any action was 0.6229 and those who participated in any action was 0.2654. This may be due to the fact that in some semesters the political situation in the country led to class stoppages during part of the semester and the suspension of complementary actions. In this way, students with greater needs ceased to have some support actions.

Table 5.8: Academic performance indicator by semester and participation.

	No Participation		**Participation**	
Course	P_A	**Var**	P_A	**Var**
2016_1	0.8977	0.0488	0.7982	0.0927
2016_2	0.9080	0.0386	0.6924	0.1336
2017_1	0.8004	0.0950	0.7641	0.0972
2017_2	0.6509	0.1352	0.6492	0.1297
2018_1	0.2030	0.1262	0.2182	0.0855
2018_2	0.6229	0.1744	0.2654	0.0756
2019_1	0.7020	0.1494	0.7577	0.0999

The percentages of students appearing in each group (Excellence, Fail or Expect) show disparate results depending on the semester analyzed, but in all of them students who do not participate in actions present a higher percentage of Excellence (Table 5.9). For example, in 2016_2, the percentage of Excellence among students who did not participate in any action was 64.62%, while that of those who participated in at least one was 39.81%. Surprising again, the results obtained for the 2018_1 and 2018_2 semesters by the few students who have an Excellence score, the 8.33% and 38.46% of those who do not participate in complementary actions and the 3% and 1.26% of those who participate, but above all, the percentage of students in the Fail situation, 41.67% and 23.08% in the group of those who do not participate and 49.50% and 40.69% of those who have participated in some complementary action, is surprising in these periods.

Table 5.9: Percentage of students according to whether or not they have participated in the support program.

	Percentage of students				
	Course	**Excellence**	**Fail**	**Expected**	**$\alpha \geq 0.6$**
No Participation	2016_1	62.62	2.49	34.89	90.97
	2016_2	64.62	1.54	33.85	90.77
	2017_1	49.62	6.77	43.61	79.32
	2017_2	28.43	14.21	57.36	62.94
	2018_1	8.33	41.67	50.00	16.67
	2018_2	38.46	23.08	38.46	61.54
	2019_1	46.67	13.33	40.00	66.67
Participation	2016_1	40.21	7.22	52.58	81.44
	2016_2	39.81	12.04	48.15	67.59
	2017_1	39.38	6.16	54.45	76.37
	2017_2	27.51	13.59	58.90	64.08
	2018_1	3.00	49.50	47.50	13.00
	2018_2	1.26	40.69	58.04	13.56
	2019_1	38.84	8.37	52.79	77.25

Next, the data of the students who have obtained a value of $P_A \geq 0.6$ will be analyzed. According to the UIS criteria, we consider that from this value, they present an adequate academic performance. In all periods the percentage was higher than 61%, except in 2018_1, where the percentage of those who did not receive complementary actions was 16.67% and of those who received was 13% and in 2018_2, where the percentage of students who received complementary actions was 13.56%. If the results are compared for each semester, it is seen that in all periods the percentages are higher for students who do not receive complementary actions, except in 2019_1, where students who do not participate account for 66.67% and those who participate account for 77.25%. In any case, what can be observed is that the complementary actions allow a high percentage of students to be above the proposed threshold. Obviously, our intention goes beyond comparing groups. The most important thing is to prevent students in vulnerable situations from being unable to continue university studies. With these actions, a high percentage of students achieve the expected results and reduce the dropout rate, which is a success for the institution.

If the groups by gender are analyzed (Table 5.10), it can be seen that the results are not very different between men and women. Both in the groups that have not participated in complementary actions and in those that have participated in at least one, women obtain better results. The percentage of women with an Excellence score among those who do not participate in any action is 56.98% while that of men is 47.07%. The same is true for those who participate in any complementary actions. The percentage of women is 26.14% while that of men is 24.20%. If we focus on the percentage of students by gender in the case of $\alpha \geq 0.6$, we see that the pattern is repeated. In the case of females who do not participate in any action, the percentage is 86.43%, compared to 76.55% for males. The same is true for students participating in complementary actions (55.54% of females have a value of P_A higher than 0.6, while the percentage of males is 51.88%).

Table 5.10: Gender of the students that have not participated in the support program (NP) and the students that have participated, at least once, in the support program (P).

		Percentage of students			
	Group	**Excellence**	**Fail**	**Expected**	$\alpha \geq 0.6$
NP	M	47.07	8.24	44.69	76.55
	F	56.98	5.04	37.98	86.43
P	M	24.20	23.70	52.09	51.88
	F	26.14	17.58	56.28	55.54

Regarding the study of the economic conditions of the students, a higher percentage of excellent students is observed in the groups that have not participated in the SEA support program (Table 5.11). Thus, 55.29% of students who have not participated in complementary actions and who have a low economic risk are excellent versus 32.98% of those who have participated in some action. Among those with high economic risk, ER_h, of those who have not participated in complementary actions, 49.38% are excellent versus 23.06% who have participated in at least one action. In any case, if it is considered that the students who have participated in complementary actions, 61.93% of those with a low economic risk, ER_l and 51.14% of those with a high economic risk, ER_h have obtained an academic performance indicator score, P_A above 0.6. Given that more than half of the students who have participated in the complementary actions organized by the SEA, have had a score higher than the threshold set as success of the strategy, it is justified to recommend to the UIS to continue with the actions carried out and implement some more so that the success rate is close to that of the students who do not have support needs.

Table 5.11: Economic risk of the students that have not participated in the support program (NP) and the students that have participated, at least once, in the support program (P).

		Percentage of students			
		Excellence	**Fail**	**Expected**	$\alpha \geq 0.6$
NP	ER_l	55.29	1.18	43.53	84.71
	ER_h	49.38	7.96	42.66	78.86
P	ER_l	32.98	3.49	63.54	61.93
	ER_h	23.06	25.69	51.26	51.14

5.1.4 *Discussion*

We do not want to finish this work without including a discussion about the construction and use of indicators and the application of a MCDM method, the uwTOPSIS, where it is not necessary to assign the weights a priori.

An indicator of academic performance has been constructed which lets see the evolution of the students in two moments of time, comparing the final value with the initial score obtained using the uwTOPSIS multi-criteria method. For alternative, a lower bound and an upper bound are considered, which allows constructing an interval $R_i = [R_i^L, R_i^U]$ that represents the worst and the best possible score. In addition, comparing the information obtained by applying the uwTOPSIS at a time t, with the data obtained at time $t+1$ for the same dimension but considering different variables, we can construct an indicator of academic performance that determines whether the performance of a student has worsened, maintained, or improved.

The main use of the uwTOPSIS method is that it considers the weights as variables, and allows the evaluation of the alternatives without a prior assignment of the weights. It means an appreciable improvement since in previous works using data from the Universidad Industrial de Santander, allocation of weights was made based on expert judgment. This sometimes implied difficulties in reaching a consensus, and consequently, studies were not fully implemented in the strategy. Undoubtedly, uwTOPSIS allows progressing in improving of the procedures we have applied since it has several advantages:

- An interval associated with the worst and the best possible result of each student is obtained, which allows knowing their aspirations.
- It has been avoided that the results depend on the data handled (*rank reversal* Cables et al., 2016), using $(1, 1, \ldots, 1)$ and $(0, 0, \ldots, 0)$ as ideal and antiideal respectively.
- A standardization has been applied that allows taking into account the criteria of the institution, which makes easier the future applicability of the results.

As happens in all studies that are used in real problems, with a large number of data and involving several decision-makers, our proposal does not resolve some issues:

- In this study, stability/regularity in the data has not been taken into account. As it was said in the different sections, there are periods with clear differences with respect to the others, and this could require an in-depth study of the homogeneity between periods. However, once the results are known, it is clear that these differences should be taken into account in a subsequent study.
- The intervals assigned to the weights, although they make the results more flexible and objective, determine (in some way) the groupings. This could be avoided by allowing weights that could span the entire interval $[0, 1]$. However, this would not make complete sense in a multi-criteria environment. According to Ishizaka and Labib (2011) a weight greater than 50% in any of the criteria in a decision context with more than two criteria, it is not sustainable with the multi-criteria character.

The choice of the TOPSIS method is based on the conclusions of Liern et al. (2020b), since it was proved that this technique fitted appropriately with the SEA-UIS dataset. Moreover, the approach of unweigthed MCDA techniques is a field in which we must research in depth.

We also consider the use of machine learning algorithms for future works, in which we could implement supervised learning techniques to estimate future academic performance or unsupervised learning techniques for cluster analysis. Nonetheless, in this paper we have not applied such techniques because we have followed the UIS guidelines in terms of the data preparation and both evaluation and classification of their students.

Finally, it is pointed out the difficulties involved in applying some methods in practice due to the associated computational cost of solving uwTOPSIS, when calculating uwTOPSIS for 2975 alternatives and 5 criteria, which means a decision matrix with 14875 entries. When calculating the optimal solutions of the relative proximity function (R), a total of 5950 mathematical optimization problems are performed to obtain each $[R^L, R^U]$. With all this, our code (López-García, 2021a) has been compiled on a computer with AMD Ryzen 7 3700U with Radeon Vega Mobile Gfx processor and 16GB of RAM, obtaining a computational cost of 4651.98 seconds (1h, 17′ and 31.98″) to run the code in full, which is 1.27 iterations per second. However, when adding the Python Joblib module to execute parallel computation, the computational cost was 1629.83 seconds (27′ and 9.83″) when using 8 CPUs simultaneously, which is 3.65 iterations per second.

5.1.5 *Conclusions*

By constructing composite indicators that incorporate all the variables and meet a series of indicator properties, it is possible to synthesize the information and draw simple and rapid conclusions that help institutional decision-makers. However, the proper construction of indicators requires overcoming or resolving several obstacles. Once the standardization problems have been solved, by applying the proposal made by Liern et al. (2020b), one of the pending issues is the assignment of weights. The objective of this work is to propose a composite indicator in which the weights are considered variable. Taking these aspects into account, the proposed indicator, called *Academic Performance Indicator*, allows academic institutions to obtain the value of the indicator for each student and to know whether or not there has been an improvement in the results.

The use of the proposed indicator allows the institution to know whether the complementary actions implemented have been successful or not, since at the beginning of the semester, based on the available data of the five dimensions analyzed, it is determined which students can participate in the actions promoted by the institution. At the same time, global data on the students is provided, and groups can be established according to the result of the academic performance calculated from the indicator.

The proposed method was applied to the data obtained from the students of the Industrial University of Santander (Colombia) to draw conclusions that can be applied to decision making. The UIS committed, investing economic and human resources, to offer complementary actions and wants to determine whether this decision is adequate or needs to be modified. One way to determine this is by calculating the academic performance indicator, so that if it improves or is maintained, the investment is adequate; if not, the university will have to reinforce the proposed activities, determine other alternatives or even change the policy carried out so far.

By applying the method to the UIS data, it is verified that there are no significant differences in academic performance when calculating the indicator according to gender; however, these differences are greater if the economic condition of the students is taken into account. However, by applying complementary actions, the institution guarantees an improvement in the results of students who presented deficiencies at the beginning of the semester, improving academic results and reducing the dropout rate.

This work is expected to continue deepening in the construction of indicators that allow the UIS to have a global vision of the results of the policies implemented in 2014, the year in which the SEA is founded and actions to help students with academic and economic problems began to be implemented. It is also intended that the use of indicators will allow comparison with other universities and facilitate other universities or local governments in the design of student curricula and the implementation of educational policies.

5.1.6 *Python implementation*

The script that yields the results before mentioned through the uwTOPSIS implementation is presented in § A.2.

5.2 MCDM-CASE 2: UWVIKOR: AN UNWEIGHTED MULTI-CRITERIA DECISION MAKING APPROACH FOR COMPROMISE SOLUTION

5.2.1 *Experiment Setup*

Decision-making requires objective frameworks that satisfy the needs of the stakeholders implicated. The more flexible the algorithm, the more balance the results will be. However, rankings not only have to sort elements but offer guarantees that the best alternative is as satisfactory as required. Despite having classic outranking methods that help us to arrange a set of alternatives, they require the use of weights. The definition of the weighting scheme is a critical step in any multi-criteria decision-making method, which can heavily affect the final result. As a solution to this problem, we extend a methodology, previously used with the TOPSIS algorithm, and introduce an unweighted version of the VIKOR method. We show that this new method offers flexible tools to decision-makers for dealing with compromise solutions. We illustrate the new VIKOR method with three study cases, the first one is a comparison with the classical VIKOR method for a problem of locating hydropower plants, the second is a comparison with the current state-of-the-art uwTOPSIS method, and the third one is a direct application to the selection of the Most Valuable Player (MVP) of the NBA league, where we compare our results with the actual rankings and with other MCDM methodologies.

5.2.2 *Introduction*

The field of multi-criteria decision-making helps us to select, from among several alternatives, which is the best, according to different pre-established criteria that may conflict with each other. Moreover, these criteria do not necessarily have to be of equal importance. Therefore, one of the steps that every multi-criteria decision-making (MCDM) method has is the weighting of the criteria. This is a critical step, since, on many occasions, a small change in these weights can cause a substantial change in the final decision choice. A widespread option for determining weights is the use of expert advice. Here, following a predefined methodology, the help of a panel of experts in the field, who define the preferences of the criteria, is used. This can be achieved either by consensus (Delphi Method, Sackman, 1974), or by pair-wise comparisons (AHP, Saaty and Vargas, 2012). However, one of the most important criticisms that these methods based on expert panels raise is that they inevitably introduce subjective opinions, which can lead to biases in the process.

Thus, finding methods that allow the determination of criteria weights as objectively as possible is a topic that has attracted the interest of many researchers in the field of multi-criteria analysis, for example, the Entropy method (Shannon, 1948), the LINMAP method (Srinivasan and Shocker, 1973), or more recently, the IDOCRIW method (Zavadskas and Podvezko, 2016), the Bayesian approach (Vinogradova et al., 2018), or the FUCOM method (Pamucar et al., 2018).

Notwithstanding this, a new approach to the problem of weight determination has been proposed for the TOPSIS method by Liern and Pérez-Gladish, 2020 and by Benítez and Liern, 2021. This new approach focuses on completely eliminating the precise definition of the criteria weights in the problem, allowing the decision-maker to state only some sensible bounds for such weights, resulting in an unweighted TOPSIS method.

The aim of this paper is to analyze how this unweighted methodology presented in Liern and Pérez-Gladish (2020) can be extended to another MCDM method such as the VIKOR outranking algorithm (Opricovic, 1998), used for finding compromise solutions. We will also illustrate this new methodology with three applications: in the first one, we compare the newly-introduced uwVIKOR with the classical VIKOR for a problem of locating a hydropower system studied by Opricovic and Tzeng (2007), the second one is another classical problem of Jacquet-Lagreze and Siskos (1982) in which we compare uwVIKOR with the results obtained in Benítez and Liern (2021) with the uwTOPSIS method, and the third case study is an application of the method to the determination of the Most Valued Player (MVP) of the NBA Basketball League from 2001 to 2020.

The aim of this paper is to analyze how the unweighted methodology presented in Liern and Pérez-Gladish (2020) can be used for finding compromise solutions by means of the VIKOR outranking algorithm. We will also illustrate this new methodology with two applications: the first one is a classical problem of Jacquet-Lagreze and Siskos (1982), and the second one is an application of the method to the determination of the Most Valued Player (MVP) of the NBA Basketball League from 2001 to 2020.

5.2.3 *Literature review*

In the 1990s, Serafim Opricovic presented the *VIseKriterijumska Optimizacija I Kompromisno Resenje* (VIKOR) method, in English translated as Multicriteria Optimization And Compromise Solution (Opricovic, 1998). It was introduced as an MCDM outranking strategy to rank alternatives based on the search for a feasible compromise solution.

From then on, many variations of the method with multiple perspectives have arisen (See Mardani et al. (2016), and references therein). For instance, Opricovic and Tzeng (2002) and Opricovic and Tzeng (2003a) introduced crisp sets to solve conflicting cases of fuzzy approaches for defuzzification methods, later Opricovic and Tzeng (2003b) added strategies with incomplete information. A scheme for compromise solutions under fuzzy logic was written in Opricovic (2007). For membership of the weights, Devi (2011) added triangular fuzzy numbers to give the intuition of feasible importance. We can also find papers that extend the VIKOR problem to interval numbers (Sayadi et al., 2009). Finally, there are combinations of the VIKOR methodology with other fields such as gray relational analysis (Kuo and Liang, 2011) and group decision making (Park et al., 2011).

A fundamental problem in multi-criteria decision analysis methods is the determination of the relative importance of the various criteria (Triantaphyllou, 2000). Many methods require a quantitative definition of the weight of each criterion, and such a definition is often difficult. Sometimes the definition of the weighting itself is not precise or the values themselves are given with a certain degree of uncertainty. Although weights sometimes have no meaning beyond relative preference or importance, their value can significantly affect the outcome of the decision-making.

The first approach is to consider, given an initial set of weights, the ranges within which these weights can vary without changing the result of the ranking of alternatives. Such stability intervals were first considered by Mareschal (1988), and Wolters and Mareschal, 1995 for additive and PROMETHEE methods. Later, Opricovic and Tzeng, 2007 defined the stability intervals for the VIKOR method. However, the stability intervals simply determine how to vary the weights so that the ranking of alternatives does not change with respect to a given ranking with initially indicated weights. Therefore, this approach does not solve the problem of determining the importance of the criteria in the first place.

In order to determine the relative importance of the different criteria considered in an MCDM problem, a multitude of methods has been developed. Those weighting methods are usually divided into subjective and objective methods (Nutt, 1980; Odu, 2019; Roszkowska, 2013). In the former, the task of assigning weights to the different criteria is placed on the shoulders of the decision-maker (or, better still, on an expert who may be external to the decision-making process). The expert must assign the weights based on previous experience so that the assignment will necessarily be subject to his or her preferences. On the other hand, in the objective methods, the decision-maker takes no role in the weighting process. This is particularly useful when either the relatively subjective weights cannot be obtained, or when there is no expert available. It is also sometimes the case that the decision-maker does not want to make a judgment on the importance of the criteria in order to maintain a degree of impartiality with respect to the possible outcomes of the ranking of the alternatives.

Subjective methods can be further divided into direct weighting methods and pairwise comparison methods. In the direct weighting schemes, the expert directly assigns values to the importance of the different criteria. For example, in the Simple Multiattribute Rating Technique (SMART) (Edwards, 1977), the criteria are first ranked according to their importance, from the worst attribute levels to the best levels, and then the least important is given 10 points, and the rest are scored relative to the one ranked lowest in importance. Later, Edwards and Barron (1994) improved the method with the SMARTER method, where the centroid method of Solymosi and Dombi (1986) was used, and then the criterion ranked in the i-th position among n different criteria, was given the numerical weight $w_i = \frac{1}{n}\sum_{k=i}^{n}\frac{1}{k}$. Another widely used direct weighting scheme is the Simos method (Figueira and Roy, 2002), where a *"playing cards"* strategy was implemented in order to assess the relative dissimilarities between the different weights. However, some authors have risen concerns regarding the robustness of this method (Shanian et al., 2008; Siskos and Tsotsolas, 2015).

Pairwise comparison methods include the well-known Analytic Hierarchical Process (AHP), which derives relative importance of criteria from paired comparisons of those criteria (Saaty, 1987; Saaty, 1977; Saaty, 1980; Saaty and Vargas, 2012), the eigenvector weighting scheme, initially considered by Saaty (1980), and then extended by Cogger and Yu (1985) and Takeda et al. (1987), incorporating to Saaty's eigenweight vector a least-distance approximation model. Another pairwise comparison method worth mentioning is the Modified Digital Logic (MDL) (Dehghan-Manshadi et al., 2007).

The main characteristic of subjective methods is that they are all based on the initial determination of the relative importance of the criteria by the decision maker (or expert) according to his or her personal experience, beliefs, or preferences.

In contrast to these subjective methods, there are the aforementioned objective methods, among which the following are worth considering: the Mean weighting scheme (or equal weights), which despite its simplicity, it has been found to be very reliable (Dawes and Corrigan, 1974; Schmidt, 1971), the Standard deviation weighting (Jahan et al., 2012), where the weights of the criteria are proportional to their standard deviation, the CRITIC (Criteria Importance Through Inter-criteria Correlation), developed by Diakoulaki et al. (1995) in a financial framework, where the weight of each criterion is depends on both, its standard deviation and the correlation coefficients between the criterion and the rest of the criteria. Finally, several objective weighting schemes have been based on the definition of entropy in classical information theory given by Shannon (1948), where, likewise the SD weighting and the CRITIC method, the weight of each criterion is proportional to some characteristic of the criterion, namely the entropy (see for instance, Kumar et al. (2021) for a recent review on entropy methods applied to engineering problems).

Recently, Liern and Pérez-Gladish (2020) developed an unweighted MCDM TOPSIS-based algorithm (Yoon et al., 1995) to evaluate alternatives avoiding the use of a priori weights. Such an approach meant a significant advance in the field of decision making because one of the most controversial steps was partially solved. Despite the fact that they needed to define a boundary set that limits the weights to vanish or saturate, such selection is way more flexible due to the use of intervals. Among many experimental applications, we would like to mention the selection of alternatives in the automotive sector (Liern and Pérez-Gladish, 2020), evaluation of measures for sustainable development (Benítez and Liern, 2021), the definition of performance indicators for academic purposes (Blasco-Blasco et al., 2021), the assessing of sustainable tourism management in Spain after COVID-19 (Vicens-Colom et al., 2021), and the ranking companies based on their diversity and financial performance (Bouslah et al., 2022). The uwTOPSIS method has been already implemented in R (Benitez and Liern, 2020) and Python (López-García, 2021a), which greatly facilitates the use of this methodology.

5.2.4 *New proposals for compromise solutions*

One of the major contributions of the VIKOR is the introduction of compromise solutions by means of the Q-score that defines the final ranking. In the unweighted approach, we cannot sort the alternatives since there is no direct binary relation that defines a partial order for pairs. Therefore, we present here some possible solutions to ascertain whether the resultant output of uwVIKOR has a compromise solution.

5.2.4.1 *Ranking functions*

When we construct a ranking, we usually make use of a sequence that allows us to arrange the alternatives. In the case of VIKOR, such a sequence of Q_i elements is increasingly sorted since the main comparison is against ideal conditions (item Step 4). It should be noted, however, that there also exist MCDM methods that rank the other way around, such as WSM, WPM, or TOPSIS among others.

Unlike classic multi-criteria optimization problems, the unweighted approaches give us an interval of possible values per each individual $[\min f_i, \max f_i]$ and a set of optimal points $\{W_i^-\}$ and $\{W_i^+\}$. Even though we could define an ordering relationship to deal with intervals, it is more intuitive if we transform the boundaries into a single value. As a consequence, we might consider a continuation of the method by following the last step of ordering.

For the uwVIKOR, we propose three different options to convert the optimal results of $Q_i(w)$, and their respective $S_i(w)$ and $R_i(w)$ evaluated in W_i^L and W_i^U, into a value:

Option 1 Given $0 < \nu_S, \nu_R, \nu_Q < 1$ and $p \in \mathbb{R}$, we define per each $i \in \{1, \dots, N\}$:

$$S_i = \left[\nu_S (S_i^L)^p + (1-\nu_S)(S_i^U)^p\right]^{\frac{1}{p}}, \tag{5.4}$$

$$R_i = \left[\nu_R (R_i^L)^p + (1-\nu_R)(R_i^U)^p\right]^{\frac{1}{p}}, \tag{5.5}$$

$$Q_i = \left[\nu_Q (Q_i^L)^p + (1-\nu_Q)(Q_i^U)^p\right]^{\frac{1}{p}}. \tag{5.6}$$

So we basically operate with the generalized p-mean per each score to aggregate the results obtained.

Option 2 Given $0 < \nu_S, \nu_R, \nu_Q < 1$, we calculate the values of:

$$S_-^L = \min_{1\le i\le N}\{S_i^L\}\ ; \quad S_-^U = \min_{1\le i\le N}\{S_i^U\}\ ; \quad S_+^L = \max_{1\le i\le N}\{S_i^L\}\ ; \quad S_+^U = \max_{1\le i\le N}\{S_i^U\}.$$

$$R_-^L = \min_{1\le i\le N}\{R_i^L\}\ ; \quad R_-^U = \min_{1\le i\le N}\{R_i^U\}\ ; \quad R_+^L = \max_{1\le i\le N}\{R_i^L\}\ ; \quad R_+^U = \max_{1\le i\le N}\{R_i^U\}.$$

$$Q_-^L = \min_{1\le i\le N}\{Q_i^L\}\ ; \quad Q_-^U = \min_{1\le i\le N}\{Q_i^U\}\ ; \quad Q_+^L = \max_{1\le i\le N}\{Q_i^L\}\ ; \quad Q_+^U = \max_{1\le i\le N}\{Q_i^U\}.$$

to propose per each $i \in \{1, \dots, N\}$:

$$S_i = \nu_S \frac{S_i^L - S_-^L}{S_+^L - S_-^L} + (1-\nu_S)\frac{S_i^U - S_-^U}{S_+^U - S_-^U}, \tag{5.7}$$

$$R_i = \nu_R \frac{R_i^L - R_-^L}{R_+^L - R_-^L} + (1-\nu_R)\frac{R_i^U - R_-^U}{R_+^U - R_-^U}, \tag{5.8}$$

$$Q_i = \nu_Q \frac{Q_i^L - Q_-^L}{Q_+^L - Q_-^L} + (1-\nu_Q)\frac{Q_i^U - Q_-^U}{Q_+^U - Q_-^U}. \tag{5.9}$$

Option 3 Following the same line as Option 2, we propose the same strategy as in Eq. 3.31 with the new synthetic S_i and R_i scores defined in equations 5.7 and 5.8.

In any of the mentioned cases, we would proceed with the final scores in the same way that VIKOR Step 5 works with (R_i, S_i, Q_i) to define a new version of the compromise solution for the non-weighted approach.

5.2.4.2 *Optimal dominance*

When multiple alternatives are studied, it is essential to measure how good the compromise solution is. For instance, the acceptable conditions of VIKOR (stages Step 5a and Step 5b) help us to know the dominance of the selected set of alternatives over the remaining ones. In this regard, it is interesting to determine whether an alternative is capable of obtaining the extreme values for the score Q, no matter the final position achieved in the ranking. Hence we introduce the following concepts:

Definition 5.2.1. *Given an MCDM decision matrix $F = [f_{ij}]$ consisting of N alternatives and M criteria, we say that the i^{th} alternative is:*

$$\begin{aligned} &\textit{uwVIKOR-L-dominant} \quad \textit{if} \quad Q_i^L = 0. \\ &\textit{uwVIKOR-U-dominant} \quad \textit{if} \quad Q_i^U = 1. \end{aligned}$$

According to this definition, an alternative is L-dominant if there exists a set of weights for which the alternative can obtain the best possible score ($Q = 0$), while if there are some weights for which the alternative obtains the worst score ($Q = 1$), then we will consider this alternative as U-dominant. Note that those two concepts are not mutually exclusive (i.e., an alternative can be lower and upper dominant at the same time, or be none).

5.2.4.3 *Evaluation of the optimal solutions*

In the step Step 5 of the uwVIKOR, we not only obtain the optimal values of $Q_i(w)$ but also the optimal points (W_i^L, W_i^U). Once we have defined the Ω set and the boundaries of each weight l_j and u_j per each criterion $1 \leq j \leq M$, we have a decision space of feasible elements to solve the mathematical problem. Therefore, each pair (W_i^L, W_i^U) satisfies the condition to be acceptable for a weighting scheme of a MCDM method.

It is well-known that the stage of importance assessment of criteria is very problematic due to the bias of the decision-makers involved. Then, we also have to take into account how suitable are the resultant set of weights. Our proposal is to evaluate the centrality of the (W_i^L, W_i^U) elements. That is to say, we have to check whether a weight w_j saturates the l_j, u_j-restriction, i.e $w_j \in \{l_j, u_j\}$. For this, we define the w-centrality indicator $\phi_{l,u}$, which is basically a membership function of each interval $[l_j, u_j]$. Some examples of w-centrality functions are depicted in Fig. 5.6.

Once we have computed $\{\phi_{l_j,u_j}(w_j)\}_{j=1}^{M}$, we can get information about how balanced is the weighting scheme. We define the joint centrality indicator as the averaged mean of the sequence, as shown in Eq. 5.10. We would like to emphasize that such indicator can be applied over both W_i^L and W_i^U, so we can study the lower and upper balance of the optimal weights.

$$\Phi(w) = \frac{1}{M} \sum_{j=1}^{M} \phi_{l_j,u_j}(w_j), \quad \forall w \in \Omega. \tag{5.10}$$

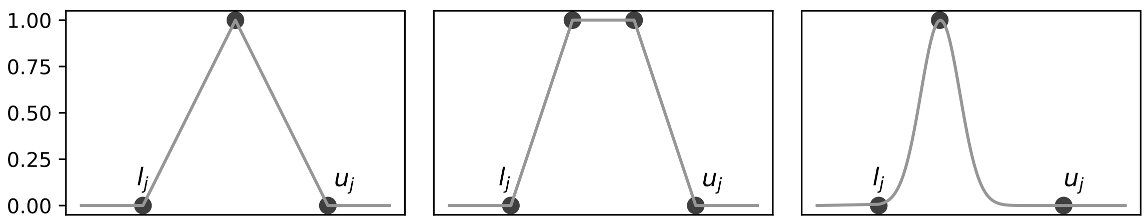


Figure 5.6: Examples of $\phi_{l,u}$ representation with their stages. Lines represent the image of possible values of w_j. Lower points indicates the interval $[l_j, u_j]$ and top points the core of the membership.
Source: Own elaboration.

5.2.5 *Cases study*

5.2.5.1 *Location of a hydropower system*

Opricovic and Tzeng (2007) presented an extension of the VIKOR method for evaluating alternative hydropower systems on the Drina river, a 346 km long river that forms a large portion of the border between Bosnia and Herzegovina and Serbia. From a given set of potential dam sites for reservoirs to provide hydropower, the authors performed a comparative scheme also considering other MCDM techniques. In Table 5.13 is displayed the 6 reservoir systems that were considered as alternatives and a total of 8 criteria composed of 4 economic and 4 social features. With respect to the economic criteria, the profit and cost are measured in 10^6 Dinar currency (Din) and the total and peak energy produced in GW per hour. For the social criteria, the number of homes to relocate and villages to displace are integer features, the surface reservoir area is measured in hectares, and the environmental protection is a grade between 1 and 5.

5.2.5.2 *Choice of the best car*

Jacquet-Lagreze and Siskos (1982) published an example of a real decision-making case in which the aim was to select the best alternative in the automotive industry. As shown in Table 5.12, the problem is composed of ten cars whose criteria are based on engine, space, and economic specifications. The alternative selection has been conducted with detail since there has been a combination of maximization and minimization attributes, in particular three of each. Given that the selection of a model car could be attached to a conflict of interest and so a non-objective framework, it is crucial that we pay special attention to the importance of each feature. Subsequently, Kao (2010) studied the same problem with the aim of determining a weighting scheme with a posterior application of the TOPSIS method.

5.2.5.3 *Selection of the MVP of the NBA*

The study of sports performance is a matter that has been approached in terms of the estimation of characteristics player-dependent. In regard to the NBA case, authors have studied the contribution of players to the win of their teams (Berri, 1999) or their estimated

Table 5.12: Decision matrix from Kao (2010) with 10 cars and their 6 attributes.

Car Model	Maximal speed (km/h)	Horse power (CV)	Space (m^2)	Consump. in town (l/100km)	Consump. at 120km/h (l/100km)	Price (CHF)
Peugeot 505 GR	173	10	7.88	11.4	10.01	49500
Opel Record 2000 LS	176	11	7.96	12.3	10.48	46700
Citroën Visa Super "E"	142	5	5.65	8.2	7.30	32100
VW Golf 1300 GLS	148	7	6.15	10.5	9.61	39150
Citroën CX 2400 Pallas	178	13	8.06	14.5	11.05	64700
Mercedes 230	180	13	8.47	13.6	10.40	75700
BMW 520	182	11	7.81	12.7	12.26	68593
Volvo 244 DL	145	11	8.38	14.3	12.95	55000
Peugeot 104 ZS	161	7	5.11	8.6	8.42	35200
Citroën Dyane	117	3	5.81	7.2	6.75	24800
Optimization criteria	Max	Max	Max	Min	Min	Min
Positive ideal (F^*)	182.00	13.00	8.47	14.50	12.95	75700
Negative ideal (F^-)	117.00	3.00	5.11	7.20	6.75	24800

Table 5.13: Decision matrix from Opricovic and Tzeng (2007) with the 6 different reservoir systems and 8 economic and social criteria.

	Profit	Costs	Total energy produced	Peak energy produced	Homes to be relocated	Reservoirs area	Villages to displace	Environm. protection
	(10^6Din)	(10^6Din)	(GW/h)	(GW/h)	(Integer)	(ha)	(Integer)	(Grade)
A_1	4184.3	2914.0	407.2	251.0	195	244.0	15	2.41
A_2	5211.9	3630.0	501.7	308.3	282	346.0	21	1.41
A_3	5021.3	3920.5	504.0	278.6	12	56.0	3	4.42
A_4	5566.1	3957.9	559.5	335.3	167	268.0	16	3.36
A_5	5060.5	3293.5	514.1	284.2	69	90.0	7	4.04
A_6	4317.9	2925.9	432.8	239.3	12	55.0	3	4.36
F^+	5566.1	2914.0	559.5	335.3	12	55.0	3	4.42
F^-	4184.3	3957.9	407.2	239.3	282	346.0	21	1.41
	Max	Min	Max	Max	Min	Min	Min	Max

Table 5.14: Statistics of the main nominees to the MVP, voted at least once "Pts Won", of the 2019-2020 regular season where the maximum number of points was 1010.

Player	Age	Tm	Pts Won	1st Pos	G	MP	PTS	TRB	AST	STL	BLK	FG%	3P%	FT%	WS	WS/48
Giannis Antetokounmpo	25	MIL	962	85	63	30.4	29.5	13.6	5.6	1.0	1.0	0.553	0.304	0.633	11.1	0.279
LeBron James	35	LAL	753	16	67	34.6	25.3	7.8	10.2	1.2	0.5	0.493	0.348	0.693	9.8	0.204
James Harden	30	HOU	367	0	68	36.5	34.3	6.6	7.5	1.8	0.9	0.444	0.355	0.865	13.1	0.254
Luka Dončić	20	DAL	200	0	61	33.6	28.8	9.4	8.8	1.0	0.2	0.463	0.316	0.758	8.8	0.207
Kawhi Leonard	28	LAC	168	0	57	32.4	27.1	7.1	4.9	1.8	0.6	0.470	0.378	0.886	8.7	0.226
Anthony Davis	26	LAL	82	0	62	34.4	26.1	9.3	3.2	1.5	2.3	0.503	0.330	0.846	11.1	0.250
Chris Paul	34	OKC	26	0	70	31.5	17.6	5.0	6.7	1.6	0.2	0.489	0.365	0.907	8.9	0.193
Damian Lillard	29	POR	23	0	66	37.5	30.0	4.3	8.0	1.1	0.3	0.463	0.401	0.888	11.6	0.225
Nikola Jokić	24	DEN	18	0	73	32.0	19.9	9.7	7.0	1.2	0.6	0.528	0.314	0.817	9.8	0.202
Pascal Siakam	25	TOR	17	0	60	35.2	22.9	7.3	3.5	1.0	0.9	0.453	0.359	0.792	5.4	0.123
Jimmy Butler	30	MIA	9	0	58	33.8	19.9	6.7	6.0	1.8	0.6	0.455	0.244	0.834	9.0	0.221
Jayson Tatum	21	BOS	1	0	66	34.3	23.4	7.0	3.0	1.4	0.9	0.450	0.403	0.812	6.9	0.146

skills and abilities with a statistical perspective (Fearnhead and Taylor, 2011). For team analysis, some authors as Mikołajec et al. (2013) have designed performance indicators and other underlying factors that affect the game such as ethic (Kraus et al., 2010), pressure (Cao et al., 2011) or payments (Sigler and Sackley, 2000).

Since 1956, the National Basketball Association of United States grants the player that has shown the best performance during the regular season with an iconic award. Such accomplishment is rewarded with the so-called MVP trophy, whose abbreviation means Most Valuable Player. The trophy is also known as Maurice Podoloff in honor of the first chief executive of the NBA. The judging criteria to assess the players has changed over the years. Until 1980 the election system was held by the players of the league, who voted at the end of it. From then onwards, the selection is conducted by an expert committee of journalist and broadcasters of USA and Canada.

In regard with the voting scheme, each person in charge votes for first to fifth place to five different players, where each position is worth 10, 7, 5, 3, and 1 respectively. Once the scrutiny is conducted, the player with higher points is the one who wins the trophy. Even though there exist multiple statistics followed by a follow-up report that help the experts to make their final decision, such response is based on their personal perspective. In addition, the number of features to analyze and the pair-wise comparisons are not an easy task. Hence, we cannot assert that the Maurice Podoloff Trophy is given under objective conditions.

In order to limit the computational cost that would imply a joint decision matrix with every single player that have participated at least once in the NBA, we have decided to pick the players that have obtained at least one point in the voting system. The experimental dataset contains the statistics of the players from 2001 to 2020, in which the number of players per year have varied because of the amount of votes received. As a result, the number of alternatives to consider per problem has ranged between 9 and 17 players. Altogether, we have collected a total 283 entries. The data is publicly available in the basketball-reference website[1]. The entire dataset is presented in Table 5.14, where players are divided by season.

5.2.6 *Results and discussion*

In this section, we present the application of the uwVIKOR algorithm in three different scenarios based on the three case studies. The first case study (5.2.5.1) studies the extension of our method with respect to the classical VIKOR. We show the advantage and flexibility that supposes the use of weighting bounds instead of precise weight vectors. The second case study (5.2.5.2) contains a step-by-step procedure of how to carry out the uwVIKOR for a given MCDM problem. As the approach to this problem is rather problematic, we show the importance of the concepts of L,U-dominance offered by uwVIKOR that cannot be studied with uwTOPSIS. The third case study (5.2.5.3) contains an in-depth evaluation of a predictive ranking system for NBA players. The aim is to extract the most performing player by means of basketball statistics, thus removing non-objective parameters during

1 https://www.basketball-reference.com/

Table 5.15: Statistics of the main nominees to the MVP, voted at least once (Pts Won), of the 2020 regular season. The $*$ index means that such value is the average per number of games played.

Acronym	**Description**
Tm	Name of the player's team
Pts Won	Points won in the voting system
1st Pos	Number of times voted as 1st place
G	Games played in the season
MP*	Minutes played
PTS*	Points
TRB*	Total rebounds
AST*	Assists
STL*	Steals
BLK*	Blocks
FG %*	Field goal as percentage
3P %*	3-Point field goal as percentage
FT %*	Free throw as percentage
WS	Winning shares per game
WS/48*	Winning shares per 48 minutes

the selection process. Furthermore, the performance of uwVIKOR when ranking has been analyzed using fitting measures.

5.2.6.1 *Hydropower system localization*

Table 5.16: Analysis of the compromise solutions obtained per each ranking method and the study of the components of such compromise solutions.

	Ranking	$Q_{\sigma_Q(2)} - Q_{\sigma_Q(1)}$	Advantage	Stability
Balanced	$A_5 \succeq A_3 \succeq A_6 \succeq A_4 \succeq A_1 \succeq A_2$	0.4731	✓	✓
Economy	$A_5 \succeq A_2 \succeq A_3 \succeq A_4 \succeq A_6 \succeq A_1$	0.5329	✓	✓
Social	$A_5 \succeq A_3 \succeq A_6 \succeq A_4 \succeq A_1 \succeq A_2$	0.1059	-	✓
Unweighted	$A_5 \succeq A_3 \succeq A_6 \succeq A_4 \succeq A_2 \succeq A_1$	0.2763	✓	✓

In view of the fact that the problem posed by Opricovic and Tzeng (2007) involves a major conflict of interests, we have computed both VIKOR and uwVIKOR for different weighting schemes and boundaries. The main authors proposed three different assessment strategies based on the expected judgment per each criterion. The weighting schemes were

Table 5.17: Decision matrix from Opricovic and Tzeng (2007) with the 6 different reservoir systems and 8 economic and social criteria.

	Balanced			**Economy**			**Social**			**Unweighted**					
	S	R	Q	S	R	Q	S	R	Q	S^L	S^U	R^L	R^U	Q^L	Q^U
A_1	0.692	0.125	0.991	0.701	0.167	1.000	0.683	0.113	0.684	0.636	0.734	0.092	0.167	0.528	1.000
A_2	0.700	0.125	1.000	0.600	0.114	0.533	0.800	0.167	1.000	0.611	0.731	0.092	0.167	0.382	1.000
A_3	0.290	0.121	0.473	0.386	0.161	0.552	0.193	0.080	0.147	0.254	0.386	0.080	0.161	0.042	0.552
A_4	0.423	0.125	0.670	0.365	0.167	0.563	0.480	0.122	0.554	0.311	0.534	0.083	0.167	0.015	0.779
A_5	0.280	0.067	0.000	0.317	0.089	0.000	0.243	0.044	0.041	0.263	0.243	0.046	0.044	0.000	0.041
A_6	0.346	0.125	0.578	0.459	0.167	0.686	0.232	0.083	0.191	0.232	0.459	0.083	0.167	0.107	0.686

designed to perform a ranking system by means of balanced (5.11), economy (5.12), and social (5.13) approaches.

$$\text{Balanced } (W_B): \quad w_j = \tfrac{1}{8}, \quad \forall j \in \{1,\dots,8\}, \tag{5.11}$$

$$\text{Economy } (W_E): \quad w_j = 2w_k, \quad \forall j \in \{1,\dots,4\}, k \in \{5,\dots,8\}, \tag{5.12}$$

$$\text{Social } (W_S): \quad 2w_j = w_k, \quad \forall j \in \{1,\dots,4\}, k \in \{5,\dots,8\}. \tag{5.13}$$

The objective is to study such a problematic decision scenario by means of the schemes proposed and show the advantage that supposes our unweighted framework. Instead of performing several experimental tests, our method is able to analyze all possible scenarios among the approaches proposed in the original problem. For the implementation of uwVIKOR, we have selected decision bounds composed of the minimal and maximal weights of the three strategies. In other words, we have the following pairs:

$$\begin{cases} l_j = & \min\{\min\{W_B\},\min\{W_E\},\min\{W_S\}\} = & 1/12, \\ u_j = & \max\{\max\{W_B\},\max\{W_E\},\max\{W_S\}\} = & 1/6. \end{cases} \tag{5.14}$$

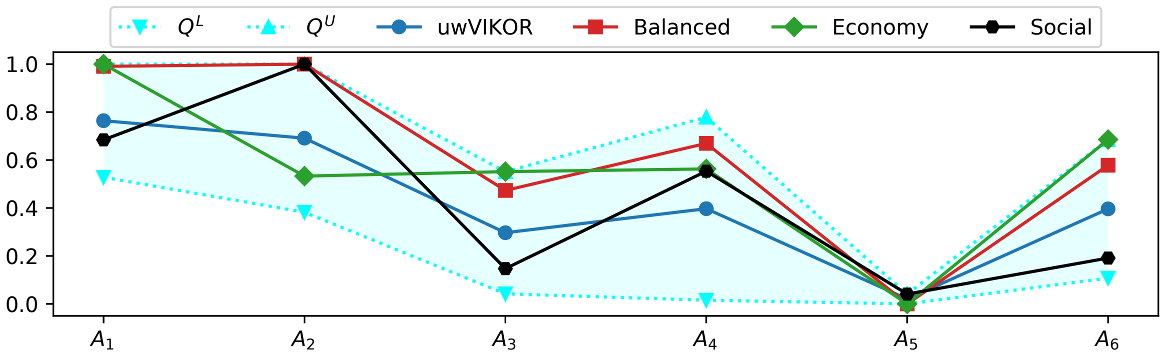


Figure 5.7: Comparison of the score systems obtained for the balanced, economy, and social weighting schemes of Opricovic and Tzeng (2007) and our uwVIKOR method ($\nu = 0.5$ and $p = 1$) for the hydropower plant location problem.

By doing this, the resultant (Q^L, Q^U) score vectors contain the Q vectors of the VIKOR method for the three strategies defined by Opricovic and Tzeng (2007). In addition, we can observe how susceptible the alternatives are regarding the weight selection by analyzing the amplitude of the scores as mentioned in Equations 5.4, 5.5, and 3.67. The results obtained are shown in Table 5.17, where it is presented the (S, R, Q) score vectors of each ranking system and the intervals of these scores through our unweighted VIKOR method. Moreover, we have depicted the results in Fig. 5.7, in which the uwVIKOR has been aggregated using the arithmetic mean of Q^L and Q^U.

The advantage of uwVIKOR over standard VIKOR implementations in terms of offering ranking systems is worth noting. As we can see in Fig. 5.7, each of the scores of the weighted methods are contained into the $[Q^L, Q^U]$ score intervals of uwVIKOR since their weights belong to the Ω set. In addition, we can study the dominance of the alternatives when determining compromise solutions. In this particular case, we conclude that $A_5 \succeq A_i$ because $Q_5^U \leq Q_i^L$ for all $i \neq 4$, in other words, $Q_5(w) \leq Q_i(w)$ as long as $w \in \Omega$ and $i \neq 4$.

In regard to the compromise solutions, the results obtained per each method have been analyzed to check whether we can guarantee advantage and/or stability conditions for the methods. In Table 5.16, we have displayed the ranking of the alternatives with the conditions required to build each compromise solution.

5.2.6.2 *Cars ranking system*

For the first experiment, we want to obtain the best car from a given set of alternatives with different automotive sector attributes. In order to carry out the raking process, we conduct a step-by-step procedure to show the whole process. As we described in 3.4.3, we have to extracts ideal solutions F^* and F^- to subsequently normalize decision matrix $[f_{ij}]$ (Table 5.12). The transformation $[r_{ij}]$ is presented in Table 5.18.

Table 5.18: Car decision matrix normalized as in Eq. 3.28

Car Model	Maximal speed (km/h)	Horse power (CV)	Space (m^2)	Consump. in town (l/100km)	Consump. at 120km/h (l/100km)	Price (CHF)
Peugeot 505 GR	0.1384	0.3000	0.1755	0.5753	0.5258	0.4852
Opel Record 2000 LS	0.0923	0.2000	0.1517	0.6986	0.6016	0.4302
Citroën Visa Super “E”	0.6153	0.8000	0.8392	0.1369	0.0887	0.1434
VW Golf 1300 GLS	0.5230	0.6000	0.6904	0.4520	0.4612	0.2819
Citroën CX 2400 Pallas	0.0615	0.0000	0.1220	1.0000	0.6935	0.7838
Mercedes 230	0.0307	0.0000	0.0000	0.8767	0.5887	1.0000
BMW 520	0.0000	0.2000	0.1964	0.7534	0.8887	0.8603
Volvo 244 DL	0.5692	0.2000	0.0267	0.9726	1.0000	0.5933
Peugeot 104 ZS	0.3230	0.6000	1.0000	0.1917	0.2693	0.2043
Citroën Dyane	1.0000	1.0000	0.7916	0.0000	0.0000	0.0000

In the same line than Liern and Pérez-Gladish (2020), we have selected the boundaries $l_j = 0.01$ and $u_j = 0.75$ per each criterion $j \in \{1, ..., 6\}$. This constraint selection was already based on the weights used by Kao (2010). The Ω is thus defined and the score functions can be utilized to perform the steps Step 3, Step 4 and Step 5. The results obtained are shown in Table 5.19, where the uwVIKOR column is the arithmetical mean between Q^L and Q^U, so we have decided to set $\nu_Q = 0.5$ for the Option 1 in the ranking system.

The uwVIKOR ranking score places the Peugeot 505 GR in the first position, making it the best choice with regard to the set of alternatives. A comparative plot is depicted in Fig. 5.8 to see the score variations between alternatives. In addition, the score of Liern and Pérez-Gladish (2020) is added with a change in their result of $1 - R_i$, being R_i their uwTOPSIS score. For the dominance approach, it is interesting that just the VW Golf 1300 GLS is considered U-dominant with a minimal score of 0.0841, despite having balanced

Table 5.19: Score functions of the uwVIKOR method per each alternative.

Car Model	S^L	S^U	R^L	R^U	Q^L	Q^U	uwVIKOR
Peugeot 505 GR	0.3281	0.5218	0.1053	0.2278	0.0000	0.5911	0.2955
Opel Record 2000 LS	0.2167	0.6424	0.1152	0.5239	0.0000	0.6940	0.3470
Citroën Visa Super "E"	0.2913	0.8016	0.1102	0.5287	0.0000	0.8963	0.4481
VW Golf 1300 GLS	0.3994	0.5349	0.1947	0.2533	0.0841	0.8098	0.4470
Citroën CX 2400 Pallas	0.1417	0.8782	0.1014	0.7400	0.0000	1.0000	0.5000
Mercedes 230	0.0997	0.8397	0.0526	0.7500	0.0000	1.0000	0.5000
BMW 520	0.1295	0.8561	0.0516	0.5876	0.0000	0.9435	0.4717
Volvo 244 DL	0.1814	0.8903	0.0947	0.7500	0.0000	1.0000	0.5000
Peugeot 104 ZS	0.3052	0.8357	0.0999	0.7500	0.0000	1.0000	0.5000
Citroën Dyane	0.1175	0.8575	0.0600	0.7500	0.0000	1.0000	0.5000

automotive conditions. Moreover, we have five U-dominant cars that are actually placed in the last positions of the ranking.

In regard to the optimal solutions attached to the uwVIKOR, Table 5.20 contains the weight matrices W^L and W^U. We can see that the weights have saturated the lower bounds l_j 23 times (10 for Q^L and 13 for Q^U) and the upper bounds u_j 8 times (3 for Q^L and 5 for Q^U). It indicates that, for this particular problem, the weighting scheme is more prone to vanish than explode to 1. Moreover, there is only one case of upper saturation in which the alternative has not achieved the best optimal possible. That is the case of maximization of the Opel Record 2000 LS, where the maximal value was 0.6940. In all other cases, there has been a result of either 0 or 1.

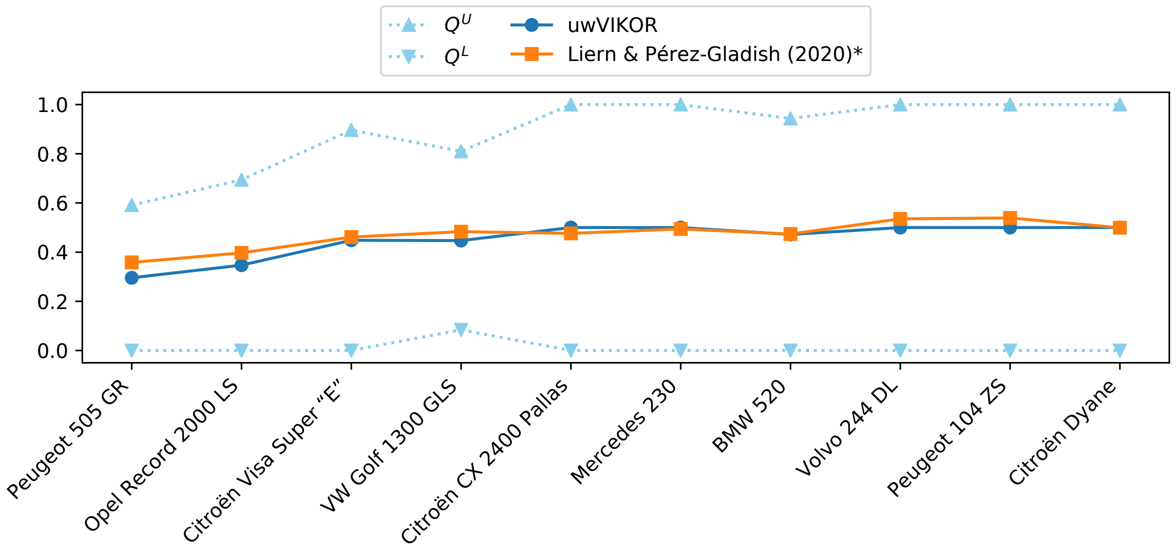


Figure 5.8: Ranking systems obtained for our uwVIKOR method (Q^L, Q^U, and uwVIKOR) and the uwTOPSIS of Liern and Pérez-Gladish (2020), where the $*$ mark indicates that such score has been reversed in order to present the same ascending order than VIKOR technique

Table 5.20: Optimal weights of the uwVIKOR optimization problem for both Q^L and Q^U cases. Bold values highlight the upper saturation of the weight, i. e. $w_j = u_j$.

	Car Model	Maximal speed (km/h)	Horse power (CV)	Space (m^2)	Consump. in town (l/100km)	Consump. 120km/h (l/100km)	Price (CHF)
Q^L optimal weights: W^L	Peugeot 505 GR	0.4406	0.0100	0.0783	0.1830	0.1314	0.1565
	Opel Record 2000 LS	0.5985	0.0412	0.0412	0.0100	0.0412	0.2677
	Citroën Visa Super "E"	0.1752	0.1378	0.0100	0.0462	0.5842	0.0462
	VW Golf 1300 GLS	0.0100	0.3245	0.0205	0.0100	0.0100	0.6248
	Citroën CX 2400 Pallas	0.0462	0.7470	0.0336	0.0100	0.0336	0.1293
	Mercedes 230	0.0600	**0.7500**	0.0600	0.0600	0.0600	0.0100
	BMW 520	**0.7500**	0.0600	0.0600	0.0600	0.0100	0.0600
	Volvo 244 DL	0.0327	0.0327	0.7319	0.0327	0.0100	0.1596
	Peugeot 104 ZS	0.2572	0.1666	0.0100	0.4867	0.0396	0.0396
	Citroën Dyane	0.0100	0.0600	0.0600	**0.7500**	0.0600	0.0600
Q^U optimal weights: W^U	Peugeot 505 GR	0.0100	0.0100	0.0100	0.3961	0.2295	0.3443
	Opel Record 2000 LS	0.0100	0.0100	0.0100	**0.7500**	0.1133	0.1066
	Citroën Visa Super "E"	0.0100	0.3285	0.6300	0.0114	0.0100	0.0100
	VW Golf 1300 GLS	0.2345	0.0100	0.3668	0.1184	0.1296	0.1404
	Citroën CX 2400 Pallas	0.0100	0.0100	0.0749	0.7400	0.0100	0.1550
	Mercedes 230	0.0600	0.0100	0.0600	0.0600	0.0600	**0.7500**
	BMW 520	0.0100	0.0100	0.0100	0.0100	0.6612	0.2987
	Volvo 244 DL	0.0600	0.0600	0.0100	0.0600	**0.7500**	0.0600
	Peugeot 104 ZS	0.0600	0.0600	**0.7500**	0.0100	0.0600	0.0600
	Citroën Dyane	**0.7500**	0.0600	0.0600	0.0100	0.0600	0.0600

The results show a big amplitude in the intervals $[Q_i^L, Q_i^U]$, where there are five cases of full amplitude $[0, 1]$. It is noteworthy that our results are very similar to the ones of Liern and Pérez-Gladish (2020), in which the major contrast can be seen in the first alternative. Another remarkable detail is that both schemes return the same winner, i. e. the Peugeot 505 GR.

5.2.6.3 *Selection of the MVP player per season*

For the second experiment, our goal is to present an objective solution to the choice of the MVP problem by applying the uwVIKOR technique. In such a way we guarantee an objective scheme to extract compromise solutions that lead to the choice of the best performance during the year. For this purpose, we have collected the statistics of the players for each year, as shown in Table 5.14. For a better understanding of the acronyms, Table 5.15 explains their meaning. We would like to highlight that we have selected features in regard to attack, defense, and physical characteristics.

For the uwVIKOR implementation case, the procedure of the algorithm was held by narrowing down the headers of the dataset to G, MP, PTS, TRB, AST, STL, BLK, FG%, and

WS/48. Then, we gave relevant importance only to seven variables. Thus, in an equitable scenario, the contribution of each feature to the final decision would be 14.29%. Their relative importance within the ranking ranges from 0.1 to 0.4, so $l_j = 0.1$ and $u_j = 0.4$, $\forall j \in \{1, ..., M\}$.

In order to ease the computational cost that requires both MCDM application and the mathematical programming under $Q(w)$ optimization, we have developed a repository in GitHub with open access to the project in PyPI (López-García, 2021b). An example of the implementation of the technique can be found in C.2.2.

We have fixed $v_Q = 0.5$ so as to study the generalized p-mean over $p \in \{0, 1, 2\}$, i.e we have computed the geometric, arithmetic and quadratic means as in Option 1. Fig. 5.12 illustrates the results obtained per season and the impact of the p-mean when estimating the final score. In regard to the results, it is easy to note that $GM \leq AM \leq QM$, due to the properties of the mean. However, the results significantly vary when a player is considered uwVIKOR-L-dominant, most notably for the geometric mean. For this instance, a player considered L-dominant will always be ranked in the first position, even though this first position could be shared with other contenders.

During the 20 years studied, we have realized that in seven years we could unanimously assign the MVP trophy, i.e. that there is a number one regardless of the p-median selected. That is the case of Garnett (2004), four times James (2009, 2010, 2012, 2013), Durant (2014), and Antetokounmpo (2019). Stated another way, these four players have shown their dominance in statistical terms, so their level on the court is undoubtedly insuperable.

Another interesting result is that not every player that won an MVP award in a particular year, has obtained the first position in our ranking system in the same year. This is the case for Nash (2005 and 2006), Bryant (2008), Rose (2011), and Curry (2015). With respect to the three different scores considered, they did not achieve the first position in any circumstances. We would want to mention that they can get the 2nd position with the arithmetic mean, except for Curry whose best possible position is the 3rd place. For the resultant cases, although there are cases where they are not classified as the best, there exists an option for which they can be ranked far beyond their competitors.

Since we want to evaluate the uwVIKOR estimations, Fig. 5.9 illustrates the results obtained (averaged) of the cardinal position obtained with the generalized p-means implemented. Then, we can study the deviations between the real position obtained by the expert panel and the predicted by our method. It is essential to note that each p-score has been averaged, so now they do not have to respect the property said in the first paragraph of the section. Regarding the cardinal system obtained, we got an increasing tendency which follows the result that we expected.

An additional evaluation of Fig. 5.9 is the Table 5.21. By applying two performance measures, we have numerically evaluated the averaged results of Fig. 5.12. On the one hand, we evaluated the linear relationship between the results obtained and the actual ranking with the Pearson correlation or R^2. On the other hand, we measured the cardinal deviation within the ranking with the mean absolute error or MAE.

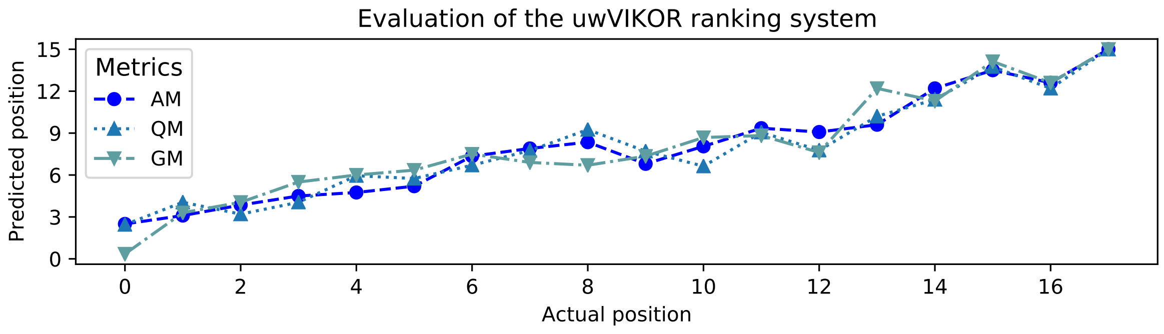


Figure 5.9: Cardinal ranking comparison of the uwVIKOR with arithmetic (AM), quadratic (QM) and geometric (GM) means. X-axis is the official ranking by year and Y-axis is the ranking uwVIKOR-based.

Source: Own elaboration.

Table 5.21: Performance measures (R^2 and MAE) for the predicted positions given by the generalized p-means.

Measure	**Arithmetic**	**Quadratic**	**Geometric**
R^2	0.9706	0.9455	0.9485
MAE	1.7954	2.0244	1.8195

We can conclude by saying that, in our dataset, the arithmetic mean has been the most suitable summary function for the Option 1. Such mean has shown better results in both deviation and positioning as we can see in Table 5.21.

5.2.6.4 *L,U-dominance per year*

In order to study the performance of the top members of each year, Table 5.22 shows the number of players considered as lower or upper dominants. Since their definition is not complimentary, we have decided to figure out how many players belong to each category as an independent event.

WWe also want to underline the particular analysis of the actual MVPs. It is interesting that in five years, the trophy has been given to U-dominant players. As stated in 5.2.6.3, these players are Nash, Bryant, Rose, and Curry. The most notorious cases are Derrick Rose because his Q^L is 0.2239 for 2011, and Steve Nash and Stephen Curry who obtained a minimal value of 0.0049 in 2005 and 0.0051 in 2015, respectively. This proves that, although we have attempted to create a methodology that studies the particular case of each individual, there still exist some limitations associated with the criteria selected and their boundaries $\{l_j, u_j\}$.

Nevertheless, when considering the dominance condition, there is no MVP rewarded member that can be labeled as U-dominant in any studied year. In addition, there are no players that can be considered lower and upper dominant at the same time. Even though it might seem obvious, the condition based on Definition 5.2.1 allows an alternative to belong to both groups at the same time. That can be understood as proof of the consistency of the procedure that constitutes the uwVIKOR algorithm.

Table 5.22: Summary of the players by the conditions stated in Definition 5.2.1. The table is broken down by total amount of elements and its percentage.

	L-dominant		**U-dominant**	
Year	Total	Percentage	Total	Percentage
2001	2	11.76	3	17.65
2002	6	33.33	2	11.11
2003	5	38.46	2	15.38
2004	2	12.50	4	25.00
2005	4	25.00	2	12.50
2006	5	45.45	2	18.18
2007	5	29.41	3	17.65
2008	4	23.53	2	11.76
2009	4	33.33	2	16.67
2010	2	13.33	5	33.33
2011	3	23.08	4	30.77
2012	2	13.33	1	06.67
2013	2	12.50	2	12.50
2014	6	35.29	2	11.76
2015	3	25.00	2	16.67
2016	4	40.00	1	10.00
2017	6	54.55	2	18.18
2018	3	23.08	1	07.69
2019	4	33.33	4	33.33
2020	4	33.33	1	08.33

5.2.6.5 *Evaluation of optimal weights*

We have decided to generalize the stability of the weights by implementing the standard w-centrality function defined in Eq. 5.15. It matches with the fuzzy membership function of a linear trapezoidal number with spread and core of the same size, in our case $\frac{1}{3}(u-l)$. Its graph would be similar to the center plot of Fig. 5.6.

$$\phi_{l,u}(w_j) = \begin{cases} 3 \cdot \dfrac{w_j - l}{u-l} & \text{if } \ l \leq w_j \leq \frac{1}{3}(u+2l), \\ 1 & \text{if } \ \frac{1}{3}(u+2l) \leq w_j \leq \frac{1}{3}(2u+l), \\ 3 \cdot \dfrac{u - w_j}{u-l} & \text{if } \ l + \frac{1}{3}(2u+l) \leq w_j \leq u, \\ 0 & \text{if } \ w_j \notin [l,u]. \end{cases} \tag{5.15}$$

Fig. 5.13 depicts, for each year, the joint stability indicator for both the lower and upper optimal solutions. There, the degree of saturation of the optimal solutions can be easily determined.

In addition to being a useful measure to assess the selection of criteria and how it affects the empirical performance of the players, the function Φ can also play a key role when solving conflicts in matches. For instance, when there is a theoretical match between alternatives just like in 2020, we can make the decision of picking the player with the higher balance. Thus, we are promoting either versatile players or players with high influence on the success of their teams.

We would like to focus on the 2020 season because it has been one of the most challenging in terms of decision making. In that year, the uwVIKOR output gave us a complicated situation in which four players deserved the prize in almost similar conditions: Antetokounmpo, James, Harden, and Davis. Even though the scenario was very complex, the Φ-indicator led us to the final decision, in which the actual MVP winner (Antetokounmpo) was the one that dominated the compromise solution.

5.2.6.6 *Comparison against other MCDM methods: MVP award selection in 2021*

When evaluating a new MCDM technique, it is essential to compare it with other similar approaches in order to analyze the main differences among them. Owing to the raise of applications with uwTOPSIS, we have decided to contrast the results in the case of the NBA league. Moreover, we have published a GitHub repository (López-García, 2021a) with the algorithm that speeds up the optimization problems. Then, we have two techniques with two different approaches, i.e. classic and unweighted.

In order to give a slightly different approach, we now consider the player performance for the 2021 case. The data collected is presented in Table 5.23, in which the source and procedure were the same as in § 5.2.5.3.

5.2.6.7 *Comparing the output of un-weighted methods*

Owing to the lack of literature related to unweighted MCDM theory, we compare our method with the state-of-the-art technique uwTOPSIS. Thus, we can analyze the resultant p-mean score (average mean) and the oscillation of the intervals $[Q^L, Q^U]$. In the same way as the beginning of § 5.2.6, we have limited $l_j = 0.1$ and $u_j = 0.4$ per each attribute $j \in \{1, ..., M\}$ and we set $p = \frac{1}{2}$. With this purpose, we have represented the central value with its range for both methods in Fig. 5.10.

It is worth mentioning that the numeric cardinal produced for TOPSIS is the opposite of VIKOR. Instead of minimizing the score, in TOPSIS greater values mean better positioning. If we analyze Fig. 5.10, we can notice a more flexible score system for the uwVIKOR method. When comparing the amplitude of the optimal intervals, we have 0.6131 ± 0.1476, for (Q^L, Q^U)-scores, and 0.3531 ± 0.0897, for (R^L, R^U)-scores. The underpinning idea of this result is that exists a higher range of values that could be achieved by modifying the initial weighting schemes when applying classic MCDM models.

As we highlighted at the beginning of the paper, the order is important but the acceptance of the final decision should verify some rules to increase the reliability of the output. That is the advance presented in uwVIKOR. For the uwTOPSIS, we carried out the ranking and we just pick the first one, in our case, Rudy Gobert. For uwVIKOR, Antetokounmpo

Table 5.23: Statistics of the main nominees to the MVP, voted at least once "Pts Won", of the 2020-2021 regular season where the maximum number of points was 1010.

Player	Age	Tm	Pts Won	1st Pos	G	MP	PTS	TRB	AST	STL	BLK	FG%	3P%	FT%	WS	WS/48
Nikola Jokić	25	DEN	91.0	971.0	72	34.6	26.4	10.8	8.3	1.3	0.7	0.566	0.388	0.868	15.6	0.301
Joel Embiid	26	PHI	1.0	586.0	51	31.1	28.5	10.6	2.8	1.0	1.4	0.513	0.377	0.859	8.8	0.266
Stephen Curry	32	GSW	5.0	453.0	63	34.2	32.0	5.5	5.8	1.2	0.1	0.482	0.421	0.916	9.0	0.201
Giannis Antetokounmpo	26	MIL	1.0	348.0	61	33.0	28.1	11.0	5.9	1.2	1.2	0.569	0.303	0.685	10.2	0.244
Chris Paul	35	PHO	2.0	139.0	70	31.4	16.4	4.5	8.9	1.4	0.3	0.499	0.395	0.934	9.2	0.201
Luka Dončić	21	DAL	0.0	42.0	66	34.3	27.7	8.0	8.6	1.0	0.5	0.479	0.350	0.730	7.7	0.163
Damian Lillard	30	POR	0.0	38.0	67	35.8	28.8	4.2	7.5	0.9	0.3	0.451	0.391	0.928	10.4	0.209
Julius Randle	26	NYK	0.0	20.0	71	37.6	24.1	10.2	6.0	0.9	0.3	0.456	0.411	0.811	7.8	0.140
Derrick Rose	32	TOT	1.0	10.0	50	25.6	14.7	2.6	4.2	1.0	0.4	0.470	0.388	0.866	3.1	0.118
Rudy Gobert	28	UTA	0.0	8.0	71	30.8	14.3	13.5	1.3	0.6	2.7	0.675	0.000	0.623	11.3	0.248
Russell Westbrook	32	WAS	0.0	5.0	65	36.4	22.2	11.5	11.7	1.4	0.4	0.439	0.315	0.656	3.7	0.075
Ben Simmons	24	PHI	0.0	3.0	58	32.4	14.3	7.2	6.9	1.6	0.6	0.557	0.300	0.613	6.0	0.153
James Harden	31	TOT	0.0	1.0	44	36.6	24.6	7.9	10.8	1.2	0.8	0.466	0.362	0.861	7.0	0.208
LeBron James	36	LAL	0.0	1.0	45	33.4	25.0	7.7	7.8	1.1	0.6	0.513	0.365	0.698	5.6	0.179
Kawhi Leonard	29	LAC	0.0	1.0	52	34.1	24.8	6.5	5.2	1.6	0.4	0.512	0.398	0.885	8.8	0.238

would be the one that deserved the MVP, nonetheless, this solution is not acceptable according to Step 5a and Step 5b conditions. Therefore, we can create a compromise solution shared by Jokić, Embiid, and Antetokounmpo so that it creates a compromise solution.

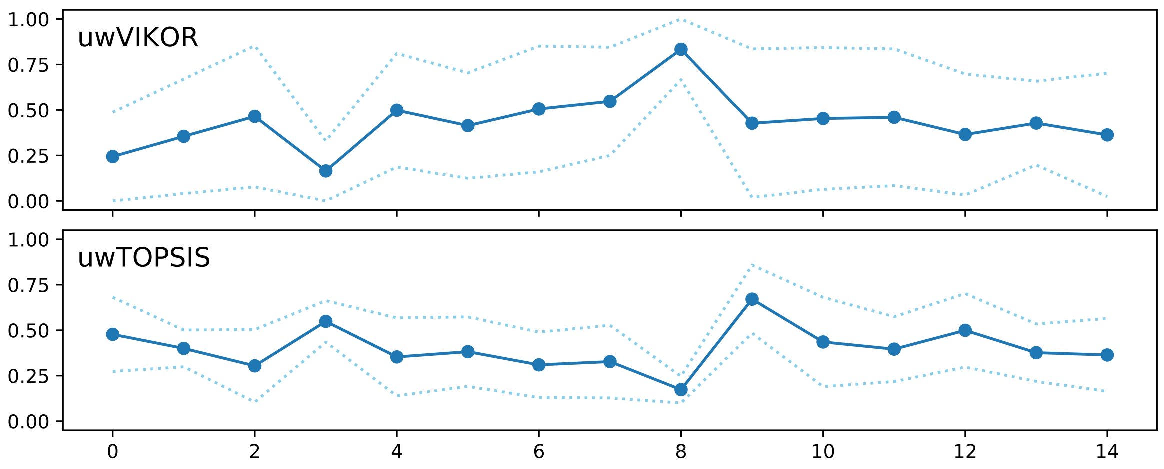


Figure 5.10: Comparison between uwVIKOR (top) and uwTOPSIS (bottom). Dotted lines illustrates the lower an upper interval per each case.
Source: Own elaboration.

5.2.6.8 *Ranking systems: VIKOR vs TOPSIS*

The use of outranking models to arrange alternatives is wide use. In particular, we can say that both VIKOR and TOPSIS are the most popular techniques to sort members when there exist conflicting criteria. That is why we have decided to compare such methods for the NBA ranking system. Moreover, this kind of comparison is very extended, Opricovic and Tzeng (2004) made a numerical example showing similarities and some differences.

The data utilized in § 5.2.6 is supposed to be well structured and criteria independent, we accept that the impact of the seven features on the final decision is the same. In other words, the classical algorithms have a weighting scheme of $w_j = 1/7$ for all $j \in \{1, \dots, M\}$. In addition, the unweighted algorithms ranked the alternatives by utilizing the arithmetic mean of the optimal interval obtained. The Python implementation of classic approaches is explained in § A.3.

We have decided to transform the R_i score of TOPSIS (both approaches) per each alternative per $1 - R_i$, being $i \in \{1, ..., N\}$. By doing so, we still respect the ranking system, but now we have the same order that VIKOR does when sorting the nominees. Therefore, we can judge the results obtained by taking into account that, the lower the value, the better positioned in the ranking.

We could remark that VIKOR method is the one with the most variation score between alternatives, in fact, it ranges from 0.0383 to 1.0. By contrast, TOPSIS has returned the least variation with values between 0.1682 and 0.6336, which means an oscillation of 0.4653. For the models with no weights, it is interesting to point out that, except for Rudy Gobert's case, the uwVIKOR got values way closer to the ideal condition.

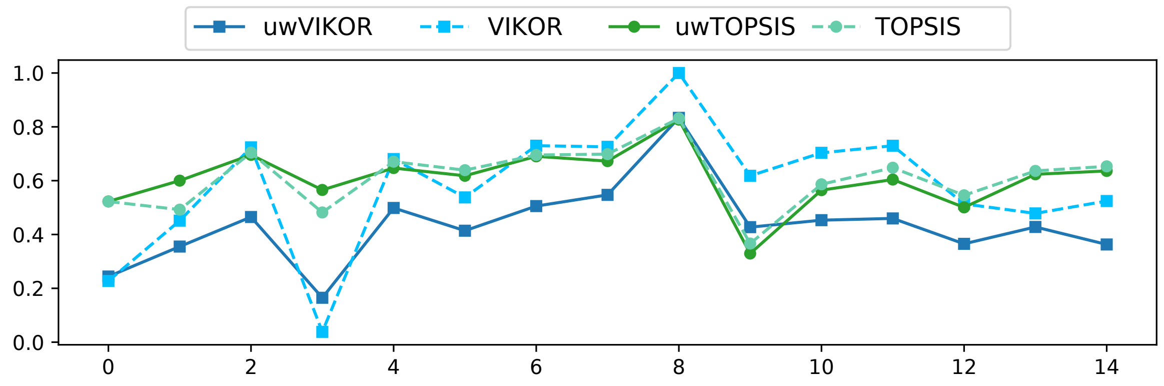


Figure 5.11: Comparison between VIKOR and inverse TOPSIS, both with classic and unweighted approach.

Source: Own elaboration.

In order to complement the results depicted in Fig. 5.11, we have sorted the NBA players according to the scores obtained. In Table 5.24, we can find the cardinal ordination with their respective names per each system.

Table 5.24: Comparison of the ranking systems obtained for the 2021 MVP selection. First row matches with the real positions in the voting scheme. The shortened " ∗" item indicates the player Antetokounmpo.

Actual ranking		**uwVIKOR**	**VIKOR**	**uwTOPSIS**	**TOPSIS**
Jokić	(1st)	Antetok∗	Antetok∗	Gobert	Gobert
Embiid	(2nd)	Jokić	Jokić	Antetok∗	Antetok∗
Curry	(3rd)	Embiid	Embiid	Harden	Embiid
Antetok∗	(4th)	Leonard	James	Jokić	Jokić
Paul	(5th)	Harden	Harden	Westbrook	Harden
Dončić	(6th)	Dončić	Leonard	Embiid	Westbrook
Lillard	(7th)	Gobert	Dončić	Simmons	James
Randle	(8th)	James	Gobert	Dončić	Dončić
Rose	(9th)	Westbrook	Paul	James	Simmons
Gobert	(10th)	Simmons	Westbrook	Leonard	Leonard
Westbrook	(11th)	Curry	Curry	Paul	Paul
Simmons	(12th)	Paul	Randle	Randle	Lillard
Harden	(13th)	Lillard	Simmons	Lillard	Randle
James	(13th)	Randle	Lillard	Curry	Curry
Leonard	(13th)	Rose	Rose	Rose	Rose

Even though the ranges of values in each method are very different, we can highlight some interesting points. First, every method has returned Derrick Rose as the last one in the ranking. When looking at his stats during the season, it does not seem like a major error. Actually, he only got just a vote (which was a first choice). Indeed, uwVIKOR points Rose as an U-dominant for the 2020-2022 season. Second, each method has a winner clearly stated regarding their methodology. For VIKOR, the champion is Antetokounmpo, but for

TOPSIS Gobert is the chosen one. It also gives us information about the stability of the uwVIKOR algorithm, because Nikola Jokić got the second position in the ranking and he is uwVIKOR-L-dominant as we can see in Fig. 5.11.

5.2.7 *Conclusions*

The unweighted VIKOR technique has shown several advantages with respect to the MCDM approaches. By making use of the compromise solutions, we can solve conflicting decision problems and select alternatives as acceptable as we require. Among many strengths addressed in the paper, we would like to emphasize the following pros of the uwVIKOR:

1. We have removed the problematic step of weighting scheme selection. Then, we can process an immersive analysis of the chosen criteria to give an integral answer to the decision makers.
2. Thanks to the ranking functions developed in § 5.2.4.1, we can transform the $[Q_i^L, Q_i^U]$ interval obtained in the last step of the algorithm. Hence, we can provide compromise solutions by means of the comparison of these scores and the acceptable conditions Step 5a and Step 5b.
3. We have developed an algorithm that relates outranking methods with productive efficiency. The Q_i^L score allows us to ascertain whether an alternative can get the best score for a given set of weights.
4. By means of the optimal weights (W_i^L and W_i^U) and their evaluation via w-centrality and joint centrality indicators, we can determine when a weighting scheme is biased or if such weights just suit some alternatives.
5. Even though the computational complexity attached to the optimization problem is elevated, the code available at our GitHub repository makes the implementation of the uwVIKOR straightforward (the code of two sample scripts can be seen in Appendix A.3).

It would be unfair to finish this paper omitting some of the counterparts presented in our methodology. Unlike classical methodology, we may emphasize the following cons:

1. Although we have removed the introduction of a priori weights, we still need to establish their boundaries. It is clear that the unweighted scheme is more flexible, however, the relative importance now is addressed via the (l_j, u_j) pairs. Moreover, we have to pay attention to such intervals in order to prevent criteria vanishing or the existence of a dominant criterion.
2. As long as we define weights as variables in Ω, we could obtain saturated points in the boundaries (l_j, u_j). Therefore, it could be considered misconduct by the decision-makers. We still need to formalize a way to evaluate optimal points returned by uwVIKOR. So far, we just have the w-centrality and joint centrality indicators.

3. The concepts of lower and upper dominance depend on how the data is structured and the constraints given by the (l_j, u_j) pairs.

Finally, in regard to the NBA dataset selected, we have assumed some limitations when processing a small number of players. First, it would have been more reliable if we had considered the entire dataset of the NBA. It would not only take into account the results of the league, but we also would add more players to the feasible set. Second, the criterion system just required seven features to rank alternatives. Despite having a complete scheme of basketball variables, we could also include statistics such as turnovers per game, plus-minus (+/-), player's usage %, player impact estimate, or true shooting % among many others. This would give a way more general perspective when making decisions. However, we would like to mention that, the more variables we add, the less impact they individually have on the model.

5.2.8 *Python implementation*

The code utilized to perform the experiments of this case study is presented in § A.3, where five different scripts are given to conduct the data preprocessing, uwVIKOR implementation, uwVIKOR ranking outputs, uwVIKOR-domination, and weights evaluation.

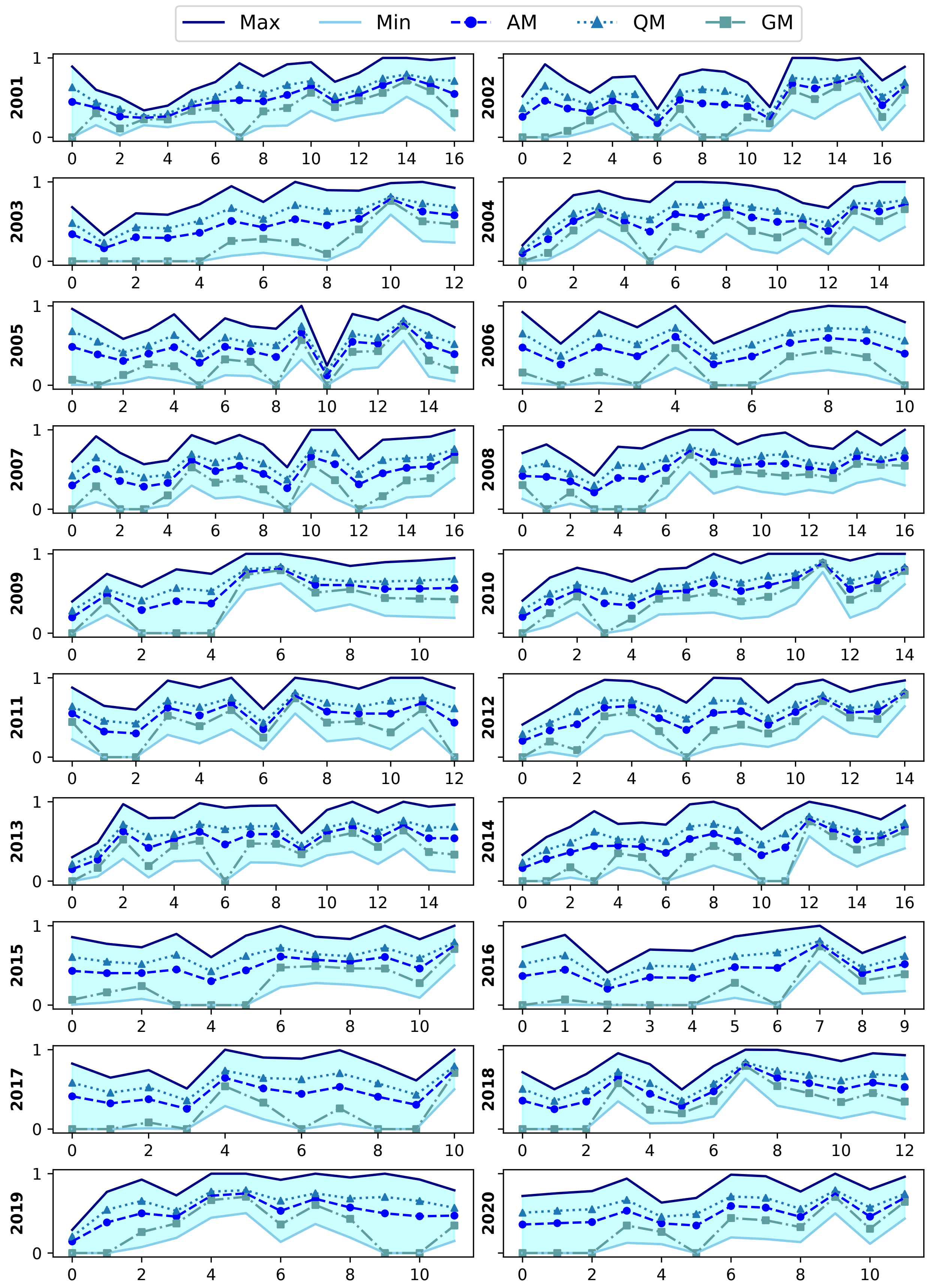


Figure 5.12: Q_i values of the uwVIKOR with arithmetic (AM), quadratic (QM) and geometric (GM) means of the players broken down per year.

Source: Own elaboration.

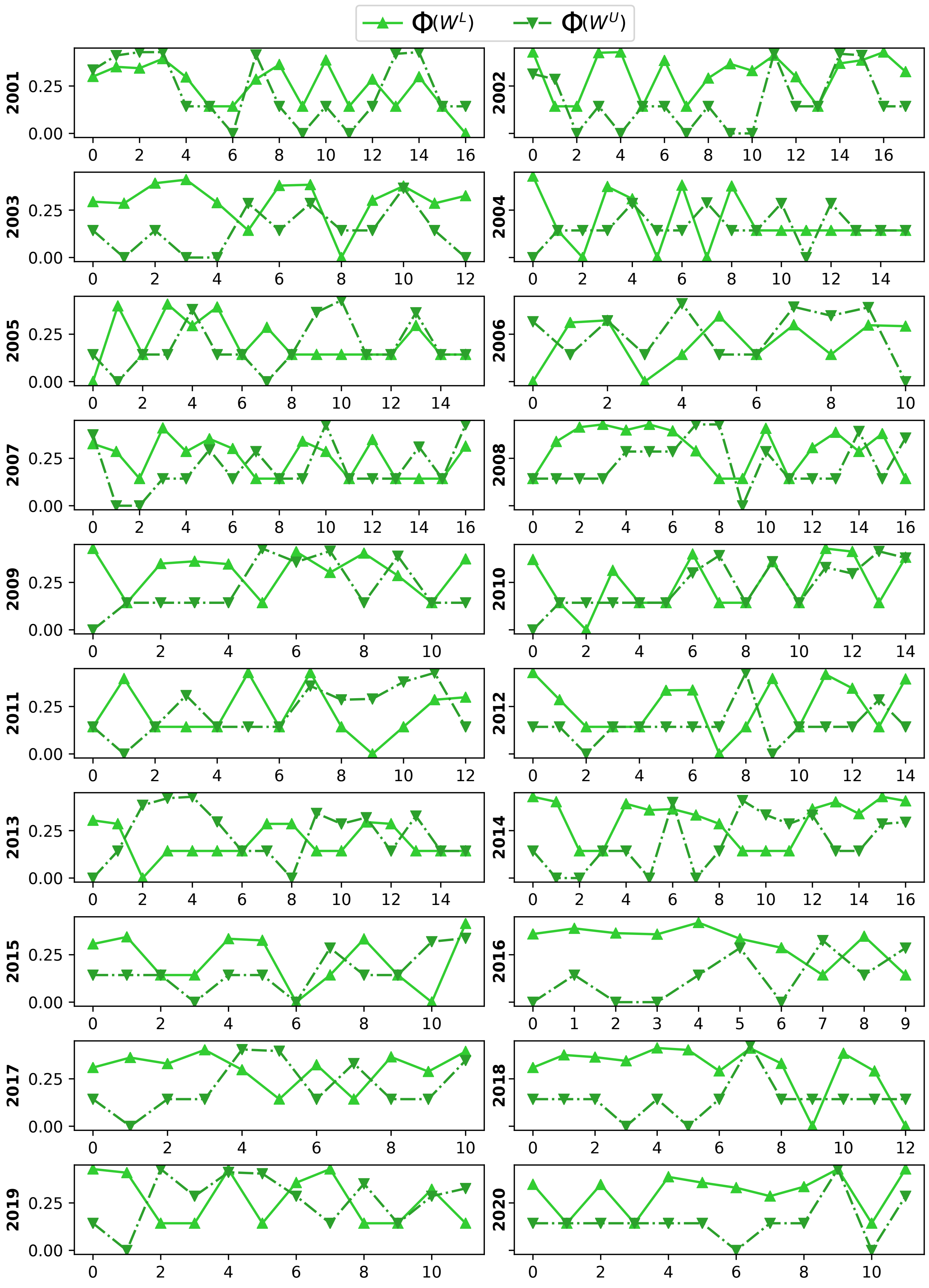


Figure 5.13: Q_i values of the uwVIKOR with arithmetic (AM), quadratic (QM) and geometric (GM) means of the players broken down per year.

Source: Own elaboration.

5.3 SUMMARY

In this chapter, we have addressed two case studies by means of the newly field of unweighted Multiple-Criteria Decision Making. In the first case study § 5.1, we have built composite indicators by means of the uwTOPSIS technique. For this purpose, the (R^L, R^U) scores obtained from the initial situation of the students of the Universidad Industrial de Santader gave us signification about the early stage of the people studied. The final marks of the students in mathematical competencies were utilized as final benchmark. Then, the academic performance indicator (P_A) was defined to analyze the overall performance of the UIS institution. In the second case study § 5.2, a novel unweighted approach named uwVIKOR was presented in order to study the compromise solutions without the need for incorporating any a priori weighting scheme. The applicability of the new technique was proven over two datasets, where not only the ranking systems were displayed but also the analysis of the optimal solutions.

The main objective of this chapter was, in a nutshell, to present the unweighted Multiple-Criteria Decision Analysis approach and show the advantages and profits of its implementation regarding the classical MCDM methods. The flexibility of the unweighted outranking techniques has been remarkable, in which the final ranking system is not the only outcome that decision-makers will take into account when selecting from a set of alternatives. Finally, the publication of the GitHub repositories (López-García, 2021a,b, 2022a) means a breakthrough, since they offer us great advantages with a straightforward implementation.

6 ARTIFICIAL INTELLIGENCE CASE STUDIES

6.1 AI-CASE 1: A PROPOSAL TO COMPARE THE SIMILARITY BETWEEN MUSICAL PRODUCTS. ONE MORE STEP FOR AUTOMATED PLAGIARISM DETECTION?

This case study is based on our published research: López-García A., Martínez-Rodríguez B., Liern V., "A Proposal to Compare the Similarity Between Musical Products. One More Step for Automated Plagiarism Detection?". In International Conference on Mathematics and Computation in Music (pp. 192-204). Springer, Cham, 2021, ISBN: 978-3-031-07015-0, DOI: 10.1007/978-3-031-07015-0_16, (López-García et al., 2022). Hence, all the information or results mentioned in this section have been already addressed in that paper.

6.1.1 *Experiment Setup*

In Martínez-Rodríguez and Liern (2019), the authors presented a measure of similarity between melodies by identifying them with sequences of ordered vectors and using a clustering process based on fuzzy logic. Along the same line, we propose a measure of musical similarity between fragments of digital audio. We present the SpectroMap algorithm (López-García, 2022b) that allows us to detect the local maxima of the audio spectrogram representation, also known as audio fingerprint or constellation map, and we compared the similarity between different maps belonging to different audio excerpts. As a result, it is obtained a value that represents the resemblance between two musical products. This procedure could be used as a non-subjective tool in automatic plagiarism detection. To illustrate this method, three experiments have been carried out comparing different versions of famous pop songs. The results point to the usefulness of the method, although this should be contrasted with an analysis of the human perception of this similarity.

6.1.2 *Introduction*

In past editions of Mathematics and Computation in Music (MCM) we have presented a method to estimate the similarity between different characteristics of symbolic music (melody, rhythm, harmony, tunning) (see Martínez-Rodríguez and Liern, 2017, 2019). In 2019 we presented *Mercury* ®, a computer framework in which techniques from fuzzy clustering were implemented to Computer-Assisted Musical Composition. This saved, to a certain extent, the uncertainty/inaccuracy inherent in any kind of music (Liern, 2005).

Despite the use of software, the approach has always been from the point of view of symbolic music, never from the pure treatment of sound. In this paper, we propose to extend the applicability of the techniques shown in Martínez-Rodríguez and Liern (2017) to comparison of digital audio, based on some attributes of the spectrogram representation.

To achieve our goal, it is necessary to previously process the audio. For this, we have designed an algorithm, which we have called *SpectroMap* (López-García, 2022b), for filtering local maxima (peaks) of the spectrograms. Once this filtering process has been carried out, we obtain the constellation map or fingerprint of the audio fragment (Wang, 2003). Constellation maps can be easily incorporated into the similarity calculations implemented in Mercury, thus obtaining a non-subjective numerical value of similarity between digital audio fragments.

The assessment of the similarity in the conditions described above can be considered as an important element to take into account for the detection of plagiarism. We do not mean to say that the subjective and perceptual part is not important, but if the calculation of similarity between two musical productions is automated, a high value of similarity between them should alert us. In this case, we could also conduct the traditional and pertinent tests to evaluate the existence or not of plagiarism (De Prisco et al., 2017).

We present some examples of similarity estimation between different versions of the same song, using both the spectrogram filtering methods and the similarity calculation methods implemented in Mercury over three different corpora, one for each reference song. The results obtained, beyond giving a ranking of which version is most similar to the original, also provide information about possible compositional interrelationships between the different versions.

6.1.3 *Theoretical background*

6.1.3.1 *Clustering methods*

Clustering methods are aimed to create groups of elements within an initial data set so that the elements included in each group can be considered similar to each other. Unlike the classification methods, in which the elements are assigned with a pre-existing class, in the clustering methods the different classes or subgroups in which the data set is going to be divided have to be defined beforehand the execution of the analysis phase. The clustering procedure will consist of finding a partition of a data set $\mathbf{X}$ that satisfies certain grouping criteria. Following the criteria exposed in Jain and Dubes (1988), given a data set, we will call *element* to each minimum unit of information belonging to it. Each element will have associated a total of q scalar magnitudes called *characteristics* or *attributes*. The term *cluster* (also *group* or *class*) will be used to designate each of the c groupings made from the data set. In *hard* clustering, it is understood that the elements that belong to certain cluster share properties or characteristics with each other and are differentiable from the elements belonging to another cluster. In *fuzzy* clustering this distinction is no longer so clear. The term *centroid* denotes the central point of each of the clusters. The set of n data is

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^q. \tag{6.1}$$

where each $\mathbf{x}_i \in \mathbb{R}^q$, will be a point of q characteristics belonging to a metric space q-dimensional $\mathbb{R}^q$. The index i will designate the i-th element $\mathbf{x}_i$; the number x_{ik} will designate the value of the k-th characteristics of $\mathbf{x}_i$. The total amount of characteristics q is known as the dimensionality of the data set X, and it will have to be a finite and integer number greater than zero.

6.1.3.2 *Hard and soft partitions*

In Bezdek (1981) and Bezdek et al. (1999) we find the necessary theoretical foundations to define the different types of partitioning of a data set. Suppose that $\mathbf{X}$ is a finite set of n elements such that $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ and we want to distribute the elements of the set $\mathbf{X}$ in a number c of subsets $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_c\}$ with $2 \leq c \leq n$. This family of subsets $\{\mathbf{C}_j \colon 1 \leq j \leq c\} \subset \mathbf{X}$ will be a partition of type *hard* if:

$$\bigcup_{j=1}^{c} \mathbf{C}_j = \mathbf{X}, \quad \mathbf{C}_j \cap \mathbf{C}_k = \varnothing, \quad 1 \leq j \neq k \leq c. \tag{6.2}$$

The matrix $\mathbf{U} = [u_{ij}]$ will represent the membership coefficients of each element $\mathbf{x}_i$ to each subset $\mathbf{C}_j$.

6.1.3.3 *k-means clustering*

The *k-means* algorithm, first described by MacQueen (1967), is one of the most widely used clustering methods. It can be classified as a non-hierarchical partitioning method of clustering, in which the data set is divided into a number k of groups, each with a *centroid* called *mean*. This algorithm requires setting the number of clusters k in advance, as well as perform a previous initialization of the groups. The grouping results obtained will depend deterministically both on the number of clusters and on the initialization performed, so to trust the results it will be convenient to repeat the procedure with different initializations.

As we have seen, the operation of the algorithm has two main phases: the initialization phase and the iteration phase. In the first phase, each of the n elements will be randomly assigned to one of the k clusters. Is it possible to formulate the k-means algorithm as an optimization problem of an objective function that will be minimized under given convergence conditions (Gan et al., 2007).

Definition 6.1.1. *Let $\boldsymbol{X} = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^q$ be a data set of n elements. The* k-means *objective function $J_w \colon \boldsymbol{M}_c \times \mathbb{R}^{cq} \to \mathbb{R}^+$ is defined as*

$$J_W(U, \boldsymbol{v}) = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij} (d_{ij})^2. \tag{6.3}$$

where $d_{ij} = d(\mathbf{x}_i, \mathbf{v}_j)$ is a distance function calculated between the element i and the centroid j; $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c) \in \mathbb{R}^{cq}, \mathbf{v}_j \in \mathbb{R}^q \forall j$ is the set of centroids from the clusters; $\mathbf{v}_j$ is the centroid of the cluster $u_j \in U, 1 \leq j \leq c$; and the matrix $\mathbf{U} = [u_{ij}] \in \mathbf{M}_{cp}$ is the belonging matrix to a *hard* partition, accomplishing

$$u_{ij} \in \{0,1\}, \quad \sum_{j=1}^{c} u_{ij} = 1, \quad \text{per each } 1 \leq i \leq n, 1 \leq j \leq c. \tag{6.4}$$

6.1.3.4 *Fuzzy C-Means clustering (FCM)*

The *fuzzy c-means* algorithm supposes a generalization of the functions described in Eq. 6.3, transforming them into an infinite family of functions. The first of these generalizations were made in Dunn (1973), later formulated by Bezdek (1981) as an extension of the well-known k-means algorithm.

Definition 6.1.2. *Let* $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^q$ *be a data set of* n *items. The fuzzy objective function* c-means $J_w \colon \mathbf{M}_{fc} \times \mathbb{R}^{cq} \to \mathbb{R}^+$ *is defined as*

$$J_\lambda(U, V) = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^{\lambda} (d_{ij})^2. \tag{6.5}$$

where $U \in \mathbf{M}_{fc}$ is a fuzzy partition of $\mathbf{X}$, and $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c) \in \mathbb{R}^{cq}, \quad \mathbf{v}_j \in \mathbb{R}^q$ is the set of centroids associated to the clusters $u_j, 1 \leq j \leq c$; and $d_{ij} = d(\mathbf{x}_i, \mathbf{v}_j)$ is any distance function in $\mathbb{R}^q$; u_{ij} is the membership coefficient of the element $\mathbf{x}_i$ to the cluster j; and finally $\lambda \in [1, \infty)$ is the weight exponent, or *fuzziness degree* of the process.

The function originally proposed by Dunn (1973) is obtained by setting $\lambda = 2$ and selecting the Euclidean distance $d(ij) = d_{euc}(ij)$. It was later generalized by Bezdek (1981) into the following family of functions $\{J_\lambda | 1 \leq \lambda < \infty\}$. We can now see that the objective functions have the distance weighted by the membership coefficients u_{ij}. Since $\mathbf{M}_{fc}$ is a fuzzy partition, the coefficients $u_{ij} \in [0, 1]$.

The fuzzy clustering process will be achieved through an iterative optimization of the objective function J_λ, updating in each iteration both the membership coefficients u_{ij} and the centroids $\mathbf{v}_j$ by following the expressions (see Bezdek, 1981):

$$u_{ij} = \left(\sum_{k=1}^{c} \left[\frac{d(\mathbf{x}_i, \mathbf{v}_j)}{d(\mathbf{x}_i, \mathbf{v}_k)} \right]^{\frac{2}{\lambda - 1}} \right)^{-1}, \qquad \mathbf{v}_j = \frac{\sum_{i=1}^{n} u_{ij}^{\lambda} \mathbf{x}_i}{\sum_{i=1}^{n} u_{ij}^{\lambda}}. \tag{6.6}$$

The matrix $U = (u_{ij}), 1 \leq i \leq n, 1 \leq j \leq c$ is now a fuzzy partition of $\mathbf{X}$, built by the membership coefficients u_{ij}. The fuzzy partition verifies

$$\sum_{j=1}^{c} u_{ij} = 1, \quad \text{per each} \quad 1 \leq i \leq n. \tag{6.7}$$

In what follows we will show the implementation of the fuzzy c-means clustering algorithm proposed by Bezdek (1981):

Step 1 Fix a number of clusters m, $2 \leq m < n$. Choose any inner product norm metric for $\mathbb{R}^q$; fix λ, $1 \leq \lambda < \infty$. Initialize $U^{(0)}$.

Step 2 Calculate the fuzzy centroids $\{v_j^{(k)}\}$ with $U^{(k)}$ and expression Eq. 6.6.

Step 3 Update $U^{(k)}$ using expression Eq. 6.6 and $\{v_j^{(k)}\}$.

Step 4 Compare $U^{(k)}$ to $U^{(k+1)}$ using a convenient matrix norm, being $\epsilon \in (0,1)$ and arbitrary termination criterion. If $\| U^{(k+1)} - U^{(k)} \| \leq \epsilon$ then stop, otherwise set $k = k+1$ and return to Step 2.

6.1.3.5 *A dissimilarity based on FCM algorithm*

Definition 6.1.3. *Let $\mathscr{T}^A = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^q$ and $\mathscr{T}^B = \{\mathbf{y_1}, \ldots, \mathbf{y_m}\} \subset \mathbb{R}^q$ be two data sets, where $n > m$. Let $d : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ be a distance function. Let u_{ij} be the final membership coefficients calculated with the FCM algorithm, using data set $\mathscr{T}^A$ as data to be partitioned and $\mathscr{T}^B$ as the initial set of centroids. The average dissimilarity $\mathscr{D}$ from the data set $\mathscr{T}^A$ to the data set $\mathscr{T}^B$ is defined by*

$$\mathscr{D}(\mathscr{T}^A, \mathscr{T}^B) = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij} d(\mathbf{x_i}, \mathbf{y_j}) \; . \tag{6.8}$$

It is noteworthy that dissimilarity $\mathscr{D}$ does not consider the possible natural order that could exist in both data sets, achieving a partition of $\mathscr{T}^A$ without any special weight to the elements whose degree of neighbourhood is stronger.

6.1.4 *Fuzzy Ordered C-Means clustering (FOCM)*

In Martínez-Rodríguez and Liern (2017, 2019) we presented FOCM, an improvement of the FCM algorithm in which the order of both data set and centroids sequences were taken into account during the partition process. Instead of partitioning a data set $\mathbf{X}$ with a given set of c centroids belonging to C categories, let us consider the possibility to implement the partition process by introducing the order of the elements in the fuzzy partition process. For that purpose, let us consider two sequences $\mathscr{S}^A$ and $\mathscr{S}^B$ with a different number of elements. Sequence $\mathscr{S}^A$ will be the ordered data set to be partitioned, and sequence $\mathscr{S}^B$ will represent the initial set of centroids.

In FOCM, the Neighbourhood Functions will provide higher weights of comparison to the pair of elements of the sequences that share closer positions in the order of each sequence. At the same time, they will decrease the contribution to the global dissimilarity to those pairs of elements that are ordinally distant.

The purpose of FOCM is to modify the algorithm FCM so the natural order of both data set sequence $\mathscr{S}^A = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ and centroids sequence $\mathscr{S}^B = \{\mathbf{y_1}, \ldots, \mathbf{y_m}\}$, with, $n < m$, is considered during the partition process.

FOCM algorithm works as follows: for every step in which the fuzzy partition U has been calculated, the coefficients u_{ij} will be multiplied by weight by means of a specific neighbourhood function $f(i,j)$. For accomplishing with de convergence criterion, the matrix should be normalized as follows into $\widetilde{U}$

$$\tilde{u}_{ij} = \frac{u_{ij} f(i,j)}{\sum\limits_{k=1}^{m} u_{ik} f(i,k)}. \tag{6.9}$$

Step 1 Set $\{v_j^{(0)}\} = \{y_j\}$. Let m, n be the number of notes of $\mathscr{S}^B$ and $\mathscr{S}^A$, respectively. Choose any convenient neighbourhood function.

Step 2 Choose any inner product norm metric for $\mathbb{R}^q$, and fix $\lambda \geq 1$. Calculate the initial $\widetilde{U}^{(0)}$ using Eq. 6.6, Eq. 6.9 and $\{v_j^{(0)}\}$.

Step 3 Calculate the fuzzy cluster centers $\{v_j^{(k)}\}$ with $\widetilde{U}^{(k)}$ and Eq. 6.6.

Step 4 Update $\widetilde{U}^{(k)}$ using Eq. 6.6, Eq. 6.9 and $\{v_j^{(k)}\}$.

Step 5 Compare $\widetilde{U}^{(k)}$ to $\widetilde{U}^{(k+1)}$ using a convenient matrix norm; being $\epsilon \in (0,1)$ and arbitrary termination criterion. If $\| \widetilde{U}^{(k+1)} - \widetilde{U}^{(k)} \| \leq \epsilon$ then stop; otherwise set $k = k+1$ and return to Step 3.

There is a big number of possible neighbourhood functions (Gaussian, Triangular, Exponential, Sigmoidal, etc.) (Martínez-Rodríguez and Liern, 2017). In this paper, we have chosen the Gaussian neighbourhood function, i. e.

$$f_G(i,j) = A \exp\left(-\frac{1}{2\sigma^2} \left[i + 1 - \frac{(n-1)\cdot(j-1)}{(m-1)} \right]^2 \right). \tag{6.10}$$

6.1.4.1 *Definition of a dissimilarity based on FOCM clustering*

Using the FOCM algorithm, in Martínez-Rodríguez and Liern (2017) was defined a dissimilarity between any pair of sequences with a different number of elements.

Definition 6.1.4. *Let $\mathscr{S}^A = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^q$ and $\mathscr{S}^B = \{\mathbf{y_1}, \ldots, \mathbf{y_m}\} \subset \mathbb{R}^q$ be two sequences, where $n > m$. Let $d : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ be a distance function. Let u_{ij} be the final membership coefficients calculated with the FOCM algorithm, using sequence $\mathscr{S}^A$ as data to be partitioned and sequence $\mathscr{S}^B$ as the initial set of centroids. The average dissimilarity $\widetilde{\mathscr{D}}$ from the sequence $\mathscr{S}^A$ to the sequence $\mathscr{S}^B$ is defined by*

$$\widetilde{\mathscr{D}}(\mathscr{S}^A, \mathscr{S}^B) = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} \tilde{u}_{ij} d(\mathbf{x_i}, \mathbf{y_j}). \tag{6.11}$$

In what follows we show the utility of expression Eq. 6.11 for evaluating the dissimilarity between songs or music compositions.

6.1.5 *A comparison of musical products based on FOCM*

Establishing an objective measurement for calculating the dissimilarity between musical products like pop, rock songs, or classical music compositions, can be very useful as a tool for automatic plagiarism detection. Our approach for comparing two musical products will consist of: selecting the digital audio excerpts to be compared; extracting the constellation map (proposed in Wang, 2003) from the spectrogram of each excerpt; calculating the FOCM dissimilarity between constellation maps of both excerpts, with Eq. 6.11, taking into account that a constellation map is a sequence of points formed by time and frequency.

6.1.5.1 *Fast Fourier Transform (FFT) process*

With the aim of implementing a fingerprint extraction for a given musical signal X_t, we have designed an algorithm that computes a global peak detection over the spectrogram associated to give us its constellation map. Let N_{FFT} and N_O be the length of the Fast Fourier Transform (FFT) window and the number of elements to overlap between segments respectively, we first compute the spectrogram of the signal (S_{tfa}), by using the Hamming window, in order to get the (time, frequency, amplitude) vectors by considering these two parameters. Such representation contains the amplitude spatial information to analyze. Our engine search determines whether a time-frequency point can be considered locally relevant according to its neighbourhood. Then, the detection is processed regarding a required band. Let $\{T_i\}_{i=1}^{n}$ and $\{F_j\}_{j=1}^{m}$ be the time and frequency bands of the spectrogram with the amplitude of the event, we can reformulate the spectrogram $S_{tfa} = (T_i)_{i=1}^{n} = (F_j)_{j=1}^{m}$ as its rows and columns representations.

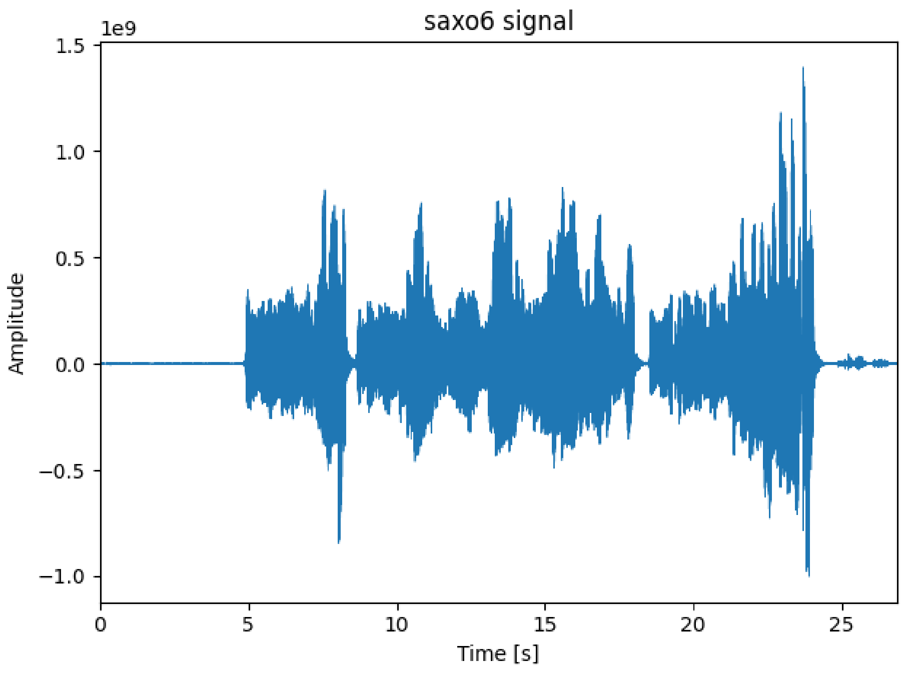


Figure 6.1: Example of waveform from an excerpt of a pop song.
Source: Own elaboration.

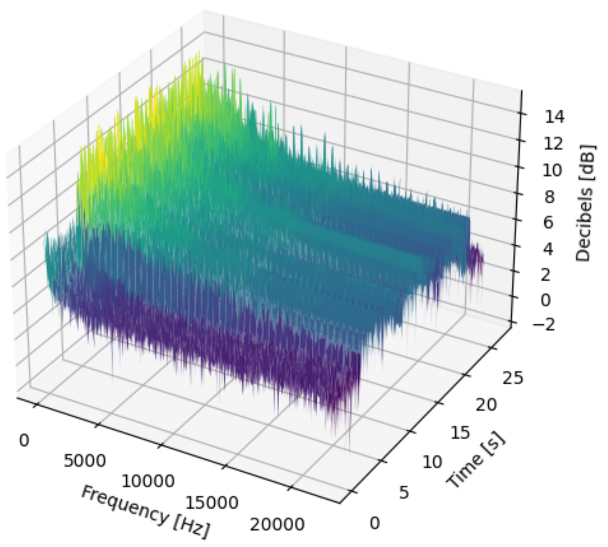


Figure 6.2: Spectrogram visualization of the previous excerpt.
Source: Own elaboration.

6.1.5.2 *Peak detection Algorithm*

As part of the engine search, we define two windows $\phi_T^{d_T}$ and $\phi_F^{d_F}$ to process the local pair-wise comparisons with a respective length of d_T and d_F, whose functionality is to extract a number of elements of the band and return the local maximum. We can describe the time-band window mechanism with a length of $0 < d_T \leq n$ and structure $T_i = (T_i^1, \ldots, T_i^n)$ as

$$\phi_T^{d_T}(T_i) = \left(\max\{T_i^k, \ldots, T_i^{k+d_T}\}\right)_{1 \leq k \leq n-d_T-1}, 1 \leq i \leq n. \tag{6.12}$$

When we group all the values we drop those elements that have an equal index to avoid duplicates. We can group the window of each band to create the set:

$$\Phi_T^{d_T} = \{\phi_T^{d_T}(T_i)\}_{i=1}^n. \tag{6.13}$$

This way, we get the topologically prominent elements per each feature vector. With Eq. 6.12, it is easy to note that even though there are $n - d_T - 1$ matches, the window $\phi_T^{d_T}(T_i)$ may contain a smaller number of elements whenever $d_T > 2$. Depending of how restrictive we need to be, we can proceed with just one of the bands or combine them to create a more stringent search since it is returned only if the peaks that are prominent in both directions. Finally, the algorithm merges all the band-dependent peaks (Eq. 6.13) to give us the total number of spatial points that determines the audio fingerprint. Our engine search, *SpectroMap*, processes audio signals in order to return an output file with the (time, frequency, amplitude) peaks detected. A graphical interpretation of the algorithm is depicted in Fig. 6.3

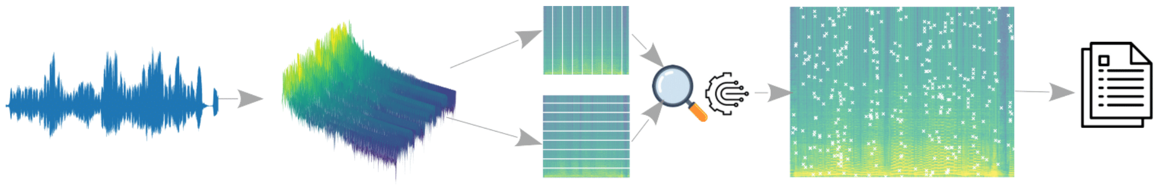


Figure 6.3: Flowchart of the SpectroMap algorithm from the audio input process to the final audio fingerprint output. The search engine is based on the illustrated time-frequency bars.

Source: Own elaboration.

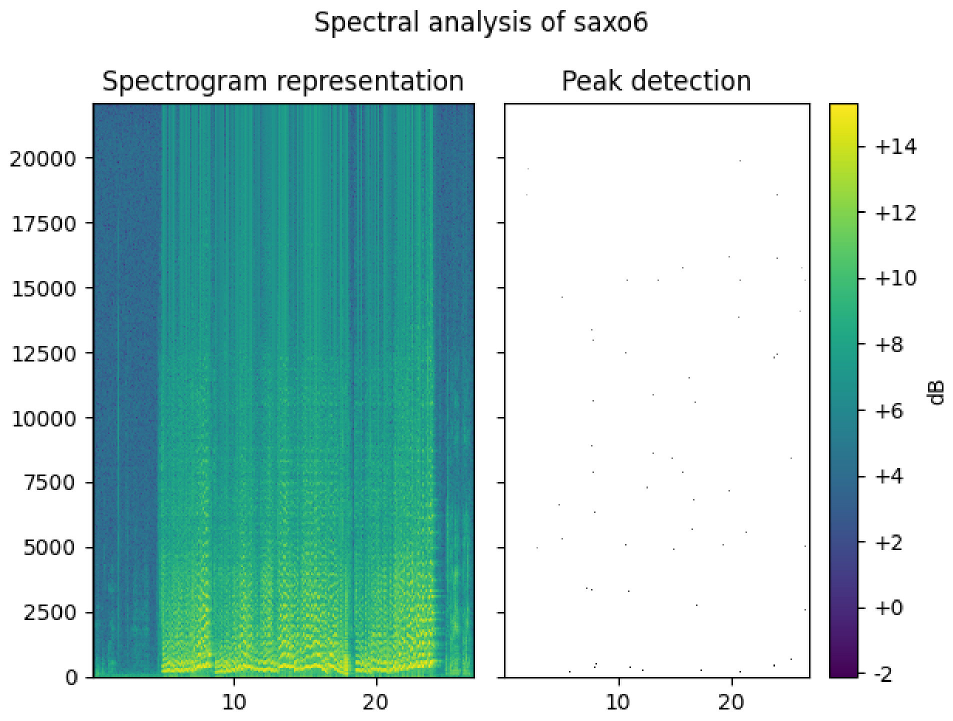


Figure 6.4: Spectrogram (left picture) and the result of peak detection algorithm in time-frequency coordinates represented as pixels (right picture).

Source: Own elaboration.

The algorithm works in these steps:

Step 1 Decide the window to use and set the parameters N_{FFT} and N_O.

Step 2 Read the audio file to get its amplitude vector and its sample rate.

Step 3 Compute the spectrogram through the associated Fourier transformations.

Step 4 Set a fixed window length (d_T, d_F or both) for the pairwise comparisons.

Step 5 Choose the settings to proceed with the peak detection over a selected band or a combination of both.

Step 6 Create an identification matrix that consists of a binary matrix with the same shape as the spectrogram with the position of the highlighted prominences.

Step 7 Extract such elements and create a file with the (time, frequency, amplitude) vectors.

Regarding step Step 5, the authors highly recommend selecting both bands to perform the peak detection since the output is more filtered and spatially consistent. For the remainder steps, the choice is a personal decision that depends on the scope of the research.

6.1.5.3 *Constellation Map*

As it was previously explained, the constellation map is obtained by means of the filtering of local maximum (peak detection) using the algorithm SpectroMap. The sequence of peaks is created by sorting each peak by its appearance time.

Definition 6.1.5. *A Constellation Map is the sequence $\mathscr{M}^A = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^2$ where each $\mathbf{x}_i \in \mathbb{R}^2$ is an observable defined by 2 features: time and frequency. Each element has been sorted by its appearance time.*

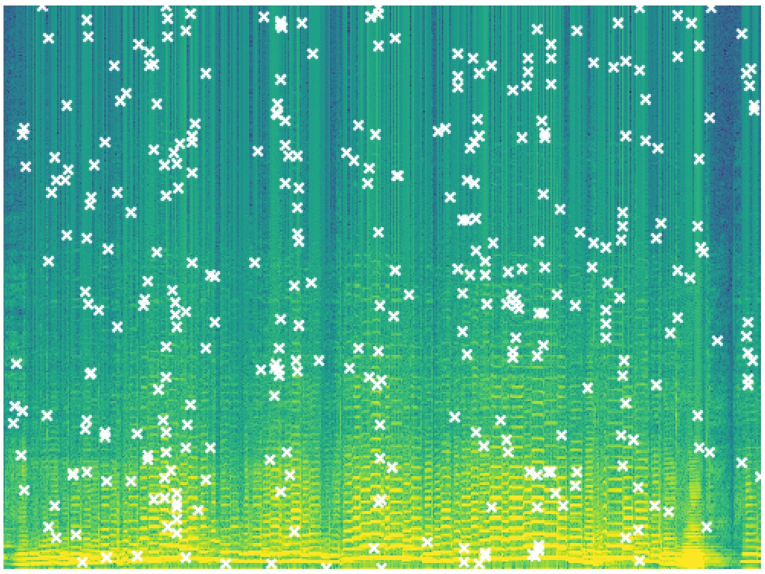


Figure 6.5: Example of constellation map generated (white "*x*" markers) from an audio excerpt. Source: Own elaboration.

6.1.5.4 *Calculation of the dissimilarity based on FOCM*

Using the FOCM algorithm, we can define a dissimilarity between any pair of constellation maps.

Definition 6.1.6. *Let $\mathscr{M}^A = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^2$ and $\mathscr{M}^B = \{\mathbf{y_1}, \ldots, \mathbf{y_m}\} \subset \mathbb{R}^2$ be two constellation maps, where $n > m$. Let $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ be a distance function. Let u_{ij} be the final membership coefficients calculated with the FOCM algorithm, using constellation map $\mathscr{M}^A$ as data to be partitioned and constellation map $\mathscr{M}^B$ as the initial set of centroids. The average dissimilarity $\widetilde{\mathscr{D}}$ between this two constellation is defined by*

$$\widetilde{\mathscr{D}}(\mathscr{M}^A, \mathscr{M}^B) = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} \tilde{u}_{ij} \cdot d(\mathbf{x_i}, \mathbf{y_j}). \tag{6.14}$$

The expression Eq. 6.14 allows us to evaluate the musical plagiarism between any two given excerpts of digital audio.

6.1.6 *Experiments*

To illustrate the applicability of this method, we have designed three experiments to estimate the similarity between different versions of three different pop songs. We have chosen the songs: *Someone Like You*, by Adele; *When I was your man*, by Bruno Mars; *All of me*, by John Legend. This selection is convenient for creating the three different corpora, since there are numerous and different covers available on YouTube. The videos have been downloaded, and the digital audio has been extracted in wav format at 44.100Hz and 16 bits, selecting the same fragment of the song. With this excerpts we have created three experimental corpora. For each corpora we will calculate the similarities using the method explained in the previous section: applying the SpectroMap algorithm and Eq. 6.14. In Table 6.4 are shown the audio sources used in each experiment. For each corpora, we will compare the different versions with each other and with the original, in order to sort them from greater to lesser similarity. The results obtained are shown in Tables 6.1, 6.2 and 6.3.

For experiment No.1, according to the data displayed in Table 6.1, the closest resemblance to the official one is the version by the artist jordan (8.21). However, the *leo*, *imy2* and *masha* versions are more similar to each other than to the official version. This fact could indicate a notable influence between these three artists. The version farthest from the official one is that of *leo* (13.62). Once this result is obtained, we listen to the version and verify

Table 6.1: Dissimilarities calculated between different covers from 1^{st} experiment.

Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.
leo	imy2	5.405	jordan	brit	9.693	oficial	brit	10.270	leo	oficial	13.620
leo	masha	5.416	oficial	masha	9.698	imy2	brit	10.455	leo	jordan	14.392
imy2	masha	5.603	brit	jordan	9.778	oficial	imy2	12.759	nursera	brit	14.440
masha	imy2	5.658	brit	masha	9.788	jordan	imy2	13.568	nursera	masha	17.487
oficial	jordan	8.212	jordan	nursera	9.831	oficial	leo	13.586	nursera	imy2	21.603
oficial	nursera	9.680	leo	brit	10.998	imy2	jordan	13.608	nursera	leo	22.847

Table 6.2: Dissimilarities calculated between different covers from 2^{nd} experiment.

Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.
smith	bml	1.018	bmo	scaccia	1.326	smith	imy2	1.427	smith	scaccia	1.571
stewart	smith	1.262	scaccia	bml	1.335	bml	imy2	1.430	imy2	stewart	1.615
imy2	bml	1.264	imy2	bmo	1.340	bml	bmo	1.466	bmo	smith	1.651
bmo	bml	1.311	imy2	smith	1.398	bml	scaccia	1.495	scaccia	stewart	1.766
imy2	scaccia	1.324	stewart	bml	1.425	scaccia	smith	1.554	stewart	bmo	2.042

Table 6.3: Dissimilarities calculated between different covers from 3^{rd} experiment.

Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.
scaccia	jll	1.935	jlo	jll	2.191	jll	hoying	2.309	hoying	jlo	2.832
jll	scaccia	1.989	leroy	hoying	2.210	jll	leroy	2.316	leroy	jlo	2.917
scaccia	jlo	2.099	scaccia	hoying	2.211	scaccia	zogbi	2.662	zogbi	hoying	2.969
jlo	scaccia	2.137	hoying	scaccia	2.270	zogbi	scaccia	2.686	zogbi	leroy	3.196
scaccia	leroy	2.142	leroy	jll	2.273	jlo	zogbi	2.722	zogbi	leroy	3.196

that the artist has made a version in *rock* style of Adele's theme, effectively far removed in perceptual terms from the original version. The furthest versions are those of *nursera* and *leo* (22.85). Again, if we listen to both versions, the auditory difference is evident, since both versions represent antagonistic musical styles.

The results for experiment No.2 are shown in Table 6.2. The closest resemblance to the official one is the live version by the same artist John Legend (1.311), due to the big similarity of tempi between two versions. The versions of artist *smith* and *imy2* are more similar to the live version. The artist closest to the official version is *scaccia* (1.326). The version farthest from the official one is of *stewart* (13.62).

In experiment No.3 in Table 6.3. The closest resemblance to the official song is the cover by artist *scaccia* (2.099), which is also the closest to the live version (1.935). The farthest version from the official one is of *leroy* (2.927).

Table 6.4: Audio sources used in the experiments.

Someone like you	**When I was your man**	**All of me**
Adele official (official)	Bruno Mars official (bmo)	John Legend official (jlo)
Adele Live (britawards)	Bruno Mars Live (bml)	John Legend Live (jll)
Angelina Jordan (jordan)	Alexander Stewart (stewart)	Leroy Sanchez (leroy)
Nursera Yener (nursera)	Sam Smith (smith)	Luciana Zogbi (zogbi)
Masha (masha)	imy2 (imy2)	Stephen Scaccia (scaccia)
imy2 (imy2)	Stephen Scaccia (scaccia)	Scott Hoying (hoying)
Leo Moracchioli (leo)		

6.1.7 *Conclusions*

The fuzzy logic-based procedures that were implemented for computer-assisted music composition in Mercury software can be used for automatic assessment of music plagiarism from digital audio files.

The Internet and social networks offer an excellent platform for the dissemination of musical content. However, plagiarism detection requires automatic tools that allow quick and effectively discriminate of those versions that may be suspicious in terms of their resemblance to others. Beyond the legal and ethical aspects, this resemblance can be useful for the performers or authors themselves, who can discover their own and other influences in other artists.

6.1.8 *Python implementation*

Throughout the section, we have discussed the use of SpectroMap as a manner to extract audio fingerprinting in music-related tasks. However, it can be utilized in other experiments within and outside the acoustics field. In § B.1 are presented two scripts to execute the SpectroMap algorithm for both raw audio signals and audio spectrograms.

6.1.9 *Extra section: Explicit URL link to the dataset sources*

As we said at the beginning of the experiments § 6.1.6, all the audio excerpts are available on YouTube. Then, any experiment could be reproduced so as to attach more audio samples or compare longer excerpts. For an easy search of the clips utilized, Table 6.5 contains the explicit URLs for every song.

Table 6.5: Explicit YouTube URL source of the audio sources used in the experiments divided by the different experiments.

	Audio reference	**YouTube URL address**
Someone like you	Adele official (official)	https://www.youtube.com/watch?v=hLQl3WQQoQ0
	Adele Live (britawards)	https://www.youtube.com/watch?v=qemWRToNYJY
	Angelina Jordan (jordan)	https://www.youtube.com/watch?v=nU9TA70fXro
	Nursera Yener (nursera)	https://www.youtube.com/watch?v=Z9iylN-IiUA
	Masha (masha)	https://www.youtube.com/watch?v=0EwSEsSvxGY
	imy2 (imy2)	https://www.youtube.com/watch?v=qIuPgPyTNKE
	Leo Moracchioli (leo)	https://www.youtube.com/watch?v=pkbbd3fhcMw
When I was your man	Bruno Mars official (bmo)	https://www.youtube.com/watch?v=ekzHIouo8Q4
	Bruno Mars Live (bml)	https://www.youtube.com/watch?v=gY4GZgZK9H0
	Alexander Stewart (stewart)	https://www.youtube.com/watch?v=j_d3gq5JCAc
	Sam Smith (smith)	https://www.youtube.com/watch?v=_ZaLIiV7c7Y
	imy2 (imy2)	https://www.youtube.com/watch?v=uBh_7PBy8cg
	Stephen Scaccia (scaccia)	https://www.youtube.com/watch?v=Nhm0MHQKYDY
All of me	John Legend official (jlo)	https://www.youtube.com/watch?v=450p7goxZqg
	John Legend Live (jll)	https://www.youtube.com/watch?v=s18cJqrBIOk
	Leroy Sanchez (leroy)	https://www.youtube.com/watch?v=Im6_k-UMJeo
	Luciana Zogbi (zogbi)	https://www.youtube.com/watch?v=39_OmBO9jVg
	Stephen Scaccia (scaccia)	https://www.youtube.com/watch?v=07McLNDuffo
	Scott Hoying (hoying)	https://www.youtube.com/watch?v=d0GR60bul4M

6.2 AI-CASE 2: MULTIVARIATE TIME SERIES PREDICTION BASED ON STOCK MARKET AND SENTIMENT ANALYSIS REGRESSORS

6.2.1 *Background*

6.2.1.1 *Financial time series/Stock market*

At the beginning of the 19th century, the study of time series was mainly marked by the approach of deterministic models. In 1927, this notion was reformulated by considering stochastic processes for prediction (Yule, 1927), which led to the definition of autoregressive (AR) and moving average (MA) models, whose combination is the autoregressive moving average (ARMA) model. Thanks to advances in the methodology of Box and Jenkins (1970), integrated autoregressive moving average (ARIMA) models, a powerful set of methods for both univariate and multivariate forecasting, were introduced.

In the second half of the last century, exponential smoothing methods constituted the state-of-the-art set of techniques for extrapolating univariate time series with a single source of error (white noise, Gooijer and Hyndman, 2006). These methods studied the classification of the trend and seasonality components of the series as additive or multiplicative, i.e., as linear or nonlinear. The Holt-Winter methods should be highlighted as one of the best-known and applied at the time. Over the years, there were many improvements in terms of model performance. For example, some researchers tackled the problem of prediction with missing values and prediction with additional constraints (Gooijer and Hyndman, 2006). Another relevant contribution in this field was the proposal of normalization schemes to eliminate bias (Archibald and Koehler, 2003).

At the turn of the century, although ARIMA models were still very popular, and the default technique for time series prediction, some researchers realized that linear statistics was not a suitable approach, as it required that time series had to be generated from linear events (Zhang et al., 1998). Then, several nonlinear models were designed, such as threshold autoregressive (Tong and Lim, 1980), autoregressive conditional heteroskedasticity model (ARCH, Engle, 1982), the generalized autoregressive conditional heteroskedasticity (GARCH, Bollerslev, 1986), and the Markov-switching models (Hamilton, 1989).

6.2.1.2 *Machine learning*

In the field of financial analysis, there exists vast literature related to the prediction of stock features. Over the years, researchers have attempted to approach such a task with the use of classical time series analysis, as described in § 6.2.1.1. Although these methods can be very useful when describing linear relationships between explanatory variables, the complexity of the stock market makes these models unable to predict nonlinear behavior. Recently, the methodology related to this problem has changed considerably. This is due to the rise of machine learning algorithms and modern statistical techniques, which have become state-of-the-art approaches. By definition, neural networks are nonlinear, making them more suitable for interpreting the structure of stock market data. Due to the uncertainty and chaos associated with the market and the speed of response required, machine learning can be considered the best alternative to take, as it has the ability to correctly adapt to the data and describe slight correlations between variables, allowing decision-makers to make more far-reaching decisions.

In this regard, both supervised and unsupervised techniques have been applied with great success. In particular, for supervised learning, we can find literature related to the application of support vector machine (SVM, Kim, 2003), Random Forest (RF, Krauss et al., 2017), Naive-Bayes (NB), multilayer perceptron (MLP, Pasero et al., 2010), convolutional neural networks (CNN, Selvin et al., 2017), or recurrent neural networks (RNN, Althelaya et al., 2018; Siami-Namini et al., 2019; Zhang and Xiao, 2000).

Within the area of artificial neural networks (ANNs), it has been shown that they perform better for prediction than classical statistical methods. Especially for longer data sequences, either with more backward steps to analyze or with a longer horizon to predict. In fact, they are better in terms of consistency, as they can obtain more inflection points (Kohzadi et al., 1996). In addition, ANNs are better at tasks involving nonstationary data (Lachtermacher and Fuller, 1995).

6.2.1.3 *Sentiment analysis*

Companies are heavily influenced by comments and rumors spread by both internal and external sources. In fact, the trust and reputation of a company are one of the most important decision factors to take into account in finance. In recent times, everyone can create content and disseminate it through social networks, which generates a scenario with massive information that lacks objectivity. Social media content can play a key role in daily trading, so both companies and public entities have to deal with fake news, hoaxes, and propaganda that could tarnish their public image. In general, this is not a trivial task, as the background is closely linked to monetary interests. Even if it is not perceived as a pernicious activity, disinformation has a major impact on societies and global economies. It is therefore essential to implement software that analyzes global content in real-time and provides us with a dashboard to monitor it on a regular basis. In this way, these programs can provide investors with a powerful tool that represents the real perception that the public has of an entity. Once we know the external opinion, we can take steps to improve it or maintain the same guidelines to keep a good reputation.

Sentiment analysis is a currently exploited text mining field. It studies the treatment of all sorts of content that have associated subjective opinions or emotions in order to analyze them by making use of Natural Language Processing (NLP). Its target features are mainly words, but some researchers have added emojis or emoticons that complement the text (Eisner et al., 2016; Fernández-Gavilanes et al., 2018). Then, the sentiment related is usually categorized with multi-labels that vary depending on the scope of the research. Mostly, it is labeled as positive or negative as a binary classification. However, there are some researchers that have considered different labels which can also be attached to neutral emotions or feelings.

Sentiment analysis is a classification problem that is commonly approached with two distinguished fields: The first is artificial intelligence algorithms and the second is lexicon and corpus-based models. The features selected are mostly categorized as terms presence/-frequency, parts of speech, opinion words/phrases, and negations (Medhat et al., 2014). Generally, such deployments make use of both big data and advanced computer science techniques to conduct, gather, and operate a large amount of data. In the first set of AI implementations, researchers have approached the task at three stages of granularity: document level, sentence level, and aspect level (Zhang et al., 2018). In this strategy, we can find very simple approaches to deep neural networks to tackle the NLP problem (Mishev et al., 2020). Supervised methods rely on the annotated labels as bag of words, noun phrases, and proper names; so. Unsupervised approaches overcome hand annotation problems by using keyword lists that capture sentiments (Basiri et al., 2021; Medhat et al., 2014). In the second field of lexicon-based models, they utilize a dictionary with types of words (nouns, verbs, adjectives, ...) that have associated scores. It can also contain features such as polarity or strength per each word or phrase. Then, they compute the sentiment per each word throughout the text to give a response as a final score (Khoo and Johnkhan, 2018; Medhat et al., 2014; Taboada et al., 2011). Moreover, we can also find techniques that partially combine both perspectives with a hybrid-based model, for instance, Bahrainian et al. (2014) defines a context-sensitive sentiment lexicon that is self-maintained via transfer learning. Thus, the number of word false matches is reduced and the updated dictionaries are generalized by appending more words, making them more robust.

Regarding finance, there is evidence that online content in social media can severely affect stock prices (Nasseri et al., 2015). Owing to the internationalization that is happening nowadays, break news weighs the returns and the volatility. Although this impact is thought to be caused for last-minute information, it is also triggered by pernicious propaganda against some financial asset or company. NLP in this field is studied from sources such as news in papers (Schumaker and Chen, 2009) and microblogging platforms (Renault, 2020) to modern social networks (Smailović et al., 2013). In contrast with the standard sentiment analysis framework, sometimes it is used labels as bullish (negative) or bearish (positive). Researchers study this concern in order to measure such impact and predict either future sentiment analysis (acceptance/rejection) or market movements (Mishev et al., 2020; Rekabsaz et al., 2017).

6.2.1.4 *COVID-19 scenario*

In December 2019, the first case of SARS-CoV-2 was identified in Wuhan[1], China. Its initials mean Severe Acute Respiratory Syndrome Coronavirus 2. Nowadays, it is still considered an ongoing global pandemic by the WHO due to its high spread and its presence worldwide. Since 2020, governments around the world decided to take action one way or another, becoming one of the major concerns in their agendas. In most countries, a lockdown was applied and lots of restrictions were added to the regulations in order to prevent recurrent spikes of infections. At the end of 2020, the COVID-19 disease took the life of 1.813.188

1 Content available online at The Wall Street Journal website (published on 26/02/2021)

people, but sources like WHO is convinced that this number could raise up to is at least 3 million, which is 1.2 million more deaths than officially reported[2].

As a consequence of this serious problem, most of the main pharmaceutical companies have invested large amounts of money in order to design an appropriate vaccine that could stop such a catastrophe. In particular, AstraZeneca is one of the companies that have led the road to the release of its own vaccine. After lots of efforts, the Oxford-AstraZeneca COVID-19 vaccine, also named AZD1222, was approved on December 30 of 2020 and its first administration was on January 4 of 2021. Despite the fact that it could be considered a turning point for the prevention of COVID-19, it had a social negative impact. During the first month of vaccination, the company registered 31 cases of thrombosis, 9 of them with fatal ending[3]. In social networks, the hype related to the topic became a trending topic alarmingly rapid. A major concern related to internet content is that propaganda is not as easy to spot, because modern techniques are able to create pseudoperiodistic style that can confuse anybody. In turn, there were lots of hoaxes and fake news that exacerbated the cases and allegedly associated its product with death. On March 31 of 2021, the vaccine was no longer administered to people under the age of 60 in some countries as a consequence of a failure in the roll-out[4]. The fake news and social pressure were not the main facts of the end of the vaccination row, notwithstanding it created social uncertainty that got to be clarified by AstraZeneca[5] and the European Union's medicines agency[6].

As time goes by, the spread of the virus was unstoppable as well as the amount of misinformation posted in internet[7] against AstraZeneca. Whatever the topic is, these kinds of arguments lack basis, rationally, and approval so it needs a priori verification from main sources. It is important to bear in mind that not only the ones to create content are solely responsible but also the ones that disseminate it take part in the problem by intensifying it. As a consequence, the anti-vaccine movement played a significant role in online platforms. The broadcast of their arguments left no one indifferent. That is a common case in social media where the massive propagation creates a stage way more severe than it really is because of amplification created for the super-spreaders[8]. Such comments can contain from little misunderstandings[9] to blatant lies[10], anyway, the response was imminent. Regardless of the degree of reliability of their posts, most of the platforms were filled up with users that warmed about dreadful outcomes because of COVID-19 vaccination. Even though it might seem a minor concern, the high spread of content on digital platforms aroused the interest of undecided people that have not resolved their doubts yet. Therefore, it is essential to provide an additional tool that can sort out this concern by verifying online content and ensuring a democratic free choice for everybody.

2 Content available online at World Health Organisation website (published on 01/03/2021)
3 Content available online at European Medicines Agency website (published on 07/04/2021)
4 Content available online at BBC website (published on 03/09/2021)
5 Content available online at AstraZeneca website (published on 14/03/2021)
6 Content available online at European Medicines Agency website (First published on 18/02/2021 and last updated on 25/05/2022)
7 Content available online at SOMA Disinformatory website (published on 31/03/2021)
8 Content available online at BBC website (last accessed on 27/06/2022)
9 Content available online at Reuters website (published on 14/12/2020)
10 Content available online at Reuters website (published on 25/05/2021)

So far, lots of researchers have analyzed this scenario with a perspective of sentiment analysis in order to capture and describe the underlying emotions and feelings posted on social media (Nemes and Kiss, 2021). As a practical application of the problem, we can find authors that put their efforts into examining the crisis communication among population (de las Heras-Pedrosa et al., 2020), authors that study the danger of propaganda and negative tagged content in digital platforms and its high spread (Chakraborty et al., 2020), and authors that contrast the impact of COVID-19 with past outbreak-related incidents of infectious diseases (Alamoodi et al., 2021).

6.2.2 *Methodology*

6.2.2.1 *Data*

With the aim of studying the situation of AstraZeneca within the stock market, we have been tracking information about the company from 2018 to the second semester of 2021. As we decided to analyze the company via an intraday market in order to design a prediction model, the features selected reported parameters extracted day-by-day. Such information contains data directly related to the stock market and the Twitter platform. In both cases, we have filtered the results in order to have the features daily distributed. Then, we can define the variables of our problem as dependent on the market (endogenous) or independent of it (exogenous), in which we have two groups depending on the source from that they have been obtained.

1. Stock market: Close, Open, Low, High, Net, %Chg, Volume, Turnover-GBP, and Flow.
2. Twitter platform: Accumulated sentiment analysis.

6.2.2.2 *Feature selection*

On the one hand, the stock market-related features have been downloaded from Refinitiv, in which we have price information (Close, Open, Low, and High), price variation (Net, %Chg, and Flow) and global position (Turnover and Volume). A financial candlestick representation is depicted in Fig. 6.7. As far as trading is concerned, they can be utilized as a set of variables that describe the financial behavior of the company since we know the full range of the event and the day-by-day differences. Regarding the minimum and maximum values of the company, during the event studied the pharmaceutical enterprise ranged between 4545 and 10120, in which their volume as a company has been varying from 309.945 to 8809.096 with an interval of change in percentage of $[-9, 8]$. We are aware that financial business requires further details about the company's situation, a selective study of the complementary stock exchange, an in-depth analysis of the stage of the countries in which the pharmaceutical operates, and also get extra data about other companies in the same market that directly compete with our target enterprise. Nevertheless, the selected subset of variables contains enough market characteristics to obtain great results as will be shown in § 6.2.8.

On the other hand, the collected tweets have been filtered and selected by using Refinitiv, in which the sentiment analysis associated is also extracted simultaneously. In this case, when we refer to sentiment analysis we mean the acceptance or disapproval attached to some message in regard to AstraZeneca. Then it can be labeled as a positive, neutral, or negative sentiment analysis. As we wanted to avoid non-relevant information, we only downloaded content from relevant sources, that is to say, the program skipped messages with low spread. As a consequence, we have only considered messages which have been either significant or highly retweeted. Since our purpose is to match the daily rate, we have accumulated the sentiments of the tweets per day and then we have transformed the labels from $\{positive, neutral, negative\}$ to $\{1, 0, -1\}$. Then, we merged the result as a sum of each list per day as an accumulated indicator that showed the overall review of the pharmaceutical company in terms of subjective emotions. During the event, the higher value was registered on January 11 of 2021 as a positive sentiment of 7, and the lower value was registered on March 15 of 2021 as a surprising negative peak of -37 reflecting how disrupted the pharmaceutical's image was on Twitter.

A more precise representation of the procedure conducted for data and feature extraction is depicted in Fig. 6.6, in which the whole process has been graphically represented as a flowchart.

Altogether, we have computed 10 different variables over 1250 days. For a better understanding, Fig. 6.8 depicts the temporal evolution of the set of regressors above mentioned.

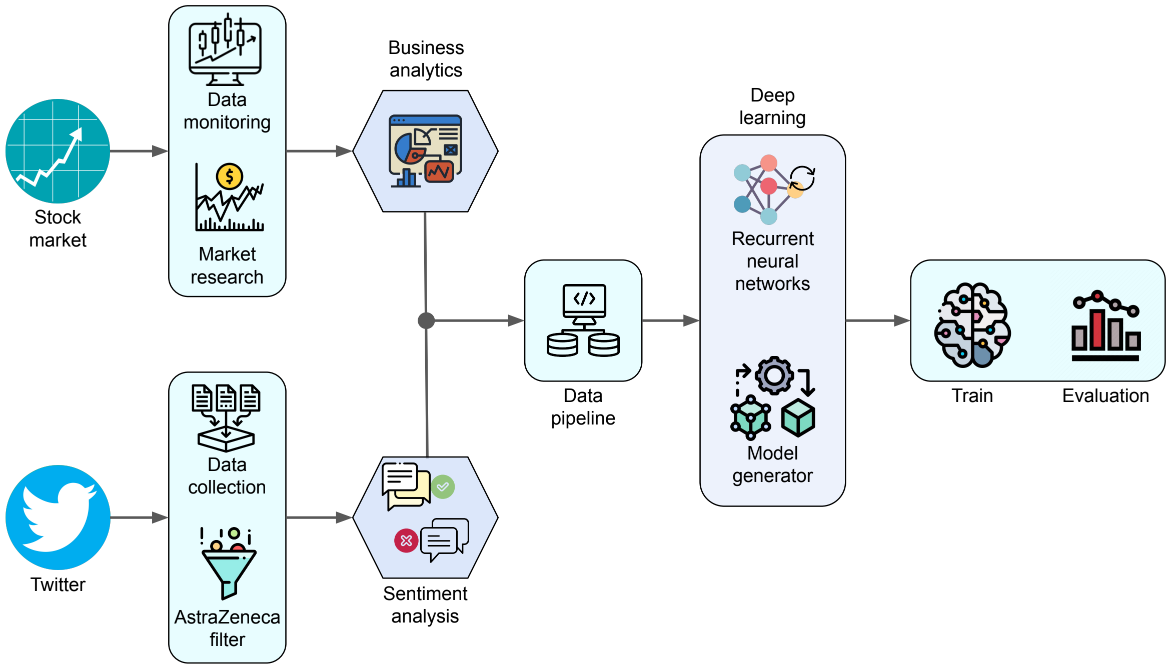


Figure 6.6: Flowchart of the procedure followed in our paper, which it is included the data preparation, feature extraction, model implementation, and evaluation.
Source: Own elaboration.

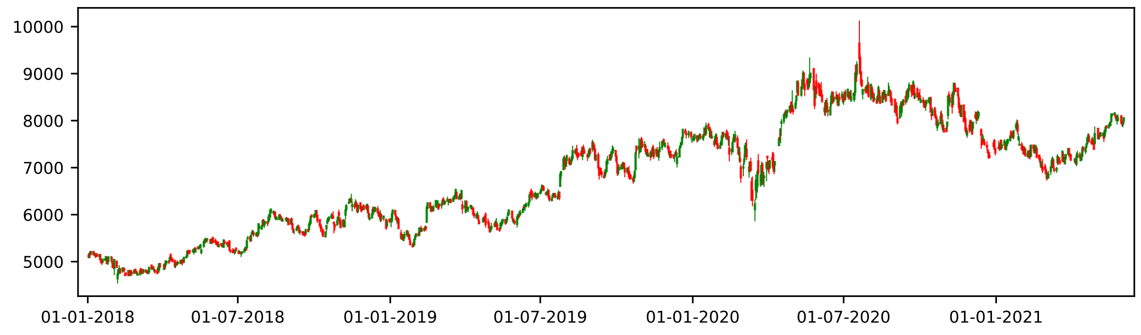


Figure 6.7: Financial candlestick chart of AstraZeneca of the entire dataset.
Source: Own elaboration.

Table 6.6: Sample with the first four days of June of 2021 with the different stock market features.

Date	Close	Open	Low	High	Net	%Chg	Volume	Turnover	Flow
2021-06-01	7,957	8,095	7,946	8,107	-89	-0.011	2,532,475	$2.03{\cdot}10^8$	$2.03{\cdot}10^8$
2021-06-02	7,932	7,935	7,885	7,992	-25	-0.003	1,697,118	$1.34{\cdot}10^8$	$6.88{\cdot}10^7$
2021-06-03	7,948	7,932	7,870	7,973	16	0.002	1,309,821	$1.03{\cdot}10^8$	$1.72{\cdot}10^8$
2021-06-04	8,055	7,993	7,932	8,059	107	0.013	841,590	$6.74{\cdot}10^7$	$2.40{\cdot}10^8$

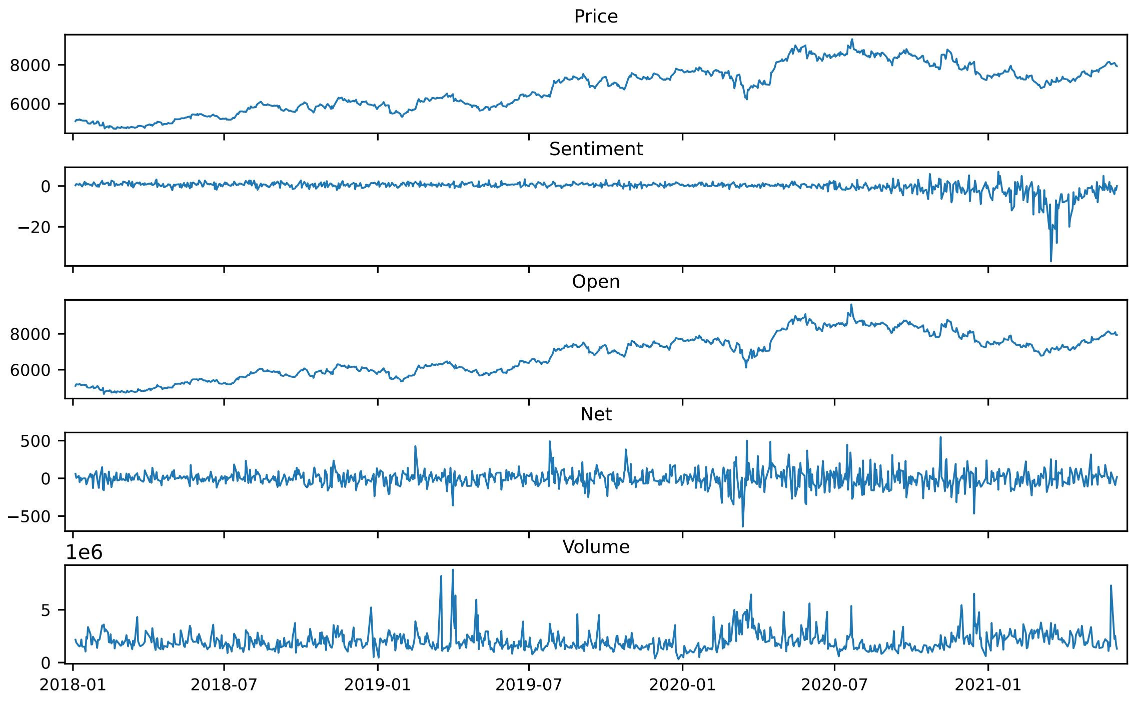


Figure 6.8: Temporal series of the regressors of AstraZeneca stored from 1/1/2018 to 4/6/2021.
Source: Own elaboration.

Tables 6.6 and 6.7 shows excerpts of data with details of the datasets.

Table 6.7: Sample with the first two weeks of March of 2021 with the accumulated sentiment analysis feature in the first two weeks of March of 2021.

Time series at March of 2021														
Day	01	02	03	04	05	06	07	08	09	10	11	12	13	14
Sentiment	-3	-2	-11	-13	-1	-1	-4	-5	-6	-9	-21	-5	-4	-16

6.2.2.3 *Data pre-processing*

Once we have computed the input data, we need to sequence the matrix as concatenated vectors with a fixed size that has associated with a defined output. Here, we have selected a 5 steps back sequence with a unique step forward value as output. The choice of the number of steps back is not trivial, however, we have fixed it as five steps because it can be understood as a complete standard business week. Additionally, after a comparative study of the correlation between the output (y_t) and an output selected from $0 < \tau < N$ steps back $(y_{t-\tau})$, we got sufficient evidence that our lag is adequately selected. Fig. 6.9 depicts the lag correlation for different steps τ.

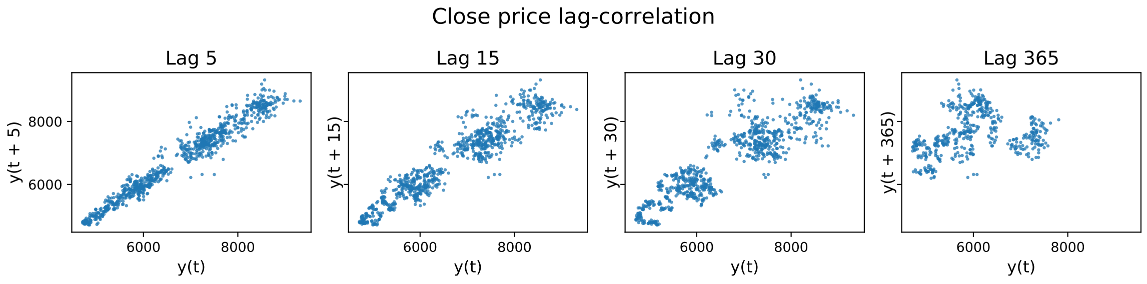


Figure 6.9: Comparative lag correlation for the AstraZeneca close price by considering a different number of steps back τ. It is considered one and two business weeks, a month, and a year as different lags.

Source: Own elaboration.

In regard to the steps forward, we have decided to model a univariate output for time series forecasting. Hence, each variable was sequentialized as $[(x_{t-\tau}, \ldots, x_t), x_{t+1}]$ per each $t \in \{\tau, \ldots, N-1\}$.

In order to standardize the sequences and avoid scale problems, we have computed the statistical typification, also known as a standard scaler or batch normalization, which consists of the quotient of each variable (X_t^k) minus its mean (μ_X^k) out to its standard deviation (σ_X^k).

$$\frac{X_t^k - \mu_X^k}{\sigma_X^k}, \quad \text{being } t \in \{1, \ldots, N\} \text{ (time) and } k \in \{1, \ldots, m\} \text{ (stock feature).} \tag{6.15}$$

6.2.3 *Model architecture*

Regarding the structure of the models, we have generated three kinds of models based on their layer serialization. We have Simple, Stacked, and Bidirectional recurrent models. This procedure was also applied so as to analyze and compare the results with respect to certain measures. For additional comparison purposes, we have computed simple RNNs and LSTM layers, then, we can determine whether the long-term is effective for prediction. Per each model, the architecture is the following:

Simple: Only one recursive layer with unidirectional flow (forward).

Stacked: Multiple recursive concatenated and fully connected. The hidden layer has the same number of neurons in order to perform a more complex process of our series.

Bidirectional: Two-way model with two recursive layers, both of them connected to the same output layer. The main advantage is that the input is processed forward and backward, adding extra information to the learning process.

6.2.4 *Regressor sets*

As we described in § 6.2.2.1, we have two different types of variables classified as endogenous and exogenous. For univariate time series, we just need to consider the number of steps back and forward to conduct the model training and its subsequent forecast. However, when adding regressors to our system a number of variables have to be selected to carry out the due processes. In our case, we split the extracted variables as Sentiment-based (S) and Price-based (P). Then, we have four sets of regressors depending on the addition of them as inputs into the model.

NSNP: Only contains the close price, which leads to univariate time series forecasting.

S: Contains the close price and the daily accumulated sentiment analysis of the company.

P: Contains the close price and the stock features of the company.

SP: Contains the entire set of variables.

In this manner, we can generate 24 distinct models because we have 2 different sorts of layers, 3 types of model architecture, and 4 types of regressor sets.

6.2.5 *Model implementation*

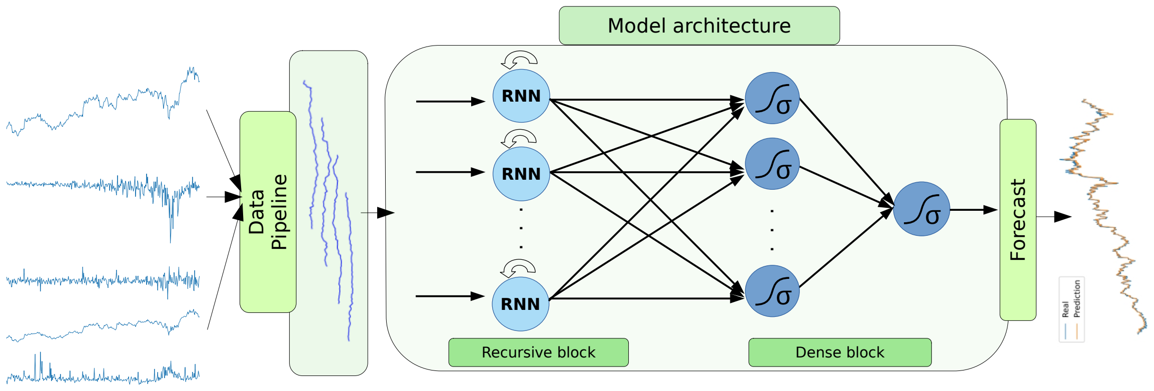


Figure 6.10: Procedure followed in this paper to get, process, serialize, train, and forecast according to our model architecture.

Source: Own elaboration.

In the mentioned models, we have designed them with 128 neurons for both LSTM and Elman's RNN. Every recursive block returns the sequences of the hidden gate h_t to maintain the long-term rendering. They also have a fully connected layer with the same neuron units as the recursive block and a last connected layer with one neuron as output. The first FC layer mixtures the sequences by merging the short and long responses, thus we obtain further relationships between series. As part of the model control, in the recursive kernel we have set an adaptive dropout of $N_f 10^{-2}$ and an L2 regularization of $N_f 10^{-4}$, in which N_f is the number of features that contains the series. Hence we have been more restrictive when more elements to analyze in order to avoid overfitting and to standardize the fitting phase. On the one hand, the dropout helps us to regularize by omitting a proportion of neurons within the neural networks during training. On the other hand, the L2 regularization in the recursive kernel allows us to prevent leverage from large outliers. For a detailed description, Fig. 6.10 shows the different stages whereby the data is computed until we get the forecast.

6.2.6 *Evaluation metrics*

In order to gauge the fitting obtained by our models, we have considered some evaluation metrics for comparison purposes. This stage is fundamental to understanding the loss given by our predictive model, so we have appropriately selected three different measures. First, we considered the mean absolute error (MAE) Eq.6.16 as the loss function when training the models (§ 6.2.7). We have decided to take this deviation measure because it is less sensitive to erratic predictions. It is easy to note that RMSE $\left(Y, \hat{Y}\right) \geq$ MAE $\left(Y, \hat{Y}\right)$, for any time series $Y = \{y_t\}_{t=1}^{N}$ real-valued. Then, unlike some papers, we decided to dispense with RMSE because it would not give us further information. Second, we needed to choose a relative error measure of accuracy, so we selected the mean absolute percentage error (MAPE) Eq.6.17. Because the interval over which the closing price varies is considerably high, we expect to obtain lower values as long as the prediction is accurate. Third, it is essential to know whether a prediction fits actual data, so we have deployed the ρ-score, known as the Spearman coefficient, to measure how correlated are the real and forecasted values. Moreover, this coefficient does not rely on linear relationships but on monotonic growths.

Given a time series sequence $Y = \{y_t\}_{t=1}^{N}$ and a prediction of such series $\hat{Y} = \{\hat{y}_t\}_{t=1}^{N}$, the definition of the above-mentioned measures is presented in the following equations:

$$\text{MAE} \quad \frac{1}{N}\sum_{t=1}^{N}|y_t - \hat{y}_t| \tag{6.16}$$

$$\text{MAPE} \quad \frac{100}{N}\sum_{t=1}^{N}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \tag{6.17}$$

$$\rho\text{-score} \quad 1 - \frac{6\sum\limits_{t=1}^{N}(y_t - \hat{y}_t)^2}{N(N^2-1)} \tag{6.18}$$

Although other accuracy metrics could be interesting when predicting stock values, we have avoided training our models with such an approach. Not only because we aim to offer an accurate global fitting with veracious predictions, but we want to quantify the effect of external elements over the target feature.

6.2.7 *Software implementation*

All the preparation, processing, methodology, and modelization described in this paper have been implemented in Python 3.8. The data has been processed with Pandas 1.3.4 and scaled with Scikit-Learn 0.22. The recurrent models have been implemented using Keras with TensorFlow 2.4.1 as the backend. For compatibility purposes, the NumPy version selected was 1.20.3.

With respect to the fitting phase, all the models were trained using Adam optimizer (Kingma and Lei Ba, 2014) with a learning rate of 10^{-3} and MAE as a loss function. As the structure of the models mentioned in § 6.2.5 are uneven, such learning rate may not be suitable per each one. Hence, we have defined a learning rate reducer (ReduceLROnPlateau) which diminishes the current learning rate by a factor of 0.5 when the loss has not improved for 10 epochs. Moreover, we have avoided big reductions by setting a minimum value of 10^{-5}. Finally, we have selected 200 epochs per each learning phase, in which the fitting stops automatically (EarlyStopping) when the validation loss did not improve, at least a progress of 10^{-4}, for 30 epochs.

6.2.8 *Results and discussion*

We have developed 24 different recursive models in order to contrast the performance obtained depending on the regressors that we have fed to them. Basically, we have four sets of regressors, three kinds of architectures, and two recursive layers. We wanted to avoid randomness during the fitting, so we have fixed the Keras seed, the batch size and the dataset shuffle option. In addition, the data pipeline is a regular process, then the only different part is the number of regressors that it returns. Provided that the number of experiments is quite big, we had no chance to search the optimal parameters to each model without losing generality among the others, anyway we have ensured that training loss curves follow an appropriate decreasing rate. Fig. 6.11 shows the forecasts obtained per each model divided into LSTM and Elman's layers as columns. Thus we can compare the output of each model by the regressors utilized.

When comparing the results obtained in tables 6.8 and 6.9, it has been shown that lightweight architectures can outperform deeper ANNs. By seeing the prediction measures, Elman's networks have obtained a mean of 75.44 whilst LSTMs obtained a mean of 173.03, which means an MAE difference of more than twice in terms of loss. However, when comparing this measure by isolated model structures, we find that LSTMs outperformed Elman's in every regressor set but the S-based as stated in table 6.10.

For the layer implementation, we can see that bidirectional Elman's models have achieved the lowest loss. It is worth mentioning that simple models have shown robustness at predicting albeit their lightweight structure in terms of the number of parameters and model depth. Another point that we would to emphasize is the limited usefulness of the stacked models for our case study.

Table 6.8: Evaluation of LSTM models broken down by the regressor sets.

		Training			**Validation**		
Regressor	Model	MAE	MAPE	ρ-score	MAE	MAPE	ρ-score
SP	Simple	33.42	0.5309	0.9997	43.56	0.5852	0.9941
	Stacked	12.57	0.1919	0.9999	59.23	0.8044	0.9622
	Bidirectional	33.23	0.5292	0.9997	62.47	0.8462	0.9830
NSNP	Simple	85.45	1.2508	0.9939	80.49	1.0869	0.9400
	Stacked	80.90	1.1790	0.9945	78.66	1.0615	0.9404
	Bidirectional	84.78	1.2392	0.9940	80.39	1.0853	0.9401
P	Simple	35.43	0.5369	0.9996	37.41	0.4993	0.9978
	Stacked	54.40	0.7799	0.9973	90.57	1.2157	0.9216
	Bidirectional	44.51	0.6867	0.9995	49.03	0.6532	0.9980
S	Simple	88.02	1.2889	0.9936	228.33	3.1407	0.9175
	Stacked	82.71	1.2124	0.9944	949.08	12.9104	0.6303
	Bidirectional	86.81	1.2708	0.9937	317.15	4.3710	0.9009

Table 6.9: Evaluation of RNN models broken down by the regressor sets.

		Training			Validation		
Regressor	Model	MAE	MAPE	ρ-score	MAE	MAPE	ρ-score
SP	Simple	34.83	0.5468	0.9998	44.42	0.5993	0.9937
	Stacked	32.94	0.5281	0.9992	93.03	1.2588	0.9580
	Bidirectional	28.96	0.4498	0.9998	36.92	0.4970	0.9924
NSNP	Simple	83.67	1.2270	0.9945	81.34	1.0993	0.9398
	Stacked	83.24	1.2264	0.9946	80.28	1.0824	0.9424
	Bidirectional	84.40	1.2413	0.9945	81.48	1.1010	0.9407
P	Simple	39.18	0.6109	0.9998	41.29	0.5512	0.9965
	Stacked	78.06	1.1538	0.9953	114.56	1.5420	0.9237
	Bidirectional	39.44	0.6155	0.9996	36.23	0.4827	0.9964
S	Simple	84.58	1.2484	0.9945	75.83	1.0218	0.9502
	Stacked	85.09	1.2837	0.9947	141.92	1.9267	0.8805
	Bidirectional	84.58	1.2448	0.9944	77.96	1.0545	0.9432

An interesting point is that social media, in particular Twitter, can influence the stock market. It has been proven that trying to predict stock features by considering the accumulated sentiment analysis intra-day is considerably chaotic, especially when there are activity spikes registered over some concern that affects the company. The contrast for tables 6.8 and 6.9 indicates that long dependencies worsen the prediction process, so it can only take into consideration the short term of the series. Although it is possible to train and obtain acceptable results if the validation set contains non-regular data the forecast will not be representative. Another important view is that the worst predictions have been obtained with the S approach.

Another evidence that sentiment analysis is influential is the lower impact obtained when training with the entire set of features. The most feasible explanation is that accumulated sentiment affects similarly the stock features, then adds exogenous information about their variation. In the training phase, the model with more complexity (Stacked) has achieved the lowest loss (12.57), unlike what happened by removing it (35.43), and almost 1 ρ-score which demonstrates an exact fitting. Therefore, robust models with appropriate tuned parameters could outperform it and show that sentiment analysis could help to fit the forecast in stock markets.

Even though the SP-forecast had great performance, the top performed forecasts have been achieved just with the stock features, with 37.41 for LSTM and 36.23 for RNN. This can be related to what we have mentioned in the last paragraph about the influence of sentiment analysis. It is worth mentioning that such impact is less significative than the S-case with a difference of 7.58% (stacked model) against 76.78% (bidirectional model). In any case, the minor gap between both sets was reached with the simple architecture with a difference of 0.69 in MAE, which is practically unnoticeable with regard to the values of the series.

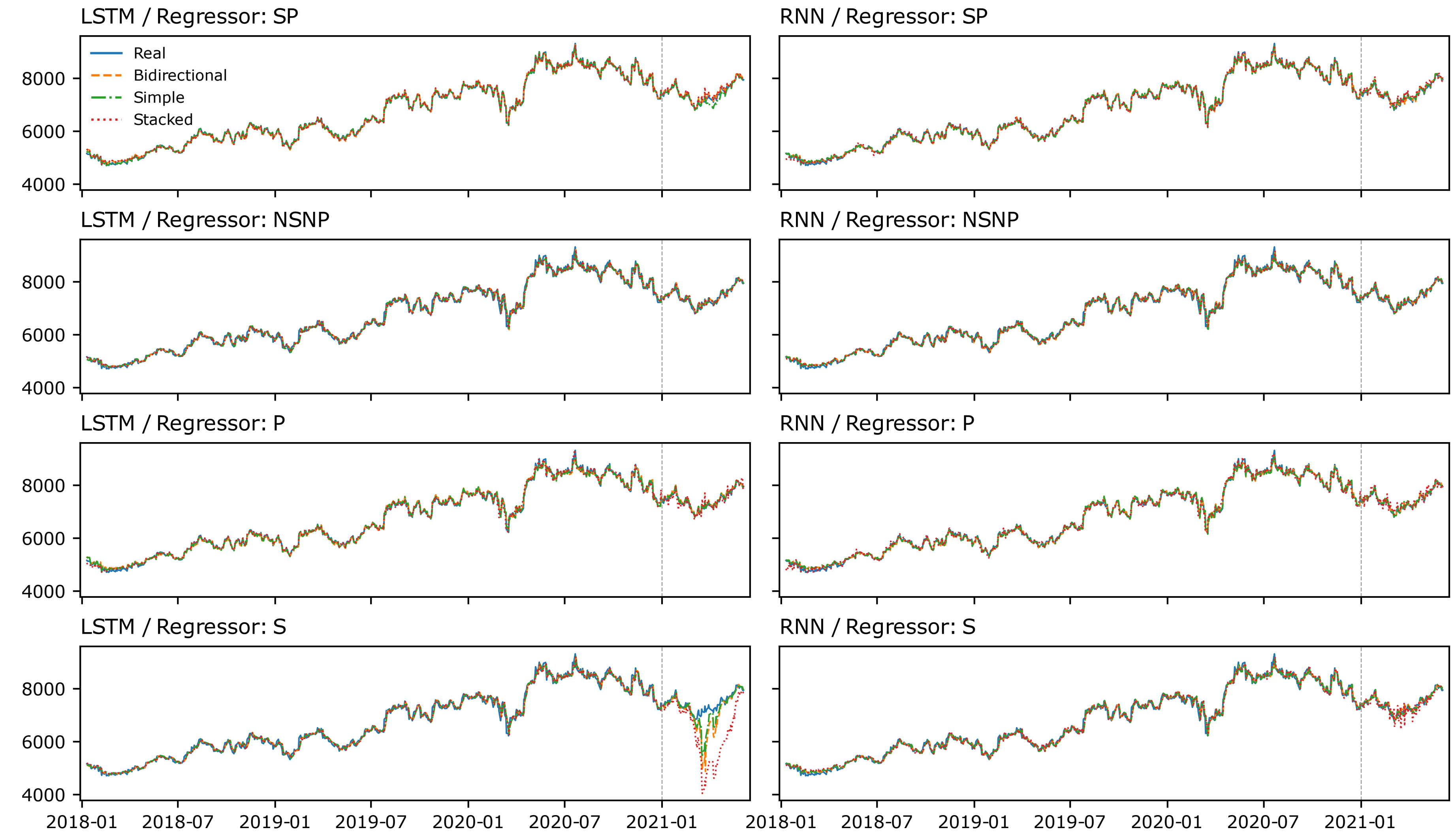


Figure 6.11: Results obtained for each recursive model and divided depending on the model architecture. The dotted gray line splits the train and validation sets.

Source: Own elaboration.

Table 6.10: Mean of the MAE validation loss obtained per each regressor set and model architecture, grouped by model type.

Model	SP	NSNP	P	S	Simple	Stacked	Bidirectional
LSTM	55.09	79.85	59.00	498.19	101.37	295.20	122.53
RNN	58.12	81.03	64.03	98.57	60.72	107.45	58.15

We would also want to highlight the importance of our selected measures in § 6.2.6. When contrasting the predictions, we can spot that Elman's RNNs obtained a slightly better ρ-score in NSNP-case (0.9722 against 0.9725) than LSTM despite having higher MAE (64.03 against 59.00). It may happen because the fitting is not as good in comparison, though it has a more adequate monotonic behavior. Additionally, it could be mentioned that it also has the worst percentage error (0.86% against 0.79%).

Regarding the model complexity, we can see that simple models have obtained great results in spite of their limited number of parameters. In cases in which the computational times are crucial, simple Elman networks may play an important role and it could also be studied as a reduction of the number of neurons.

6.2.8.1 *Outlier detection*

As mentioned above, the sentiment analysis feature seems to worsen our predictions when adding it to our model architectures. Therefore, it is necessary to study its behavior over time and understand why just this variable has this impact on our data. We have implemented a basic technique for time series anomaly detection to check local deviations in a sequence. We have decided to label the elements of a series as anomalous (i. e. an outlier point) whenever $s_t \notin \left[Q_1^{SA} - \frac{5}{2}IQR^{SA}, Q_3^{SA} + \frac{5}{2}IQR^{SA}\right]$, being $SA = \{s_t\}_{t=1}^{N}$ the accumulated sentiment time series, Q_k^{SA} the k-quartile and $IQR^{SA} = Q_3^{SA} - Q_1^{SA}$ the interquartile range of SA respectively.

Fig. 6.12 illustrates the distribution of outliers over time, in which it is shown that 10 elements belong to the training set and 46 belong to the validation set. In other words, the training set has a 1.32% of anomalous values, while the validation set has a 42.59% of

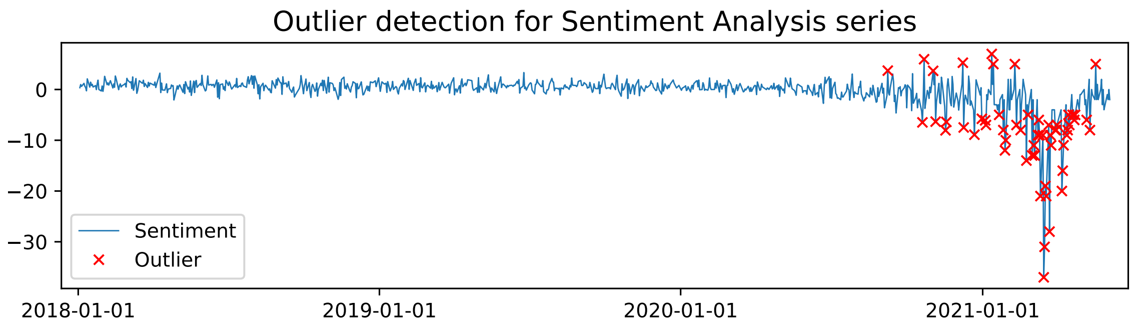


Figure 6.12: Anomaly detection performed for the accumulated sentiment analysis features. The interval selected to label values as outlier is $[Q_1 - \frac{5}{2}IQR, Q_3 + \frac{5}{2}IQR]$.
Source: Own elaboration.

them. Even though it may seem a bad choice of this feature, it clearly indicates that the underlying misinformation and fake news in Twitter have not had a decisive impact on the AstraZeneca company. As we can see in the S-case in Fig. 6.11, the long-term model has predicted a drastic decrease, however it did not actually happen. In fact, it returned a maximal error between 3.14% and 12.91% with a bad fit to the original curve. We would like to emphasize that, when applying our model architectures to real-time events, it is necessary to analyze all the possible anomalies of the series before the prediction is computed.

6.2.8.2 *Regressors impact*

Most parts of this paper rely on the study of the impact of each set of regressors. We have developed four different approaches depending on the regressors utilized, then we can measure the impact of them when we add or substrate when generating the models. This way, we can assess the performance of the models regarding the use of sentiment analysis or stock features. Table 6.11 shows the difference and the error obtained, in terms of MAE, when a regressor set is added. It is clear that the stock features played a key role when predicting the close price of AstraZeneca because this set improved the performance of the forecast in every case study. Conversely, the accumulated sentiment analysis only allows for an enhanced fitting when it is combined with the prices. It substantially became worse when we attempted to predict in the sentiment-based model, especially for the LSTMs. Such an effect could happen due to the number of features to process, since the impact in all-based models is always positive, so it possibly complemented the variations of each other descriptive variable.

In regard to the layer performance, it is noteworthy that higher improvements in loss measures are given by the LSTMs. Actually, it is also proof that RNNs are more stable in multivariate problems, in which the experimental loss has been 63.25 ± 25.08, and LSTMs can learn from more sequences in comparison with Elman layers because of the transfer of long-term information. As far as training is concerned, the reader must bear in mind that there has been a standardization process for all the models, hence we could profit as much as we tune the hyperparameters involved per each case.

6.2.9 *Conclusions and future work*

1. Design a web scrapping pipeline for real-time data through stock and news sources.

Table 6.11: Difference of the MAE validation loss obtained when comparing the addition of regressors. Values in parentheses are the error of such addition.

Model	NSNP→S	P→ SP	NSNP→ P	S→ SP
LSTM	418.34 (5.24)	-3.92 (-0.07)	-20.85 (-0.26)	-443.10 (-0.89)
RNN	17.54 (0.22)	-5.91 (-0.09)	-17.00 (-0.21)	-40.45 (-0.41)

2. Make our own custom Sentiment Analysis extractor by implementing an NLP model with deep neural networks.
3. Model and tune the hyperparameters of each model architecture, one by one, in order to get a performance as balanced as possible per each regressor set.
4. Define an indicator that measures the robustness-weakness of a company against online social content.

6.2.10 *Python implementation*

The scripts utilized to perform the regression task are displayed in § B.2. Owing to the nature of this problem and the high complexity attached, we have presented a bash file (Script 11) and a Python file (Script 12) with a callable argument parser.

6.3 AI-CASE 3: ON THE APPLICATION OF EXPLAINABLE ARTIFICIAL INTELLIGENCE TECHNIQUES ON HRTF DATA

This case study is based on our published research: De Rus J.A., López-García A., López-Ballester J., Lopez J.J., Torres A.M., Ferri F.J., Montagud M., Cobos M., "On the Application of Explainable Artificial Intelligence Techniques on HRTF Data". In 24th International Conference on Acoustics, ICA 2022. Hence, all the information or results mentioned in this section have been already addressed in that paper.

6.3.1 *Experiment Setup*

Binaural reproduction is today a widely used technique for immersive spatial audio. To provide a listener with a realistic sensation of spatial hearing over headphones, sound signals are typically convolved with Head-Related Impulse Responses (HRIRs), or filtered according to their spectral equivalents, i. e. the Head-Related Transfer Functions (HRTFs). These describe the transfer properties of sound waves as they travel from a given sound source location in space to the ear canal in free space. Since HRIRs are highly individual (they depend on a subject's anthropometric features), deviations from the user's own HRIRs can affect negatively the listening experience. Therefore, the identification of relevant localization cues and their preservation is a topic of continuous interest within the spatial audio research community. In this context, while numerous studies have been carried out in the past to identify salient localization cues, for example by applying principal component analysis (PCA) to HRIR datasets, some recent works are exploiting the feature learning capabilities of deep learning-based approaches. In this work, we explore the use of common explainable artificial intelligence (XAI) techniques, such as class activation mapping, on convolutional neural networks (CNN) trained for classifying HRIR datasets into different directional sectors, exploring further this issue.

6.3.2 *Introduction*

With the advent of immersive acoustic scenarios for virtual reality, achieving accurate localization of sound sources has become a major challenge. One of the peculiarities that complicate the development of accurate reproduction systems is that human spatial hearing is closely related to the listener's anatomy. Acoustic effects caused by the head, torso, shoulders, and pinnae have a great impact on human localization ability. It is a well-known fact that such human characteristics can be statistically described to aid the auditory localization process (Searle et al., 1976). Most acoustic models usually utilize time-domain Head-Related Impulse Responses (HRIRs) or, equivalently, Head-Related Transfer Functions (HRTFs) in the frequency domain. With regards to HRTF signals, several works have proposed the use of preprocessing and postprocessing techniques to capture the relative influence of relevant anthropometric features (Zhu et al., 2017). The localization of virtual sources can also be improved via scaling the directional transfer functions (Middlebrooks, 1999). An interesting finding that matches with the objective of this paper is that prominences (either peaks or notches) in the curves of the transfer functions may reveal information associated with the elevation of a source (Blauert, 1996; Hebrank and Wright, 1974).

The use of deep neural networks for decision aid in acoustic environments has been growing steadily. In particular, the use of convolutional neural networks (CNNs) is crucial in most of the artificial support tasks like acoustic scene classification (Abeßer, 2020), music tagging (Kim et al., 2018), and speech emotion recognition (Mustaqeem and Kwon, 2020), among many others. Along the same line, the development of binaural spatial datasets allows training neural networks in order to model personalized HRTFs and enable a more realistic listening experience. Recent studies have shown the possibility to capture spatial audio features in HRTF datasets using CNNs (Thuillier et al., 2018). These studies may be understood as a new manner to study human performance in source localization. Other studies have recently focused on the front-back discrimination of binaural music recordings (Zieliński et al., 2022), revealing interesting information about the relevant frequency bands assisting such discrimination tasks.

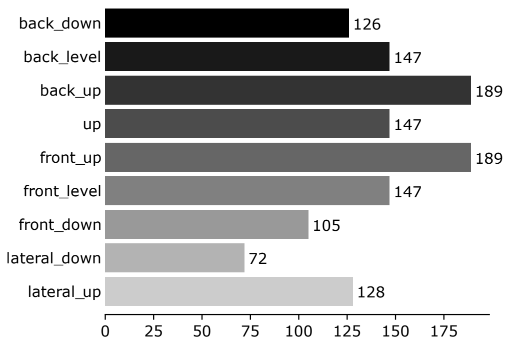
In the same spirit of Thuillier et al. (2018), this paper focuses on the application of explainable artificial intelligence (XAI) techniques for the analysis of HRTF datasets. More specifically, we consider the analysis of the HRTF CIPIC dataset (Algazi et al., 2001) using a conventional 1D-CNN model. In contrast to Thuillier et al. (2018), HRTFs transformed to a mel-scale are directly used as input to the network. Moreover, we apply two different XAI techniques for saliency analysis, namely class activation mapping (CAM) and the more general gradient-based CAM method (Grad-CAM). The results obtained using these techniques will be analyzed to discover frequency bands in the HRTFs that encode location cues relevant to the determination of the elevation of a source.

6.3.3 *Project inception*

6.3.3.1 *Experimental dataset and data preprocessing*

The CIPIC HRTF Database is a public-domain database of high-spatial-resolution HRTF measurements for 45 different subjects, including the KEMAR mannequin with both small and large pinnae (Algazi et al., 2001). It includes 2.500 measurements of HRIRs for 45 subjects at 25 different azimuths and 50 different elevations (1250 directions) at approximately 5° angular increments. The standard measurements were recorded at 25 different interaural-polar azimuths and 50 different interaural-polar elevations. The sample duration of each HRIR is 200 samples (4.5 ms at a sample rate of 44.1 kHz). Additional special measurements of the KEMAR manikin were made for the frontal and horizontal planes.

As in Thuillier et al. (2018), we focus this study on the analysis of elevation cues. To this end, we divide all the responses of the CIPIC database into a set of 9 spherical regions according to their elevation, as depicted in Fig. 6.13. Note that the number of sampled directions within each of the elevation classes is not uniform, although not severely unbalanced.



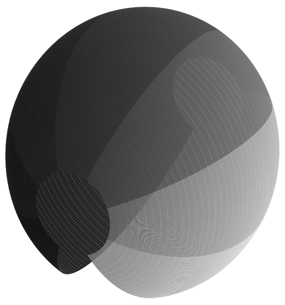


Figure 6.13: Spatial source location in the CIPIC dataset divided into 9 regions for the classification task.

Source: Own elaboration.

The full dataset is composed of 56250 HRIR samples (corresponding to the combination of the 45 subjects, 25 azimuth angles, and 50 elevation angles) with their associated ipsilateral and contralateral channels.

To obtain the corresponding HRTF signals, we compute the one-sided fast Fourier transform of each channel with 512 points, resulting in 257 frequency bins. In order to provide the network with a perceptually-motivated input, we warp the frequency axis by considering a mel-scale mapping. This is achieved by dividing the range [0-22050] Hz into 257 uniformly spaced points in the mel-scale and taking the frequency bins that are closer to their equivalent frequencies in Hz. Finally, only the magnitude spectrum of each channel in logarithmic scale is considered. The shape of each dataset example is therefore (257,2).

The dataset was divided into two partitions for training and validation. The data of 36 subjects (45000 samples) was used for the training partition and the data of 9 different subjects (11250 samples) for validation.

6.3.3.2 *Model architecture*

This work considers a fully convolutional model with a straightforward architecture, represented in Fig. 6.14. The design of the network was carried out with the aim of achieving significant classification accuracy while keeping the model simple enough to facilitate the application of common XAI techniques. It consists of three 1D convolutional blocks with ReLU activation and max-pooling in between blocks to downsample the frequency information. A last convolutional layer followed by Global Average Pooling (GAP) is used to summarize the filter responses before the final dense layer, configured with softmax activation.

The information related to the construction of each layer of the CNN is shown in Table 6.12. It shows detailed information about how the model was built, including the dimensions involved at different depths and the number of associated parameters.

The model was trained with Adam optimizer (Kingma and Lei Ba, 2014), with $\eta = 10^{-3}$, using categorical cross-entropy as a loss function. For the fitting, we set a batch of 16 elements and 100 epochs with early stopping, which actually was activated during the training processes.

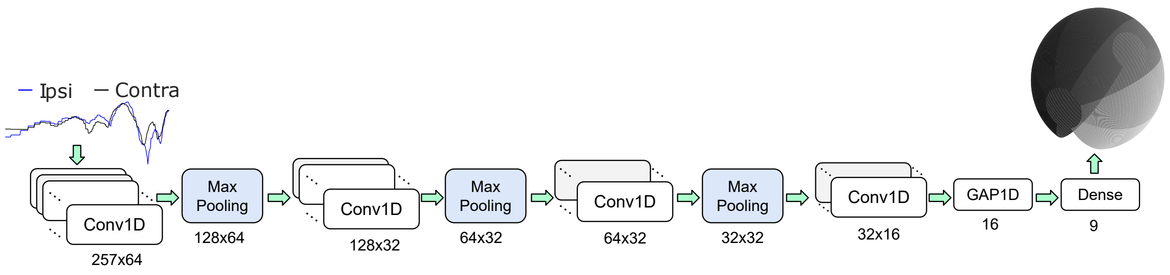


Figure 6.14: Topology of the convolutional architecture developed for this paper to classify HRTFs into nine elevation sectors.
Source: Own elaboration.

Table 6.12: Configuration of the CNN layers with its first dimension (Dims), number of filters, kernel sizes, and number of parameters (Params). Here, Conv stands for 1D-Convolutional layer and Pool for 1D-Max-Pooling operator.

	Input	**1st Block**		**2nd Block**		**3rd Block**		**Output**		
		Conv	Pool	Conv	Pool	Conv	Pool	Conv	GAP	Dense
Dimension	257	257	128	128	64	64	32	32	16	9
Filters	2	64	64	32	32	32	32	16	1	1
Sizes	0	64	2	16	2	8	2	8	0	0
Parameters	0	2112	0	32800	0	8224	0	4112	0	153

6.3.3.3 *Feature explainability*

The use of XAI techniques may be understood as an additional phase in model evaluation (Arrieta et al., 2020). As a manner to give explainability in the same way as the studies above mentioned, we have analyzed the outputs of our deep convolutional architecture. Once our model is fitted and yields an acceptable performance, we want to remove the black-box produced by the CNN architecture. With that purpose, we have applied CAM (Zhou et al., 2015) and Grad-CAM (Selvaraju et al., 2017) to analyze both saliency maps and class activation maps. Even though both XAI techniques were designed solely to cover the limitations of CNNs, their implementation is widely extended thanks to their ability to generate localization maps highlighting significant spatial regions (Li et al., 2018). CAM originally arose as a technique that yields class activation maps of CNNs applied over object detection tasks. It allowed the trained models to localize class-specific patterns in image detection. Owing to the main disadvantage of CAM is the requirement of a GAP layer to operate over the convolutional filters, Grad-CAM overcomes that limitation by the use of gradients over the first dense layer in the architecture. Regardless of their differences, both techniques rely on the assumption that the decision vector Y^c for the class c is described by means of the feature maps A^k of the last convolutional layer. Then, the saliency map for the class c is a weighted aggregation of the spatial components of each feature map, written as:

$$\begin{aligned} \text{Class-score:} \quad & Y^c = \sum_k w_k^c \sum_i \sum_j A_{ij}^k \\ \text{Class-Saliency:} \quad & L_{ij}^c = \sum_k w_k^c A_{ij}^k. \end{aligned} \tag{6.19}$$

Where the i and j indexes denote the matrix indices of the spatial components and the k index associates the number filters. Regarding the weighting scheme (w_k^c), CAM utilizes a backward projection of the output weights onto the maps of the last convolutional layer. On the contrary, Grad-CAM estimates each class-weight for each feature map as a linear combination of the partial derivatives $\frac{\partial Y^c}{\partial A_{ij}^k}$. Their implementation is highly related to image classification (Li et al., 2018; Selvaraju et al., 2017; Zhou et al., 2015) varying from clinical test (Chang et al., 2021a) to computer games simulation through reinforcement learning (Joo and Kim, 2019).

6.3.4 *Results*

6.3.4.1 *CNN performance*

After training, the model achieved global accuracy of 0.8090 on the validation set, which suggests that the model performs reasonably well in the task of determining the elevation class of a given HRTF, regardless of its azimuth. To analyze the classification performance in more detail, Fig. 6.15 shows the confusion matrix (hit percentage) for the different elevation classes. The proportion of class examples is shown in red-scale. Note that most wrong predictions are misclassified to adjacent spatial regions, showing robustness in terms of understanding the underlying spatial cues.

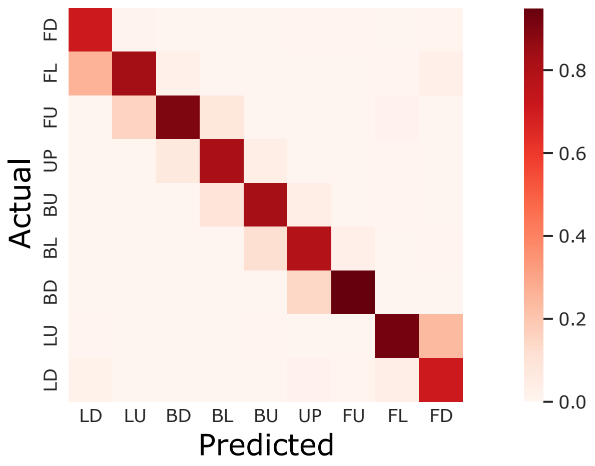


Figure 6.15: Confusion matrix computed by hit ratio for the CNN model.
Source: Own elaboration.

6.3.4.2 *Saliency Maps*

This section shows the results obtained by the XAI techniques considered in this work, analyzing the saliency maps provided by the CAM and Grad-CAM methods. These saliency maps provide meaningful information on how the convolutional kernels process a given input example to determine its corresponding elevation class. Fig. 6.16 shows representative HRTF responses selected from the validation set together with their corresponding saliency in the background. The selection was made according to a maximum-probability criterion, i. e. they correspond to the responses pertaining to each class that the model classified with the highest confidence. Dark red zones correspond to frequencies having a high saliency, while light zones correspond to frequencies that are less relevant for the classification task. For each class, the CAM saliency is shown at the top of the plot, while the Grad-CAM result is shown at the bottom. It can be observed for each of the represented examples that both CAM and Grad-CAM provide similar saliency zones, although the intensity of such zones may vary from one method to the other. In general, there is a high level of agreement in the results provided by both approaches.

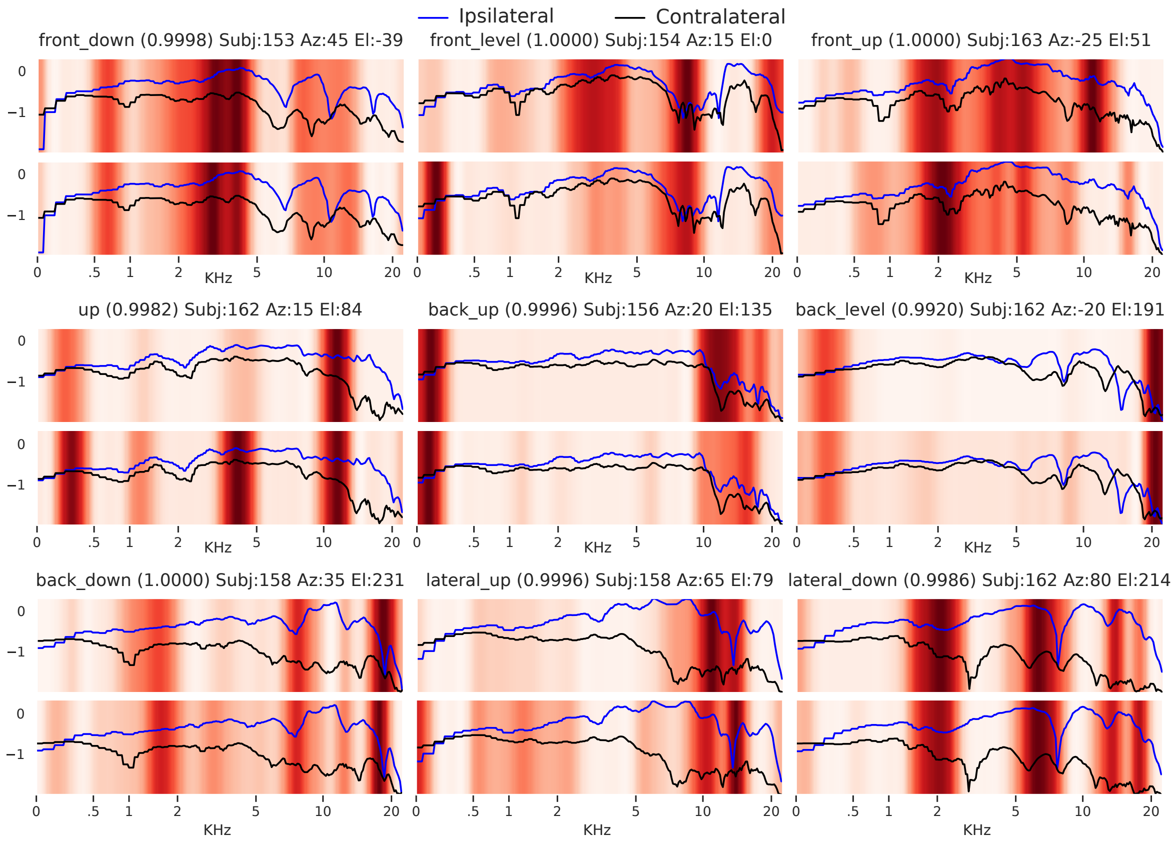


Figure 6.16: CAM (top) and Grad-CAM (bottom) saliency maps over the most representative sample of each class. The predicted class probability $\hat{y}_i$ is given in parentheses. The color bar is red-scaled, then white shades indicate low relevance and red ones have high relevance.

Source: Own elaboration.

To gain more insight into the relevant spatial components processed by the CNN model, we analyzed the average saliency maps resulting from CAM and Grad-CAM across all subjects and azimuth angles. The aim is to get an overall picture of the relevant frequencies assisting the classification of the whole dataset. The lateral regions were not considered as they do not really affect elevation. The averaged frequency-elevation saliency maps are shown in Fig. 6.17, where red indicates more importance and blue is less important for classification.

Finally, Fig. 6.18 shows as an image the saliencies of the individual responses belonging to each class in the validation set, both for CAM and Grad-CAM. Note that there is a high correlation between the saliency results obtained for the different responses within each class, suggested by prominent vertical bands located at narrower or wider frequency regions. Again, the results obtained by CAM and Grad-CAM are considerably similar.

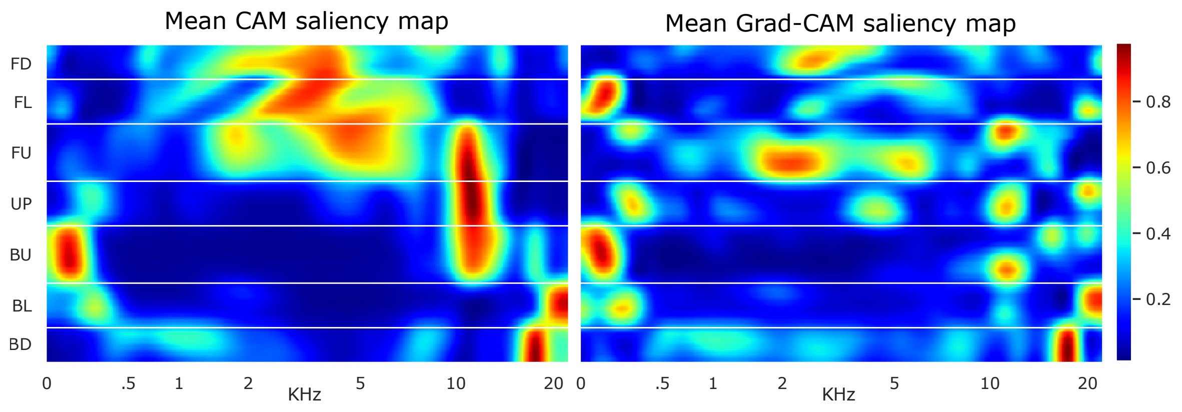


Figure 6.17: CAM (left) and Grad-CAM (right) saliency maps averaged across subjects and azimuth, showed per elevation class.
Source: Own elaboration.

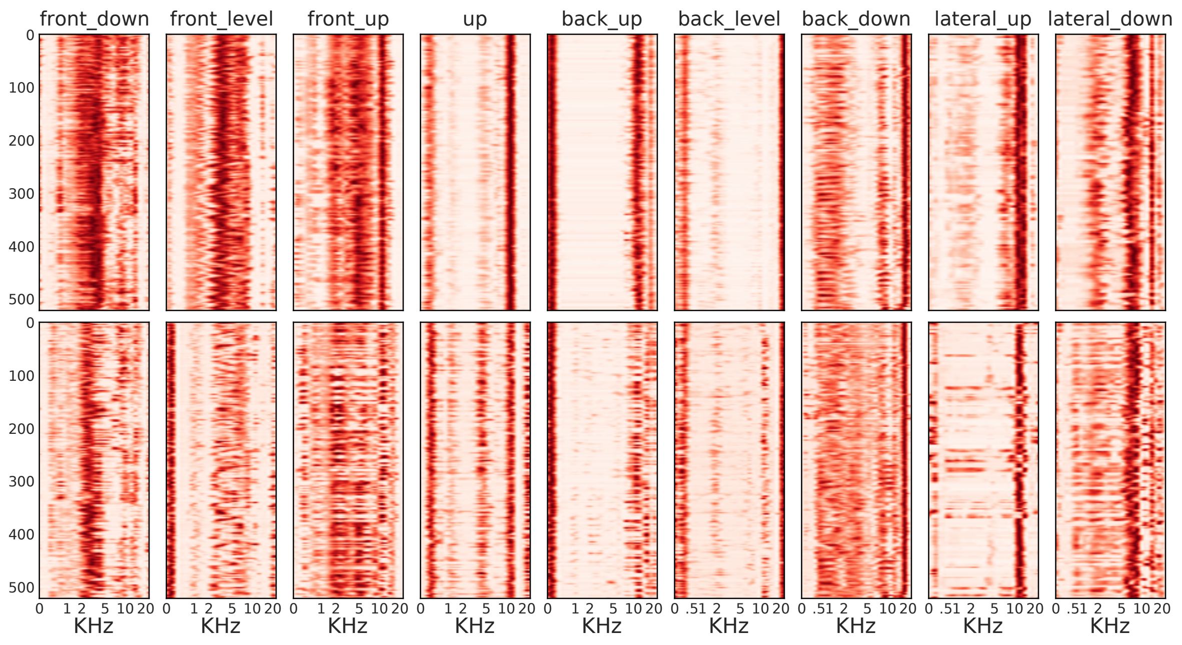


Figure 6.18: Saliency bands of CAM (top) and Grad-CAM (bottom) per class in validation partition for correct predictions. The color bar is scaled, then light shades indicate high relevance and dark ones low relevance.
Source: Own elaboration.

6.3.5 *Discussion*

The saliency maps obtained in the previous section are considered in accordance with some of the results derived from previous psychoacoustical experiments. Next, we describe how our results are linked to some known spectral cues and effects related to elevation and front-back discrimination. Note that, since the average CAM saliency map in Fig. 6.17 (left) seems to be more consistent than the one obtained from Grad-CAM, we will discuss our findings taking into account mostly the CAM results.

Many studies have shown that spectral distortions caused by pinnae in the high-frequency range approximately above 4 kHz act as cues for median plane localization (Iida and Ishii, 2011). Indeed, by looking at the resulting saliency maps, the most intense saliencies are found, as expected, in the mid and high-frequency range. However, there are some interesting low-frequency effects (below 500 Hz) in the classes “up", “back-up", “back-level" and “back-down". These may suggest that HRTFs from back directions show some low-frequency features assisting the perception of elevation that may not be present for frontal directions. In this context, although a behind cue was reported by Hebrank and Wright (1974) to appear as a small peak around 12 kHz, we observe the average saliency to be high at this frequency for “back up", but also contributions from higher frequencies at “back level" and “back down", suggesting a shift of cues towards very high frequencies when a behind source moves from down to up.

Butler and Belendiuk (1977) showed that the prominent notch moves toward the lower frequencies as the sound source moves from above to below the aural axis in the frontal half of the median plane. For frontal directions, moving from “front-up" to “front-down", we can indeed see the saliency change towards lower frequencies (from around 8 kHz to 3 kHz).

Additionally, as observed in Fig. 6.16 and Fig. 6.18, there appear to be complementary saliencies on samples of opposite spatial classes that may hint at more cues, as follows. Samples from the classes “lateral-up" and “lateral-down" present opposite saliencies on the 8-15 kHz band, while being similar on the rest of the frequencies. Similarly, samples from the classes “front-level" and “back-level" present opposite saliencies on the 2-10 kHz band, while being quite similar out of this range.

6.3.6 *Conclusions*

This work presented a preliminary study on the use of explainable artificial intelligence techniques, namely CAM and Grad-CAM, for assisting in the interpretation of HRTF elevation cues. To this end, we trained a convolutional neural network on mel-scale-warped HRTF responses extracted from the CIPIC database. The model was trained to classify responses into 9 different spatial classes related to different elevation sectors and showed considerable generalization capabilities over a validation set with responses from subjects different from the ones in the training set. The explainability techniques were applied over the trained model to obtain saliency maps indicating the relevant frequency bands used by the network to classify a given input response into one of the elevation classes. Although both CAM and Grad-CAM provided similar saliency regions, the results of CAM appeared to be more consistent across both the training and validation sets. The saliency regions identified by the applied explainable techniques were also consistent with most findings obtained through psychoacoustic experiments, although additional unexpected low-frequency effects were obtained for behind directions.

6.3.7 *Python implementation*

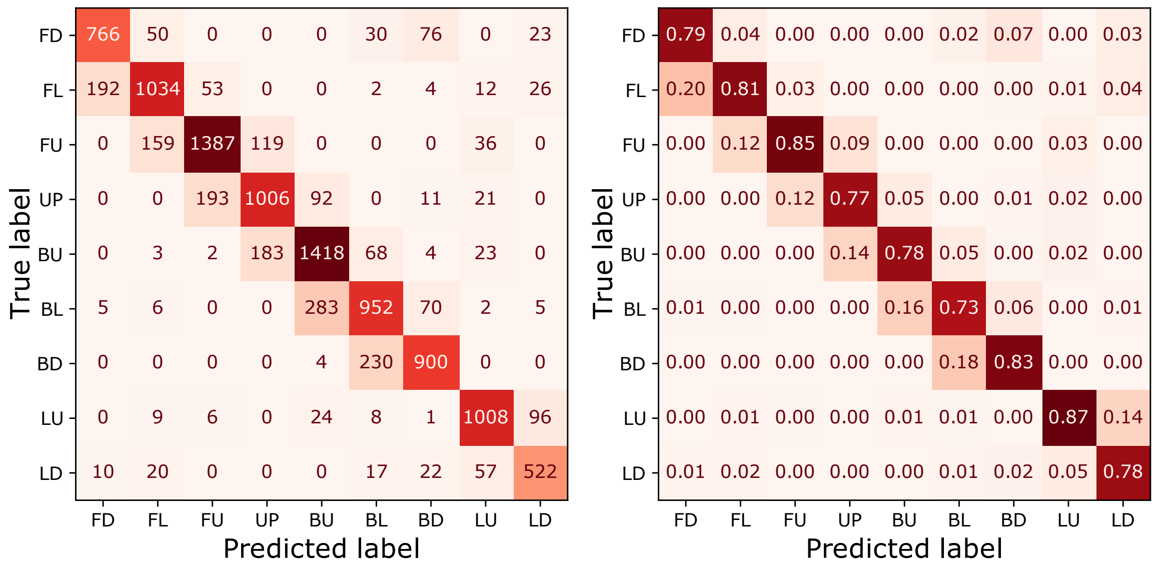
The required code to perform this case study is shown in § B.3, in which the entire procedure (Script 13) and model implementation (Script 14) are approached.

6.3.8 *Extra section: CNN evaluation*

Here we want to attach additional information shown in § 6.3.4.1 related to the CNN performance. Given that the problem is a multi-classification task with 9 different labels, we offer the automatic Sklearn report of the function classification_report.

Table 6.13: Classification report for the trained convolutional network for the validation set.

Label		**Precision**	**Recall**	**F1-score**	**Support**
Front Down	FD	0.79	0.81	0.80	945
Front Level	FL	0.81	0.78	0.79	1323
Front Up	FU	0.85	0.82	0.83	1701
Up	UP	0.77	0.76	0.76	1323
Back Up	BU	0.78	0.83	0.81	1701
Back Level	BL	0.73	0.72	0.72	1323
Back Down	BD	0.83	0.79	0.81	1134
Lateral Up	LU	0.87	0.88	0.87	1152
Lateral Down	LD	0.78	0.81	0.79	648



True \ Predicted	FD	FL	FU	UP	BU	BL	BD	LU	LD
FD	766	50	0	0	0	30	76	0	23
FL	192	1034	53	0	0	2	4	12	26
FU	0	159	1387	119	0	0	0	36	0
UP	0	0	193	1006	92	0	11	21	0
BU	0	3	2	183	1418	68	4	23	0
BL	5	6	0	0	283	952	70	2	5
BD	0	0	0	0	4	230	900	0	0
LU	0	9	6	0	24	8	1	1008	96
LD	10	20	0	0	0	17	22	57	522

True \ Predicted	FD	FL	FU	UP	BU	BL	BD	LU	LD
FD	0.79	0.04	0.00	0.00	0.00	0.02	0.07	0.00	0.03
FL	0.20	0.81	0.03	0.00	0.00	0.00	0.00	0.01	0.04
FU	0.00	0.12	0.85	0.09	0.00	0.00	0.00	0.03	0.00
UP	0.00	0.00	0.12	0.77	0.05	0.00	0.01	0.02	0.00
BU	0.00	0.00	0.00	0.14	0.78	0.05	0.00	0.02	0.00
BL	0.01	0.00	0.00	0.00	0.16	0.73	0.06	0.00	0.01
BD	0.00	0.00	0.00	0.00	0.00	0.18	0.83	0.00	0.00
LU	0.00	0.01	0.00	0.00	0.01	0.01	0.00	0.87	0.14
LD	0.01	0.02	0.00	0.00	0.00	0.01	0.02	0.05	0.78

Figure 6.19: Confusion matrices by absolute values (left) and hit ratio (right).
Source: Own elaboration.

6.4 SUMMARY

In this chapter, we have presented three Artificial Intelligence case studies focused as three different Machine Learning tasks: unsupervised (§ 6.1), regression (§ 6.2), and classification (§ 6.3). Each experiment has been a major challenge due to the high difficulty associated with each task. In order of appearance, the case studies consisted of a musical similarity algorithm based on audio fingerprinting for plagiarism detection, a study on the impact of regressors in a time series forecasting of stock prices for the AstraZeneca company particular instance, and the analysis of sound sources of head-related transfer functions (HRTFs) via Convolutional Neural Networks and explainability techniques. In this manner, we have faced complex real-life problems with customized Machine Learning models showing great performance on the application and resolution of the methods.

7 MULTIPLE-CRITERIA DECISION MAKING & ARTIFICIAL INTELLIGENCE CASE STUDIES

7.1 MCDM & AI-CASE 1: EARLY DETECTION OF STUDENTS' FAILURE USING MACHINE LEARNING TECHNIQUES

This case study has been sent for publication to Expert Systems with Applications as: López-García A, Blasco-Blasco O, Liern-García M, Parada-Rico SE "*Early detection of students' failure using Machine Learning techniques*".

7.1.1 *Experiment Setup*

The educational system determines one of the major strengths of an advanced society. For some time, all countries have had to guarantee accessible and free education to their citizens as a common goal. Actually, a country lacking culture has been shown to be less competitive due to the inequality suffered by its people. Institutions and organizations are putting their efforts into tackling that problem. In most cases, indicators and measures already exist that evaluate the situation of the educational system. Nevertheless, it is not an easy task to explain why their students have failed or what are the conditions that affect such situations. Our proposal is to demonstrate that Machine Learning can predict academic failure by means of the student features stored by the institution. The point is to generate an ensemble tree-based method with MCDM and synthetic oversampling. As a result, we can figure out the problem and how it occurs to propose customized measures. The case study of this paper has been the Industrial University of Santander, Colombia.

7.1.2 *Introduction*

The classification of students via educational features is widely studied by both institutions and research teams. There are many factors that determine the academic performance of students that must be taken into account when predicting how their evolution will be during their university studies. In most of the studies conducted, the measure has been approached from three non-exclusive perspectives: direct use of indicators (Parada et al., 2019; Parada et al., 2017, Liern et al., 2020a), traditional statistical methods (Adams and Hancock, 2000) or through machine learning techniques. (Bhutto et al., 2020, Verma et al., 2022). In other articles, such as Paliwal and Kumar, 2009, an analysis is carried out to predict academic performance using neural networks and traditional statistical techniques.

The development of well-established datasets with detailed information about different academic, economic, and social attributes is being approached for analyzing academic performance (Delahoz-Dominguez et al., 2020; Hussain et al., 2018). That stage is very important since it gives an experimental scenario for researchers with real conditions. The academic prediction is usually addressed with supervised models because the information is gathered a priori for a posteriori case study. In Imran et al. (2019), we can find the implementation of various machine learning methods in this sense. We can also find tree-based methods applied over small datasets (Hasan et al., 2018) since they can work without huge amounts of samples. Keser and Aghalarova (2022) presented an ensemble of gradient boosting machine methods to obtain the most suited possible algorithm via hyperparameter tuning. Nonetheless, juts ML-based models are not always a suited approach for this subject. For example, we can find a combination of fuzzy and neural networks in Hidayah et al. (2013) for capturing the uncertainty behind student behavior. Another example is the use of tree-based methods and fuzzy genetic algorithm (Hamsa et al., 2016). It is noteworthy to mention the efforts made to study the gender gaps that may occur within educational centers, thus guaranteeing gender equality. An example with that line is conducted by Sapiezynski et al. (2017), where they applied class performance with Linear Discriminant Analysis (LDA) via an imbalanced gender perspective. Another point to highlight is the emphasis made on the marginal groups. An example of oversampling methods for tackling imbalanced conditions is found in Thai-Nghe et al. (2009), where they applied the SMOTE algorithm to oversample the students' set.

In this paper, we have combined multiple approaches with the aim of classifying the students of the SEA-UIS program. For that purpose, we have implemented a tree-based classifier so that we cannot only know those prone to fail but also understand why it will happen. In other words, we want to remove the black boxes commonly attached to most machine learning algorithms by controlling the intrinsic feature importance when predicting. Owing to the relation success-failure for the institution is disproportionate, such natural imbalance means a problem for us. That is why we have implemented oversampling techniques focused on marginal samples to increase the accuracy and incorporate a major generalization. We have depicted the procedure conducted in Fig. 7.1, where the five different stages are remarked.

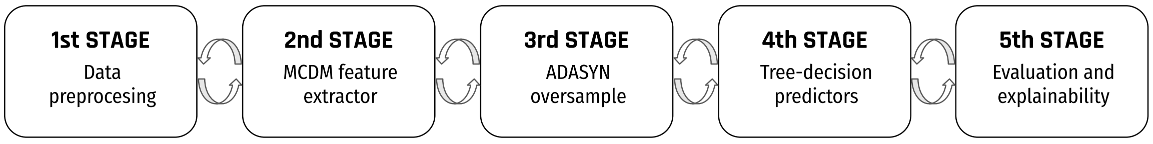


Figure 7.1: Methodological steps carried out in this paper to perform the classification of the SEA-UIS students.

Source: Own elaboration.

7.1.3 *Student classification according to its academic performance*

In this section, we explain the procedure that we have carried out during our experiments. Since descriptive analytics may play a key role in helping the decision-makers of the Industrial University of Santander, it is essential to know how the data is obtained and stored. The more rigorous and accurate we are, the greater support will be given to the educational system. Taking this into account, we have selected the information of the newcomer students known for the UIS-SEA institution before they enroll in the university. To be more precise, the sample obtained belongs to science and engineering careers.

First, feature extraction of a priori information is conducted by applying the TOPSIS and uwTOPSIS methods. The values obtained per student determine its initial stage before the beginning of the first semester. Second, an explanatory analysis is designed to assess the importance and influence of the student features. Third, we implement an XGBoost model with hyper-parameter tuning to find the best classifier possible.

For this work, we have followed the same guidelines as our last paper (Blasco-Blasco et al., 2021). Our dataset, provided by SEA with 2975 students, is composed of the same five dimensions: Academic, Cognitive, Economic, Health, and Social. Therefore, we have a set with five features that describes the initial conditions of the UIS students. In a way, such a combination of variables is supposed to mark on the course of events during the first year at the university. In Parada et al. (2019), the authors proved that using this set could lead to interesting studies about the academic achievements of the institution via adequacy indicators. Therefore, we have been trying to make the best use of the given data.

Once the semester is finished, we can keep the final marks of the students in order to verify whether they have passed the tests. In our particular case, we have analyzed the final grades of calculus and algebra subjects. Both variables range between 0 and 5 so it is assumed that a student with an average lower than 2.5 have not passed the mathematical test, and so we consider it as a failure. Then, we define the binary variable Y, which determines whether a student has passed the course or not as shown in Eq. 7.1.

$$y_i = \begin{cases} 1 & \text{if} \quad \frac{1}{2}(Algebra + Calculus)_i < 2.5 \\ 0 & \text{otherwise} \end{cases}, \text{ for all } \ 1 \leq i \leq N. \tag{7.1}$$

Where $(Algebra + Calculus)_i$ is the sum of the grades in algebra and calculus subjects for the i^{th} individual.

Once we have calculated the y vector, we noticed that the proportion of our sample is almost $3:7$ with regard to the failure event. More specifically, we have 1099 failure subjects (70.71%) and the remaining 2435 passed the course (29.29%). Although it is understood as good news for the university, due to their great work, now it leads to an imbalanced learning problem. There are many methods to face such problems, in which we can distinguish two procedures: undersampling and oversampling. Given that we want to pay full attention to every single element in the sample, we decided to apply an oversampling technique. Moreover, as we wanted to cluster the students regarding their academic failure, we would like to emphasize the $\{y_i = 0\}$-group. That is why we decided to apply the ADASYN algorithm. As we will detail in subsection 7.1.3.3, this method generates copies of those cases that are hard to learn during training, so we are highlighting the excluded classes.

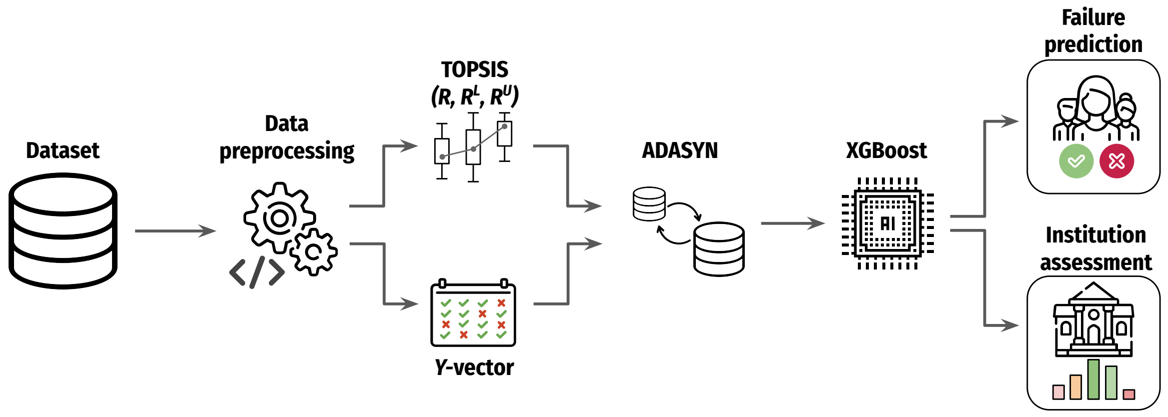


Figure 7.2: Flowchart of the procedure conducted in this paper. It involves the pipeline of collection, preparation, normalization, feature extraction, model fitting, and output presentation. Source: Own elaboration.

7.1.3.1 *Normalization*

It is well known that data must be properly normalized before the computations are applied. As we have described in § 3.2.2 and § 3.4.2, a normalization technique is needed to implement both methods. Therefore, the normalizations to be used are the same as we implemented by the authors Parada et al. (2019), in which the following expressions give the functions η and ξ:

$$\eta_{A,a,b,B;k_1,k_2}(x) = \begin{cases} \dfrac{1-e^{k_1\frac{x-A}{a-A}}}{1-e^{k_1}} & \text{if} \quad A \le x < a \\ 1 & \text{if} \quad a < x < b \\ \dfrac{1-e^{k_2\frac{B-x}{B-b}}}{1-e^{k_2}} & \text{if} \quad b < x \le B \\ 0 & \qquad \text{otherwise} \end{cases} \tag{7.2}$$

$$\xi_{A,a,b,B}(x) = \begin{cases} \dfrac{x-A}{a-A} & \text{if} \quad A \le x < a \\ 1 & \text{if} \quad a < x < b \\ \dfrac{B-x}{B-b} & \text{if} \quad b < x \le B \\ 0 & \qquad \text{otherwise} \end{cases} \tag{7.3}$$

Due to the definition of the dimensions, the (A, a, b, B) array represents the trapezoidal fuzzy shape, and the (k_1, k_2) coefficients are the left and right exponents that determine the convexity or concavity of the function. Table 7.1 shows the transformations of the data and the kind of normalization employed per each feature.

Table 7.1: Features, DDBB values, range transformations, ideal elements, and normalizations.

Dimension	Original	Transf.	Ideal	Normalization
Academic	{VL, L, LM, M, MH, H, VH}	$[1, 7]$	$[6, 7]$	$\eta_{1,6,7,7;1,0}(x)$
Cognitive	{VL, L, LM, M, MH, H, VH}	$[1, 7]$	$[6, 7]$	$\eta_{1,6,7,7;-1,0}(x)$
Economic	$[0, 1]$	$[0, 1]$	$[0.8, 1]$	$\xi_{0,0.8,1,1}(x)$
Health	$[0, 0.65]$	$[0, 0.65]$	0.65	$\xi_{0,0.65,0.65,0.65}(x)$
Social	$\{0.1, 0.5, 0.7, 1\}$	$[0.1, 1]$	$[0.7, 1]$	$\xi_{0.1,0.7,1,1}(x)$

7.1.3.2 *MCDM feature extractor*

In order to evaluate the initial scenario of each particular student, we have implemented both TOPSIS and uwTOPSIS techniques. On the one hand, the TOPSIS method has been applied as in our last paper (Blasco-Blasco et al., 2021) because the SEA-UIS has meticulously chosen the weighting scheme. This advantage allows us to understand how the institution notices the initial stage of people and then adds it to our dataset. On the other hand, the uwTOPSIS has followed our last contribution to the academic performance analysis in the UIS (Blasco-Blasco et al., 2021). By applying a rank reversal approach, we can extract the (R_i^L, R_i^U)-pairs per each alternative and consider it as an additional feature that evaluates the a priori global situation of the set of students. The required ideals in the Step 2 of the unweighted algorithm have been fixed as $PIS = (1, \ldots, 1)$ and $NIS = (0, \ldots, 0)$ to avoid an output that depends of the entire dataset. In other words, we wanted to preserve their limitations in terms of educational attributes.

Despite having three different features (R_i, R_i^L, R_i^U) that might seem highly correlated, what we have merged with this process is an indicator of institution insight plus their relative boundaries in which it varies. Although many other features could be included in the model to improve the performance, we have decided to follow the guidelines of the managers of the UIS. In turn, we have designed a dataset with size $N = 2975$ and $M = 8$ features, so each X_i array can be decomposed as $(A_i, C_i, E_i, H_i, S_i, R_i, R_i^L, R_i^U)$ to examine their impact in the classification task.

7.1.3.3 *Adaptive Synthetic Sampling Approach*

ADAptive SYNthetic (ADASYN) sampling approach is an over-sampling algorithm for learning from imbalanced datasets (He et al., 2008). The main idea of this technique is the use of a weighted distribution over the minority class examples according to their difficulty in learning. Then, ADASYN generates synthetic data elements that reduce the learning bias and the decision boundary of such samples.

Considering the dataset $D = \{X_i, y_i\}_{i=1}^N$, we distinguish the minority (m) and majority (M) samples as N_m and N_M respectively, so that, $N_m \leq N_M$ and $N_m + N_M = N$. Then, the ADASYN oversampling technique may be applied following the steps, in which we can assume, without limiting the generality of the foregoing, that the dataset is sorted so that the first N_m elements belong to the minority class.

1. Define $G = (N_M - N_m)\mu$ as the number of elements to generate, with $0 \leq \mu \leq 1$ the desired proportion to oversample.

2. Compute the KNN algorithm over the minority class to get the values $\{r_i\}_{i=1}^{N_m}$, where $r_i = \Delta_i / K$ and Δ_i is cardinal of the neighbors of X_i that belongs to the majority class.

3. Normalize $\{r_i\}_{i=1}^{N_m}$ as $\overline{r}_i = r_i / \sum_{i=1}^{N_m} r_i$ per each $1 \leq i \leq N_m$.

4. Generate $g_i = \overline{r}_i G$ samples of each $i \in \{1, \ldots, N_m\}$ instance, providing a μ-balanced dataset.

As a consequence of the step 3, we notice that $||\overline{r}||_1 = 1$, hence we have build a density distribution of $\{r_i\}_{i=1}^{N_m}$. Moreover, a new synthetic dataset can be generated with a partial proportion over the majority class, as stated in step 1.

For the case study, we have set $\mu = 1$ to create a fully balanced dataset. In such a way, we have just oversampled the less representative students labeled as "failure" so as to learn their distribution and adapt the intelligent system to their marginal situation.

7.1.3.4 *Model Implementation*

In order to compare and contrast the results obtained by the XGBoost classifier, we have decided to implement other classification tree methodologies to carry out an in-depth analysis. The models that have been trained with the Colombian set of students are XGBoost, GBoost, Random Forest, and Decision Tree. These all have been implemented by using the Sklearn library in Python.

7.1.3.5 *Evaluation metrics*

The binary classification of students is a difficult task and it can be explained by means of various factors. First, the actual behavior of students is somehow chaotic because school dropout is a major concern in public universities. Second, the annotations associated with each student can be subject to uncertainties due to the number of different entities that handle such values. Third, we are just feeding our model with data stored by the university. Therefore, we are disregarding relevant historical information that concerns each individual.

Anyway, the data collection has been duly collected, annotated, and stored by the SEA-UIS, thus facilitating the goals of this paper. Nevertheless, we are facing a problem of imbalanced learning due to the number of students that passes the course being considerably greater than the ones who fail. In fact, it made us apply the ADASYN technique for oversampling. As a result, we have to utilize metrics that appropriately evaluate the model performance. In order to understand the notation used in the following equations, table 4.1 shows how a confusion matrix splits the output of a classifier.

The metrics that will be used for model evaluation are defined by the following equations:

$$\text{Precision}: \quad P = \frac{TP}{TP+FP} \qquad F_\beta\text{-Score}: \quad (1+\beta^2)\frac{P \cdot R}{\beta^2 P + R}$$
$$\text{Recall}: \quad R = \frac{TP}{TP+FN} \qquad \text{Accuracy}: \quad Acc = \frac{TP+TN}{TP+TN+FP+FP} \tag{7.4}$$

In our particular case, we will utilize F_1-Score in which P and R metrics will help us understand the value's representation. The choice of $\beta = 1$ gives the same importance to sensitivity and specificity as most state-of-the-art studies. In a complementary way, the accuracy metric will describe the global performance of our model. This value will be relevant if we need to generalize the results by placing the same value on both groups (Pass-Fail). In addition, we also have to cover the misclassification errors that may occur (Bradley, 1997). Then, we select the AUC score since it is threshold invariant and describes the relation between hit and error.

7.1.4 *Results and discussion*

In this section, we present and describe the results obtained by evaluating the validation sets. We have basically faced three problems: Imbalanced learning, learning over synthetic datasets, and hyperparameter tuning for optimized model configuration. The implementation of the three cases is graphically described in Fig. 7.3, where the workflow is segmented per each problem.

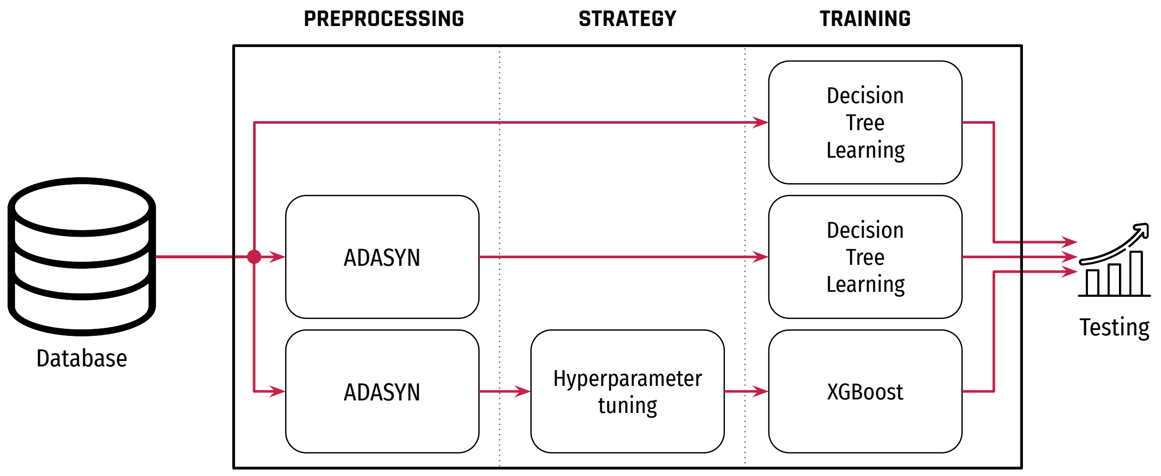


Figure 7.3: Workflow of the case study divided into three stages: top for imbalanced learning, middle for synthetic balanced learning, and bottom for hyperparameter tuning strategy. Source: Own elaboration.

7.1.4.1 *Classification task*

The sample of 2975 students has been serialized to get our experimental dataset $\{\mathbf{X}_i, y_i\}_i$, where each $\mathbf{X}_i \in \mathbb{R}^8$. For the training data split, we have decided to put aside the 20% of the data (595 students) for the evaluation set so that this number of elements that have not taken part in the fitting procedure can be analyzed a posteriori for high-reliability experiments. Hence, the following tables (7.2 and 7.3) contain the evaluation metrics associated with students that belong to the out-of-train set.

Table 7.2: Evaluation of the classification task with the evaluation set of the imbalanced strategy.

	Precision	Recall	F_1-score	Accuracy	AUC
XGBoost	0.4118	0.0524	0.0930	**0.6912**	**0.6035**
GBoost	0.3721	0.0599	0.1032	0.6855	0.5994
Random Forest	**0.4486**	**0.1798**	**0.2567**	0.6855	0.5714
Decision Tree	0.4196	0.1760	0.2480	0.6776	0.5534

Table 7.3: Evaluation of the classification task with the evaluation set of the ADASYN oversampling strategy.

	Precision	Recall	F_1-score	Accuracy	AUC
XGBoost	**0.8206**	0.5639	0.6685	**0.7080**	**0.7677**
GBoost	0.8205	0.5562	0.6630	0.7047	0.7658
Random Forest	0.7863	**0.5840**	**0.6702**	0.6999	0.7605
Decision Tree	0.7987	0.5501	0.6515	0.6927	0.7380

7.1.4.2 *Hyperparameter tuning*

In order to get the best performance possible, we implemented a hyperparameter tuning strategy. It is essential in most machine learning implementations because when more accurate, more applicable to further data. With this aim in mind, we have defined a computational grid for some of the basic parameters that XGBoost requires for its fitting. Two elements to take into account for boosting are the maximum depth of the trees (D_{max}) and the number of decision trees (DTs) to fix for our model. As we discussed in subsection 4.1.5.3, equation 4.91 presents two parameters for regularization γ and λ. Finally, the learning rate (η) is always a value to select meticulously so as to balance the learning gain during training. In table 7.4, the possible values have been presented for the cross-validation search technique, which implies 3520 different combinations. Moreover, the last column of table 7.4 indicates the optimal solution of the discrete choice values.

With the exception of the DTs parameter, the selected values were affordably selected with a usual scheme of values. It is clear that the grid could have been extended to a high cardinality, however, the computational limitations have led us to bound the complexity of the problem. For the number of decision trees implemented, we empirically discover that under a value of 200, we faced an underfitting response, while with a value over 300, we got overfitting with no generalization techniques. Therefore, we designed a uniformly distributed sequence with differences of 10 units per step.

Now, by considering the tuned configuration described in table 7.4, the validation results of the tuned XGBoost outperform the last trained models. In particular, the resultant

Table 7.4: Grid with the hyperparameters involved during the cross-validation search.

Parameter	Values	Optimal
D_{max}	5, 6, 7, 8, 9	7
DTs	$\{200 + 10i\}_{i=0}^{10}$	250
η	0.1, 0.05, 0.02, 0.01	0.1
γ	0.075, 0.1, 0.125, 0.15	0.1
λ	0.2, 0.25, 0.3, 0.5	0.25

array $(F_1, \text{Acc}, \text{AUC})$ is positioned beyond the last ones, indicating better behavior when classifying, generalizing, and avoiding mismatching.

Table 7.5: Evaluation of the classification task with XGBoost tuned as table 7.4 states.

	Precision	Recall	F_1-score	Accuracy	AUC
XGBoost tuned	0.8333	0.6028	0.6996	0.7305	0.7915

For a further assessment of the tuned classifier, we need to plot the rates of success-failure of the values of 4.1 as a way to represent and complement the evaluation metrics.

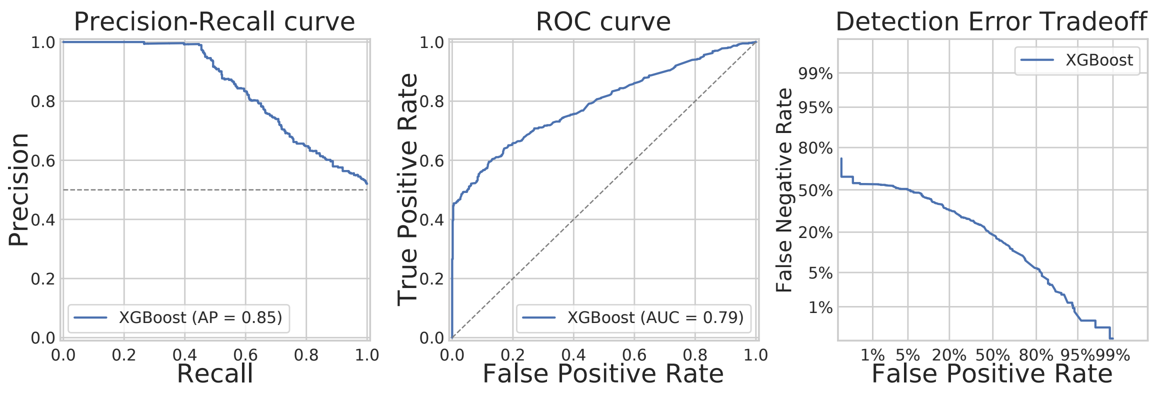


Figure 7.4: Evaluation curves of the XGBoost after the hyperparameter tuning. From left to right, it is shown: First, the Precision-Recall tradeoff for different probability thresholds with its average precision (AP). Second, the ROC curve with its AUC value (AUC). Third, the Type I and Type II error tradeoffs in percentage, known as Detection Error Tradeoff. Source: Own elaboration.

7.1.4.3 *Feature importance*

The classification has been conducted through the sequential process of the input features **X**. Once the training phase is done, we can study the impact of the model of the selected set of variables $(A_i, C_i, E_i, H_i, S_i, R_i, R_i^L, R_i^U)$. As we explained in section 4.1.5.3, the XGBoost method is a GTB ensemble architecture, then we can extract the resultant assessment of each feature when classifying. The table embedded into the Fig. 7.5 shows the importance of each feature for the trained models.

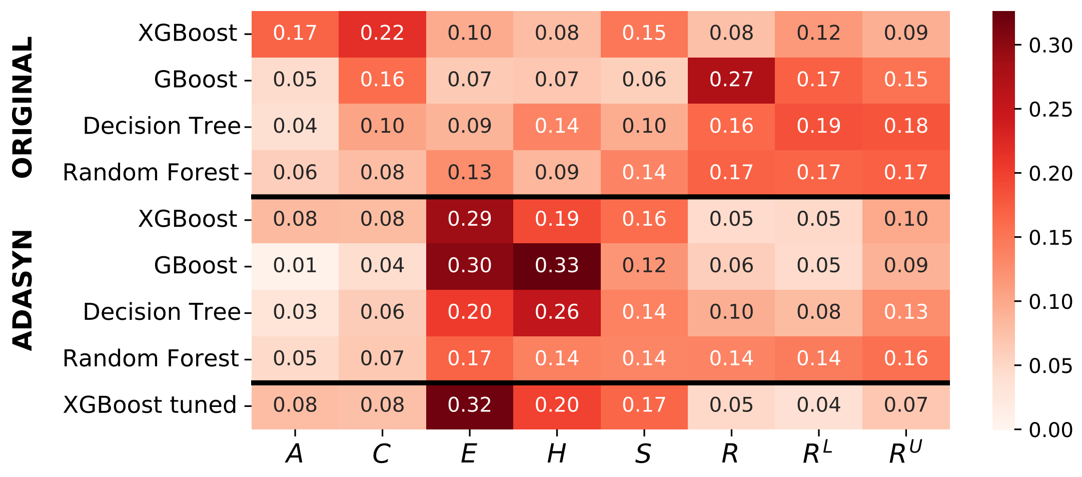


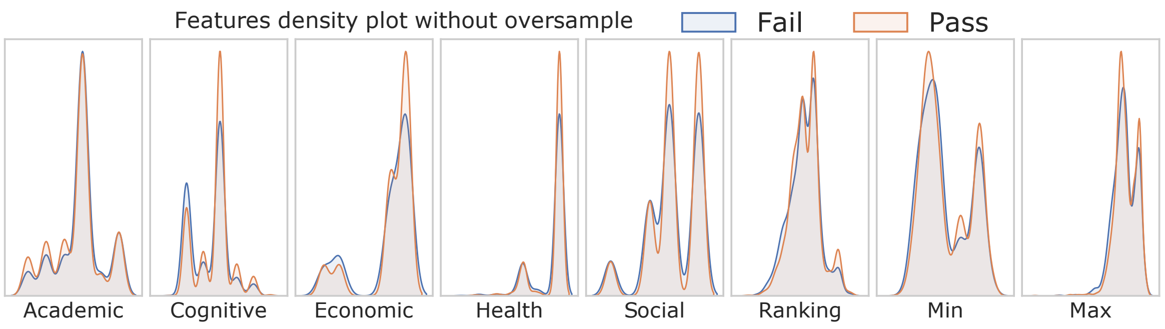
Figure 7.5: Feature importance illustrated as a heat map of the model weights with values from 0 (white color) to 1 (red color). Rows from 1 to 4 correspond to the results for the original dataset. Rows from 5 to 8 correspond to the results with the ADASYN technique. The last row is the combination of ADASYN plus hyperparameter tuning.
Source: Own elaboration.

One of the most significant points is the relevance gains associated with the economic and health dimensions. For the economic variable, the minimal value achieved has changed from 0.0892 to 0.1689, which is almost a twice increment, and for the health one, the change was from 0.0798 to 0.1912, i. e. more than double. For the maximal values, we can note in the case of GBoost that the combination of both means varied from 13.45% to 62.97%, which means an increase of its predictive power of 49.52 percentage units. We can also see a clear decreasing impact of the MCDM relative proximity values (R, R^L, R^U) when applying the synthetic oversample. Nevertheless, the sum of them means the 19.63% of the decision in its lowest case (XGBoost) and 44.01% in its greatest case (Random Forest).

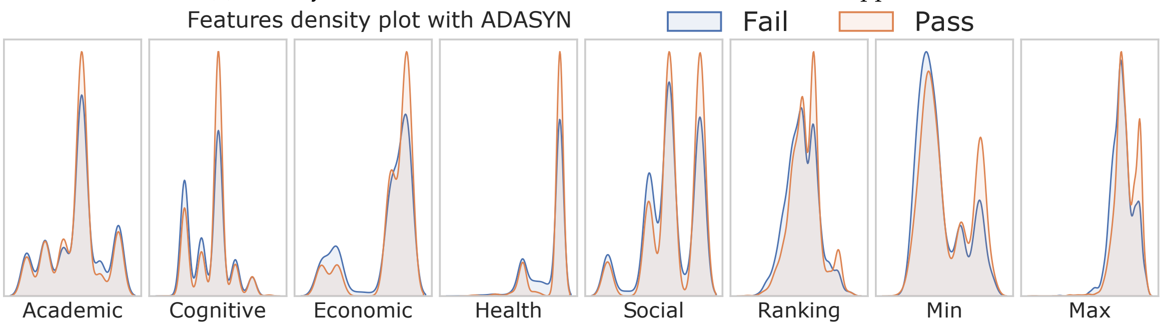
It is interesting to check that before the oversampling technique was applied, the cognitive variable got a significant impact on the predictions. For the XGBoost case, more than the fifth part of the bundle depended on the C variable. On another note, the most stable feature is the social one because its variation has not been compromised by the ADASYN technique.

For the particular case of the optimized extreme gradient boosting model, it is necessary to emphasize that over half of the forecast impact (53.41%) is due to the economical-sanitary situation of the student before their entry into the university. By adding the social dimension, it raises to 65.48%. Since none of them are directly correlated with the academic effort, it makes us think, once again, that marginal ditching is a major concern in countries such as Colombia because our model has succeeded in the 71.12% of the cases.

For a further visualization that illustrates the variation between feature importance, Fig. 7.5 shows a positional heat map of such values in a scale of $[0, 1]$. All the variations previously mentioned have been reflected in such a chart. Due to the color mapping, we can see that the decision-weighted scheme varied from the (R, R^L, R^U) TOPSIS-array (imbalanced set) to the (E, H) features (synthetic set). As far as classifiers are concerned, Random



(a) Density distribution of the features with the imbalanced approach.



(b) Density distribution of the features with ADASYN approach.

Figure 7.6: Comparison between feature distributions with the different approaches implemented. Source: Own elaboration.

Forest has been the model with minor modifications of their relative importance weights after the oversampling. It might explain why it has been the model with the best R and F_1 score in both approaches.

Regardless of the results obtained, we can say that the TOPSIS feature extraction has been very supportive of the classification problem. When we had no balance over our sample of students, the three values contributed to finding the correct response. Once we obtained a balanced case, even though their impact decreased, they still greatly impacted the decision making. On average, their importance meant the 28.50% of the weighted system.

Finally, we want to check the impact of the ADASYN technique on our dataset. To this effect, Fig. 7.6 shows the difference with regard to density before (Fig. 7.6a) and after (Fig. 7.6b) the ADASYN algorithm is applied.

7.1.5 *Conclusions*

One of the keys to academic success is the early detection of difficulties so that it allows the institutions to execute actions to correct them. With this in mind, machine learning implementation enables the estimation of future outcomes that could help academic institution managers to take actions that increase academic performance.

In this paper, we have developed a decision-support system that predicts academic failure considering only information from students prior to their entry into the university. Our system consists of an XGBoost predictor fed by SEA-UIS data and TOPSIS-based features.

In addition, the underlying imbalanced problem has been handled by implementing the ADASYN technique. For the predictive task, we have achieved a precision of 0.8333. When comparing our predictive model with other baselines, we have obtained better performance in comparison with other decision tree models. In this manner, our systems enable the implementation of a priori countermeasures for the Universidad Industrial de Santander.

7.1.6 *Python implementation*

The script that performs the process for training the dataset and obtaining the results shown in 7.1.4.1 is presented in Script 15 (§ C.1) as a continuation of Script 3 (§ A.2).

7.2 MCDM & AI-CASE 2: MULTIPLE-CRITERIA DECISION MAKING APPROACH FOR AN IN-DEPTH BENCHMARKING OF SUPERVISED MACHINE LEARNING MODELS

This case study is presented as a continuation of the problems studied in § 6.2 and § 6.3. Our aim is to present an MCDA scheme to select the best possible AI model from a set of alternatives assessed in different aspects regarding their performance. Then, the underlying problems are solved by means of artificial intelligence models, but the final step is approached with multiple-criteria decision-making techniques.

7.2.1 *Experiment Setup*

In the field of artificial intelligence, the evaluation process is one of the main stages before the model deployment. Owing to the complexity associated with the supervised machine learning techniques, there exist multiple manners to carry out the evaluation phase. In such a task, the goodness of fit through measures of the model's predictive power has been widely studied. When comparing the performance among different algorithms, methodologies, and/or ensembles; most of the final decision is usually given by a single statistic. However, it would be a big mistake if the model selection just depended on a single attribute. Furthermore, it is important to take into account that not only accuracy should be considered when evaluating, but also other attributes such as computational cost and size that affect the implementation of the AI solution. For this reason, benchmarking machine learning algorithms is a complex and time-consuming task. In this case study, we have designed an automatic work routine that contrasts the different results after an in-depth empirical analysis of supervised models. We propose an integrated end-to-end system that generates a k-fold cross-validation from a given dataset and a subsequent fuzzy unweighted multiple-criteria decision-making approach as a ranking system.

7.2.2 *Literature review*

The comparison of supervised algorithms haven approached in different ways. With an experimental purpose, Belavagi and Muniyal (2016) reported the performance results of basic data mining classifiers for automatic intrusion detection. In a similar way, Sujatha and Mahalakshmi (2020) and Malakar et al. (2018) conducted the model evaluation followed by a graphic comparison. A major contribution was given by Peng et al. (2008), where the authors designed a novel multiple-criteria mathematical programming scheme for model benchmarking. Withing the Multiple-Criteria Decision Analysis approach, Kou et al. (2012) utilized various MCDM methods with a final use of Spearman's rank correlation coefficient to resolve differences among the methods. Other authors as Mohammed et al. (2020), Malik et al. (2022), and Shakor (2022) implemented single techniques, Entropy, AHP-TOPSIS, and TOPSIS respectively, for the final selection of the best ML model. Motivated by the analysis of uncertainty in data, Sahu et al. (2015) used Fuzzy-MCDM, in particular fuzzyTOPSIS, considering linguistic qualitative criteria by means of trapezoidal fuzzy numbers. In the same line, Awodele et al. (2020) implemented fuzzy-AHP for triangular-fuzzy numbers and Salih et al. (2022) for Fermatean fuzzy sets. Out of the outranking MCDM methodology, Nandy and Singh (2021) implemented a fuzzy-DEA scheme for calculating efficiency ratios of the different data mining alternatives. Apart from MCDA methodology, Ali et al. (2022) analyzed the robustness of machine learning algorithms by means of data perturbation for genome sequence classification of SARS-CoV-2.

As we indicated, the benchmarking task requires long runtimes together with a high programming load. So, this stage is generally avoided or even omitted from the AI-operating schedule. As a manner to solve this issue, there have been various attempts at software automation. In particular, we would like to highlight PyCaret (Ali, 2019) and LazyPredict (Pandala, 2019). Both libraries prepare the data pipeline and a training phase for various ML algorithms depending on the objective of the problem, but they present a final report in distinct ways.

7.2.3 *Methodology*

For the remaining of the case study, it is assumed that the reader has fully understood the methodology presented in § 6.2.2 and § 6.3.3 since here we discuss topics with respect to both cases. The methods implemented to carry out the problems differ with respect to the experimental datasets. This distinction is due to the different machine-learning approaches. Despite the fact that both problems have a supervised background, one is a regression task and the other is a classification task. The illustration of the step-by-step methodology conducted is shown in Fig. 7.7.

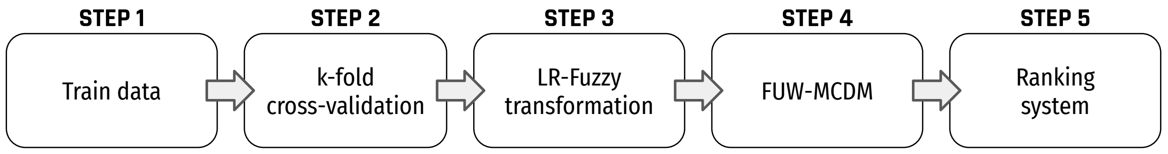


Figure 7.7: Description of the methodology implemented throughout this case study as a flowchart. Source: Own elaboration.

7.2.3.1 *k-fold cross-validation*

In the field of statistics, the estimation of prediction error is crucial when evaluating a fitted ML algorithm (Ljung, 2002). A widely utilized method for estimating such errors is cross-validation, which validates the predictive effectiveness of a statistical learning model (Browne, 2000). The first known application of cross-validation was presented in Mosier (1951), where the author evaluated a linear regression equation. In artificial intelligence, the selection of the best-performing model is usually made by means of cross-validation (Yang, 2007). Although, it is worth mentioning that such a search criterion would just lead to the best empirical model evaluated rather than a better manner for estimating the prediction error (Fushiki, 2011).

Even though many researchers apply the so-called leave-one-out cross-validation, in which the train and test sets are fixed during the entire fitting process, it is more than recommendable to implement the k-fold cross-validation (Fushiki, 2011). The motivation of the k-fold is to split the training sample in $k \in \mathbb{N}$ different ways so that the cross-validation can be conducted for the k splits. It is easy to note that this strategy is more profitable from a computational point of view.

In Fig. 7.8, we have graphically represented our particular implementation of k-fold cross-validation for our case studies. The only difference between the cases is the set of algorithms chosen for the experimental comparison.

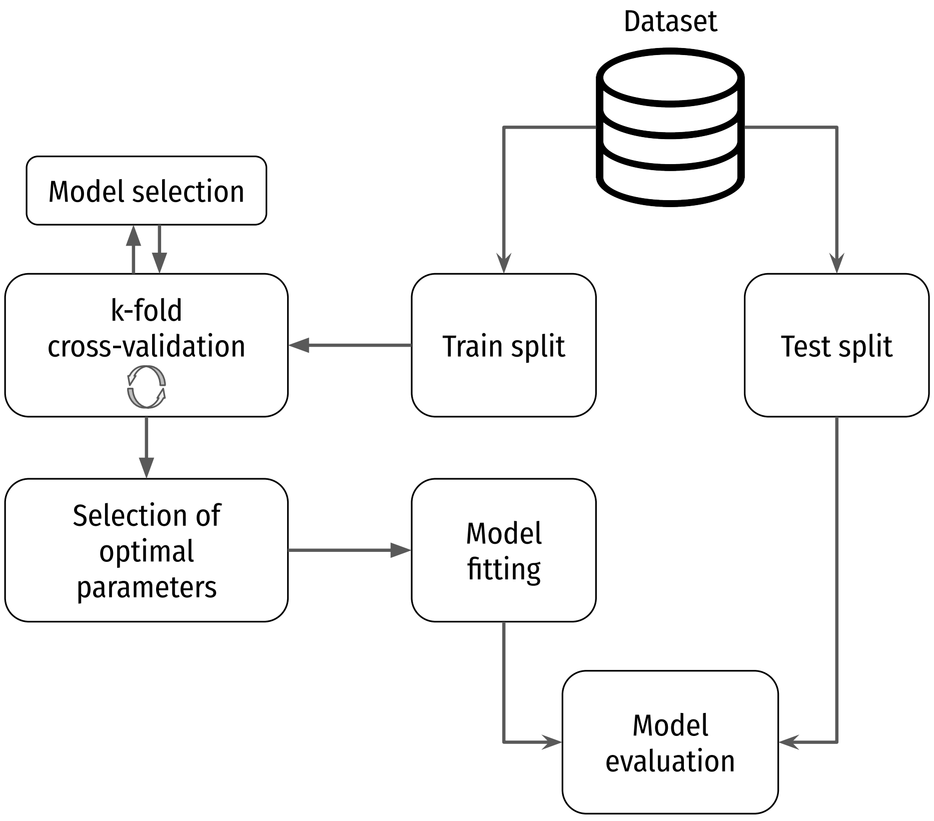


Figure 7.8: Flowchart to implement a k-fold cross-validation procedure from a given dataset. Source: Own elaboration.

Motivated by the composition of the CIPIC dataset, in both problems (regression and classification) we have set $k = 9$. The main difference is that the fold partitioning for CIPIC is made by taking into account the subject in the study, whilst for the AstraZeneca case the data split has been randomized.

7.2.3.2 *LR-Fuzzy transformation*

Once the k-fold cross-validation has finished, we obtain a k different set of results. In most cases, the user just considers the top-performing model according to a certain measure. However, this criterion is not a good practice due to the selected model has solely shown the best performance in an empirical way, as we discussed in § 7.2.3.1. In this way, such misconduct would lead to a static solution in which just one experiment is contemplated.

In our case, we have decided to take into account every single result within the k-fold cross-validations technique, thus leading us to a more integrated evaluation system. For this purpose, we implement a fuzzy transformation (as described in § 3.3) from the set of scores $S = [s_{ij}]$, in which i ranges from one to the number of experiments and $j \in \{1, \ldots, k\}$. The algorithm for fuzzy transformation is presented as follows:

Step 1 Extract the score components after k-fold cross validation per each alternative as $S = [s_{ij}]$ where i stands for the number of alternatives and j for the number of k-fold cross-validations conducted.

Step 2 Per each alternative, sort the vector $(s_{i1}, \ldots, s_{ik})$ in ascending order, i. e. as a new vector $R_i = (r_{i1}, \ldots, r_{ik})$ so that $r_{i1} \leq \cdots \leq r_{ik}$.

Step 3 Transform to trapezoidal fuzzy number as $\tilde{T} = (T_1, T_2, T_3, T_4)$, where

$$\begin{cases} T_1 &= \max\{r_{i1},\ \mu_{R_i} - \frac{1}{2}\sigma^2_{R_i}\}, \\ T_2 &= \min\{r_{ik},\ \mu_{R_i} + \frac{1}{2}\sigma^2_{R_i}\}, \\ T_3 &= \min\{r_{i1},\ \mu_{R_i} - \frac{1}{2}\sigma^2_{R_i}\}, \\ T_4 &= \max\{r_{ik},\ \mu_{R_i} + \frac{1}{2}\sigma^2_{R_i}\}, \end{cases} \tag{7.5}$$

in which the components of T are the vertexes of the support of its membership function, instead of the LR-fuzzy representation previously defined. In order to ease the notation, μ_{R_i} and σ_{R_i} stand for the mean and standard deviation of R_i respectively.

Step 4 Construct the fuzzy decision matrix $\tilde{X} = [\tilde{x}_{ij}]$ per each alternative i and cross-validation j.

7.2.3.3 *Fuzzy Unweighted Multiple-Criteria Decision Making as ranking system*

Given the corresponding decision matrix, $\tilde{X} = [\tilde{x}_{ij}]$ so that $\tilde{x}_{ij} = (x^L_{ij}, x^R_{ij}, \alpha^L_{ij}, \alpha^R_{ij})_{L_{ij}R_{ij}}$ from the set of scores $S = [s_{ij}]$, our aim is to select the best machine learning model from the set of alternatives. As we discussed in Chapter 3, the outranking field of Multiple-Criteria Decision Making gives us tools for sorting a set of alternatives when there exist multiple conflicts of interest. In this way, we can apply an outrank methodology for helping final users to select the appropriate alternative for their needs.

In this particular case study, we have implemented a fuzzy transformation in order to ascertain the uncertainty of the functionality of a model. In addition, we did not want to attach our personal bias to the final decision, since it would corrupt the resulting ranking. So, we have incorporated an unweighted technique to avoid direct assignment of importance per each attribute. As a result, we have implemented the FUW-MM algorithm presented in § 3.5 to carry out the decision-making stage.

For an additional comparison among the FUW-MM outputs, we have selected 6 different configurations. They are the result of combining the exponent $p \in \{0, 1, 2\}$ in Step 5 and the operator Φ as the vector ℓ^2 normalization and Min-Max normalization (see Table 3.1) in Step 4. Furthermore, the fuzzy order relationship $\preceq$ to obtain the score from the $\tilde{\mathcal{M}}^p_i$ is computing the distance from LR-fuzzy zero $(0, 0, 0, 0)$ to $\tilde{\mathcal{M}}_i^p$. Then, when the higher the distance is, the more valuable the score is.

7.2.3.4 *Feature selection for the ranking system*

We have decided to select four evaluation metrics that measure the fitting of the predictions. They are indicated in the following list:

- Regression task: Root Mean Squared Error (RMSE 4.50), Mean Absolute Percentage Error (MAPE 4.52), Median Absolute Error (MedAE 4.54) and R^2 (4.58).
- Classification task: accuracy (ACC 4.36), Fowlkes–Mallows index (FM 4.44), Matthews correlation coefficient (MCC 4.46) and AUC (4.47).

Moreover, other computer-based attributes have been collected to complete the MCDM stage. Regardless of the task performed, we have monitored the times (in seconds) associated with:

- Training: Time required for loading, initializing, and fitting the model.
- Predicting: For our entire dataset (train and set), the time needed to process all the input samples.
- Saving: Time required for saving the model in our local working directory.

Finally, we have stored in the working space each model once the fitting and prediction have been conducted. Then, we have the size of each model represented as the attribute:

- Storage: Size in MB of the models stored locally.

7.2.4 *Forecast of the AstraZeneca close price with the use of stock market regressors*

The experimental dataset utilized for this section is the same as the one presented in § 6.2. It has been tracked and gathered the stock features of AstraZeneca from 2018 to the second semester of 2021. As a result, we have stored data of the company creating a time series of 1250 days with 10 different variables. As well as the past study, we have done a feature selection of the open, net, and volume regressors in order to make a regression for predicting the close price of the company. Hence, the dataset is presented as $\mathcal{D} = \{X_i, y_i\}_{i=1}^{1250}$, in which $X_i \in \mathbb{R}^4 \times \mathbb{R}^5$. The data pre-processing is also the same, i. e. there has been applied the statistical typification of Eq. 6.15 to normalize data. The experiments are divided into two stages. First, an experimental proof in which AstraZeneca's close price is studied as a decomposable time series. Second, the methodology described is implemented over a set of regressor models.

7.2.4.1 *Preliminary test: Regression analysis via additive models*

Given that the information on the stock market has been tracked day-by-day, it is interesting to see whether the time series prediction can be approached by means of additive models for non-linear regression. To this effect, we apply Prophet (Taylor and Letham, 2018), a tuneable software robust to missing data and seasonal effects that produces forecasting at scale. It is an additive model based on three components: trend, seasonality, and holidays. They are combined in the following equation:

$$\hat{y}_t = g(t) + s(t) + h(t) + \varepsilon_t. \tag{7.6}$$

To make the forecast, $g(t)$ is the trend function that models non-periodic changes in the dependent variable, $s(t)$ incorporates the periodic changes (i. e. seasonality), and $h(t)$ represents the effects of holidays. The error term ε_t is assumed to be normally distributed. Similarly, it could be reformulated in the multiplicative version.

In practice, the effectiveness of Prophet forecasting financial markets has been compared with other ML regressors such as ARIMA (Garlapati et al., 2021) or LSTM (Fang et al., 2019), as well as other applications such as cloud management (Daraghmeh et al., 2021) and waste generation in bitcoin mining (Jana et al., 2022).

For this preliminary test, we have applied the Prophet 1.1.1 version of Python over the AstraZeneca dataset. The motivation of this experiment is to ascertain whether the trend, seasonality, and holiday effect are good descriptive variables for regression analysis of the close price of the company. With this purpose, we have fitted the Prophet model considering the three stock regressors to forecast the close price of the pharmaceutical enterprise. In Fig. 7.9 and Table 7.6 are shown the forecasting and the evaluation respectively.

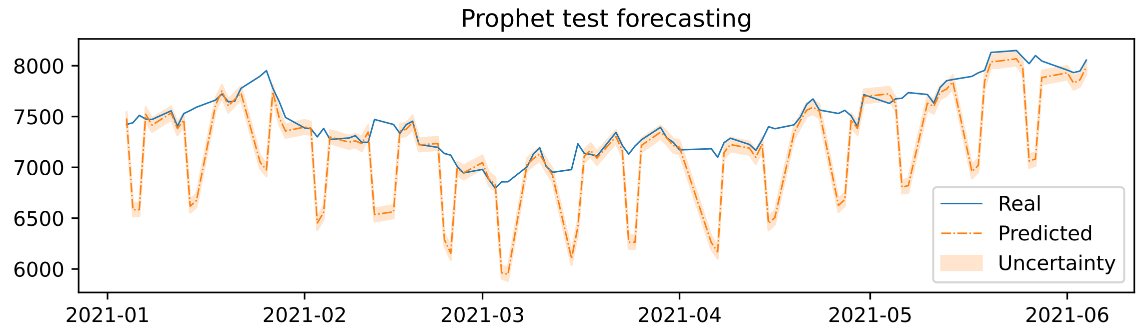


Figure 7.9: Prophet forecasting made for the testing dataset in which the uncertainty in the prediction has been considered.
Source: Own elaboration.

Table 7.6: Results obtained in the forecasting the test data of AstraZeneca for the Prophet model.

	RMSE	MAE	MedAE	MAPE	R^2
Prophet	484.62	291.65	59.95	39.11	-1.1626

The conclusion extracted from this preliminary test is that AstraZeneca is financially a strong company since its stock behavior cannot be estimated by means of seasonality or holiday patterns. As we can see, the forecast from 2021 is completely chaotic, in which even the MedAE achieves just under 60 dollars of error. Then, we can say that the Prophet additive model is not able to fit the company's behavior from the financial regression standpoint.

7.2.4.2 *Main problem: Selection of the best regression model*

The main purpose is to obtain the best fitting possible for the close price of the pharmaceutical company. To this effect, we have selected five different regression models in order to benchmark the performance among them. For this task, we have selected four tree-based machine-learning regressors and a Gaussian process regressor model. The tree regressors are Random Forest (RF, Ho, 1995), XGBoost (XGB, Chen and Guestrin, 2016), LightGBM (LGB, Ke et al., 2017), and CatBoost (CatB, Prokhorenkova et al., 2017). This selection is not trivial since this model set is commonly used for comparison purposes (see Natekin and Knoll (2013) or Bentéjac et al. (2021)). In the same line, we have selected a Gaussian process regressor model (GPR, Seeger, 2004) with radial basis function (RBF, Buhmann, 2000) kernel (for detailed information about GPR properties see Schulz et al., 2018). GPR models have shown great performance in various regression tasks (for instance: motion analysis Kim et al., 2011) and efficiency for large datasets (Banerjee et al., 2012). In addition, the use of gradient boosting machines and Gaussian processes as benchmark comparisons in data mining is quite extended in the ML literature (Yetilmezsoy et al., 2021).

For the configuration set up for the experiment, the loss function has been the root mean square loss for the gradient boosting algorithms except for the GRP, in which the L-BFGS-B[1] was selected (Zhu et al., 1997). In each data split, we have concatenated each component as a one-dimensional data input so that we turned four vectors in $\mathbb{R}^{30}$ into a single vector in $\mathbb{R}^{120}$. We have additionally modified the random state in each iteration by multiplying 13 by the iteration number.

Since the problem is given as a regression task, the k-fold splitting is not as simple as in the classification problem. In order to prevent non-significant outputs, we have defined a sliding window method with 8 days gap and size of 770 samples partitioned as $70:700$ relation between test and train. The sliding window strategy is broadly applied for time series forecasting as in Alberg and Last (2018) or Suresh et al. (2020). The results after training 9-fold cross-validation are displayed in Table 7.7 with an LR-fuzzy shape. The results are essentially stable, where the variation of the RMSE ranges from 37.98 (GPR) to 47.08 (XGB). However, it is remarkable the oscillation for the RMSE in the CatBoost algorithm, in which the maximum difference for the loss function was 100.65. Thus, the maximum standard deviation was also obtained by the CatBoost regressor, with a value of 33.17 units.

For adding visual representations, in Fig. 7.10 is depicted as the resulting prediction for the last cross-validation conducted for the experiment. We have decided to choose the last

1 The acronym stands for Limited-memory Broyden–Fletcher–Goldfarb–Shanno with Bounds algorithm.

Table 7.7: Fuzzy decision matrix of the regression models for the AstraZeneca dataset regarding evaluation metrics, processing times (seconds) with five significant figures, and computational sizes (MB). The values per each criteria are presented according to Eq. 7.5 after the 9-fold cross-validation process.

	RMSE				**MedAE**				**MAPE**				**R^2**			
Model	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4
XGB	114.22	122.81	137.18	161.31	62.197	70.433	81.621	99.000	1.2012	1.2781	1.3954	1.5703	0.6906	0.7426	0.8099	0.9079
LGB	108.22	118.26	133.58	151.53	69.619	80.093	89.479	100.380	1.1378	1.2638	1.4221	1.6131	0.6927	0.7641	0.8222	0.9048
CatB	110.65	125.31	158.49	211.30	73.577	86.920	110.468	144.206	1.1923	1.3630	1.7085	2.2469	0.5227	0.6699	0.7889	0.9077
RF	110.46	116.84	131.46	157.42	70.697	75.726	81.984	88.449	1.1927	1.2646	1.3619	1.5273	0.7351	0.7704	0.8255	0.9139
GPR	113.75	127.99	139.01	151.73	68.064	81.869	96.591	117.720	1.1324	1.3580	1.5299	1.6701	0.7041	0.7303	0.7969	0.9087

	Training (s)				**Predicting (s)**				**Saving (s)**				**Storage (MB)**			
Model	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4
XGB	1.5226	2.4342	2.8721	2.9309	0.0054	0.0058	0.0070	0.0093	0.0119	0.0130	0.0150	0.0191	0.8540	0.8579	0.8618	0.8674
LGB	4.8725	5.5865	7.1366	9.7139	0.0280	0.0321	0.0433	0.0603	0.0585	0.0641	0.0774	0.0949	2.7671	2.7768	2.7870	2.7958
CatB	9.4103	12.2631	16.2888	18.5829	0.0059	0.0081	0.0109	0.0133	0.0039	0.0061	0.0088	0.0106	1.1046	1.1048	1.1049	1.1051
RF	25.8278	27.0226	29.7354	32.6173	0.1282	0.1299	0.1490	0.1885	0.4436	0.5098	0.5854	0.6732	55.3713	55.5291	55.6091	55.6758
GPR	3.8146	6.2284	10.9514	19.6338	0.0773	0.0911	0.1072	0.1303	0.0067	0.0202	0.0316	0.0399	4.2449	4.2624	4.2682	4.2694

time series split for our sliding window because the output has a similar structure to the studied in § 6.2. Even though it is evident that the default configuration of the models implemented returned an overfitted prediction, our aim is to evaluate and benchmark the models under the same conditions. In any case, the five forecasts are good enough to carry out a benchmarking process.

Table 7.8: Results obtained in the forecasting the last cross-validation of AstraZeneca for the experimental models.

	RMSE	MAE	MedAE	MAPE	R^2
XGB	114.22	89.17	65.96	1.2011	0.9079
LGB	116.14	84.78	69.61	1.1377	0.9048
CatB	114.37	92.04	73.57	1.2359	0.9077
RF	110.46	88.46	76.57	1.1926	0.9139
GPR	113.75	83.62	68.06	1.1323	0.9087

In general, we can see a good test fit which it is shown the robustness of the algorithms. The evaluation obtained per each model is presented in Table 7.8. It is worth mentioning that the lowest R^2 obtained is higher than 0.9, indicating a great forecast in terms of adjusting the AstraZeneca stock behavior. Moreover, the highest percentage error in mean achieved has been lower than 1.25%, a value considerably low considering the close prices attached to the pharmaceutical.

When contrasting Table 7.8 with Table 7.7, we can notice that the last iteration for the 9-fold cross-validation was one of the best rated in terms of evaluation. With respect to the reference loss function, the minimum value was obtained in 3 of the 5 algorithms (XGB, RF, and GPR). For the remaining ones, the difference with the minimum was 7.92 for the LGB and 3.91 for the CatB. In either case, it is remarkable that in Table 7.8 the R^2 coefficient matches with the α_R values in Table 7.7, indicating the best result possible in the evaluation set of 9 validations.

Once we were sure that the 9-fold cross-validation returned acceptable results, we applied the FUW-MM technique for ranking the regression models. In this part of the benchmarking stage, we generated six different ranking systems in order to compare both the models and the new FUW-MCDM system. In Table 7.9 is displayed such ranking system disaggregated per normalization function and p-norm.

It is easy to note that the Extreme Gradient Boosting Machine has been the best-ranked regressor model, achieving the first position in every of the decision-making systems. In practice, the XGBoost model has shown excellent computational features, with the lowest times in every criterion but saving cost, even though it did not have good results regarding the predictive performance. For the rest of the models, it is worth noting the Random Forest case. Despite having obtained great fittings for the stock price, with the minimum MAPE and maximum R^2, its position in the ranking has placed in the last position due to the long delays in its implementation. Another similar example is the case of LightGBM, in which it succeeded in the prediction task, with the lowest loss for the common objective

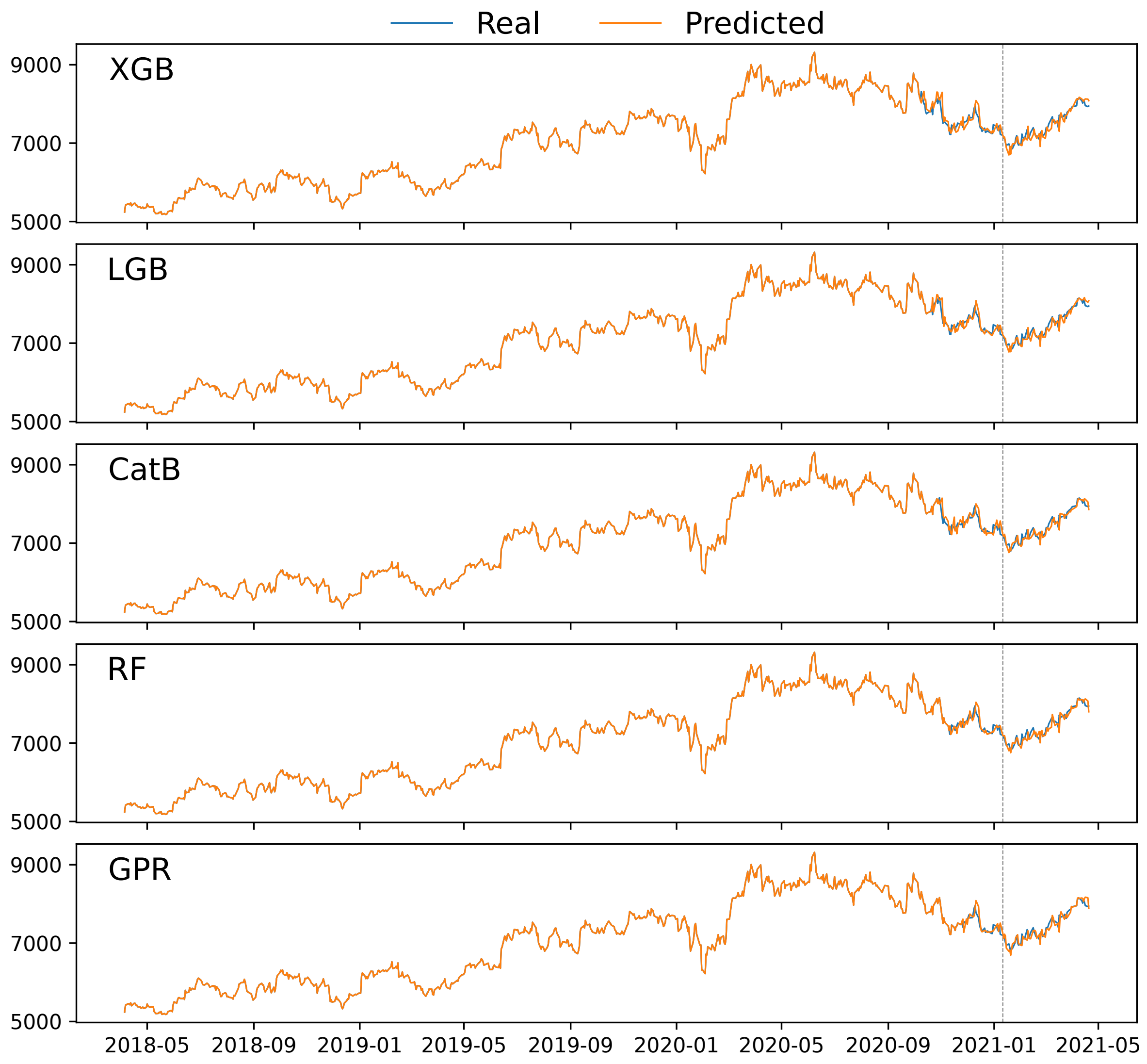


Figure 7.10: Forecasting of the close price of AstraZeneca's stock value for the different regressor models in the last cross-validation. The vertical lines indicate the test split for data. Source: Own elaboration.

Table 7.9: Score and ranking systems for the AstraZeneca regressor models by the FUW-MM technique disaggregated per normalization function and p-norm.

	ℓ^2 vector			**Min-Max**		
	$p = 0$	$p = 1$	$p = 2$	$p = 0$	$p = 1$	$p = 2$
XGB	1.5115 (1)	3.9256 (1)	5.6409 (1)	1.7859 (1)	4.6357 (1)	6.4149 (1)
LGB	1.4803 (2)	3.8726 (3)	4.8710 (3)	1.6614 (2)	4.4391 (2)	6.1067 (2)
CatB	1.4520 (3)	3.8876 (2)	4.2954 (4)	1.3328 (4)	4.3932 (3)	5.9011 (3)
RF	1.2438 (5)	3.0410 (5)	4.2747 (5)	0.4800 (5)	2.7616 (5)	3.7194 (5)
GPR	1.4428 (4)	3.8131 (4)	5.3648 (2)	1.5082 (3)	4.2935 (4)	5.8072 (4)

function (RMSE) but obtained the highest times in comparison with the other models. Anyway, the position of the LGB in the FUW-MM ranking has ranged between second and third place.

7.2.5 *Front-Back sound discrimination on HRTF data*

The experimental dataset utilized for this section is the same as the one presented in § 6.3. The 2500 measurements for the 45 subjects presented in the CIPIC dataset have been processed in the same way, however, we have now delimited the spherical region into two different sources: Front and Back. It is easy to note that now we face a binary classification problem in which the classes are complementary to each other, i. e. Bernoulli process. In order to ease the source localization, we have removed the lateral samples considered in the last experiments, so the lateral up and down have not taken part in this case study. As a consequence of such discrimination, it has been removed 200 positional areas from the original dataset, which means a loss of 9000 HRTF binaural signals from the 56250 that had been considered in § 6.3. Another significant difference with respect to the other problem is that we have now aggregated the ipsilateral and contralateral channels of the recordings via pair-wise addition of components in each frequency band as the authors in Zieliński et al. (2022).

Regarding the k-fold cross-validation, we have decided to set $k = 9$ because the original data split considered 36 subjects for training and 9 for testing. The idea is to present a more robust system since it is not mixed the different signals for different subjects. By doing so, during the fitting stage, the model has not seen any sample of any subject in the test set.

For a better understanding of the experimental procedure, Fig. 7.11 depicts the data preparation plus the k-fold cross-validation and the front-back source discrimination.

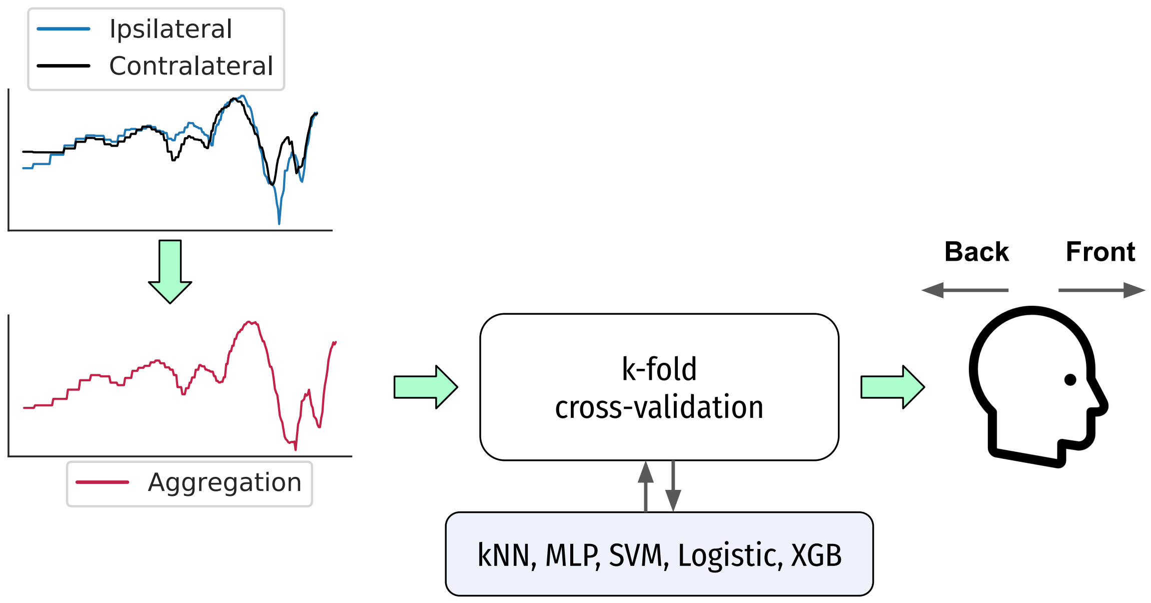


Figure 7.11: Linear Discriminant Analysis implementation for the Front and Back audio sources presented in the CIPIC dataset.
Source: Own elaboration.

7.2.5.1 *Preliminary test: Dimensionality reduction via Linear Discriminant Analysis*

In order to show a first attempt to classify the CIPIC source locations, we have implemented a Linear Discriminant Analysis approach to discover whether exists linear combinations of the features (frequency bands) that separate the data domain. The point of the experiment is also to generate a low-dimensional representation of the HRTF signals with this dimensionality reduction and illustrate it graphically. In Fig. 7.12 is depicted a two-dimensional representation of the LDA score per each audio instance. Although there is no linear separation between the projection of each class, we can see a boundary region with notorious differences in front-back localization.

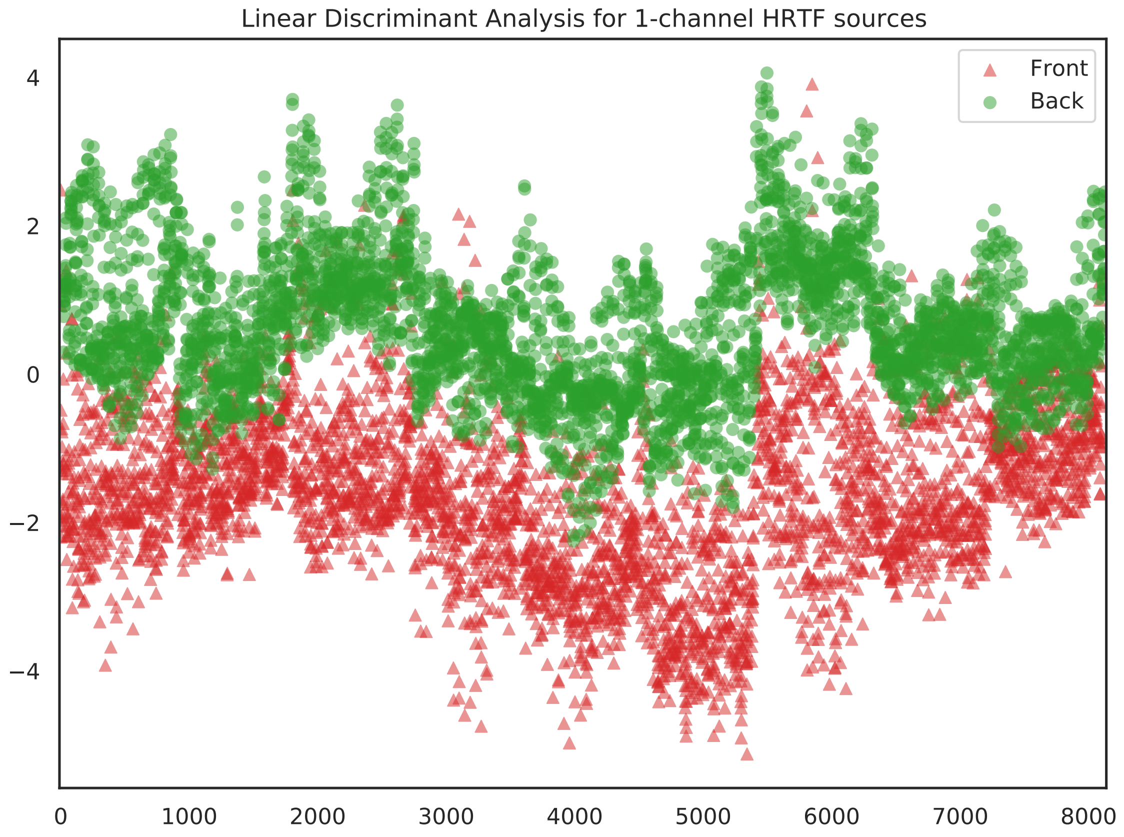


Figure 7.12: Linear Discriminant Analysis implementation for the Front and Back audio sources presented in the CIPIC dataset.
Source: Own elaboration.

We have additionally extracted the evaluation scores using the LDA algorithm as a binary classifier for the entire CIPIC dataset. The results are shown in Table 7.10.

Table 7.10: Test results obtained by Linear Discriminant Analysis in the predictive task of CIPIC for test data.

	ACC	AUC	MCC	FM
Linear Discriminant Analysis	88.74	88.56	78.20	80.29

From Table 7.10 and Fig. 7.12 we can extract interesting conclusions. Apart from having great discrimination between front-back spatial sources, with a 0.8856 AUC score, we can also spot clear differences regarding LDA components. This has been reflected in the results, especially with the MCC and FM coefficients. Regarding the positive and negative rates, it has been shown that LDA has been able to distinguish between classes since the accuracy measure is quite similar to AUC.

7.2.5.2 *Main problem: Selection of the best classification model*

In order to discriminate the front and back sound sources of the HRTF spatial sound excerpts, we have selected five machine learning models to carry out the binary classification task. As we mentioned, the models had an input vector defined in $\mathbb{R}^{257}$ with the aggregated Mel-scale frequency bands of the ipsilateral and contralateral channels of each HRTF signal. The models selected have been the K-Nearest Neighbours (KNN, Cover and Hart, 1967), Multi-Layer Perceptron (MLP, Rosenblatt, 1963), Support Vector Machine (SVM, Vapnik, 1992), logistic regression (Logistic), and XGBoost (XGB). The choice of these models is not trivial since KNN, SVM, and MLP are usually compared in terms of performance, see for instance Tsai et al. (2009) or Dino and Abdulrazzaq (2019). Moreover, we also wanted to add an explainability phase, in which the feature importance from logistic regression and XGBoost can be easily extracted. Another point to remark on is the use of Logistic regression as a benchmark since it is one of the most straightforward models in data mining. Thus, we can compare the predictive power of the other models with respect to this baseline.

The models have been trained with the default parameters set by Sklearn. In this way, it has been implemented KNN with an automatic learning algorithm, MLP with log-loss function and Adam optimizer, SVM with RBF kernel, Logistic regression with L2 penalty and L-BFGS solver, and XGBoost with log-loss. As we described the 9-fold cross-validation has been performed by dividing the number of subjects, 45 in total, instead of audio samples. So we have generated a 9:36 test-train split with the same split ratio as utilized in § 6.3.

The results obtained after the 9-fold cross-validation stage are presented in Table 7.11, where they are shown in LR-fuzzy shape. It is interesting to highlight that the minimum AUC score reached has been 0.81 for the Logistic model, thus proving great performance in the binary classification conducted. In any case, the mean for the area under the curve metric has been superior to 0.9 for each of the other classifiers. It is easy to note that, in terms of machine learning evaluation, the SVM has outperformed the remaining algorithms with a minimum accuracy of 0.9493, this value being greater than the maximum for every other model but MLP. As we assumed, Logistic regression has been the model with the worst performance in the four evaluation metrics selected, however, it is worth mentioning that it is quite efficient in terms of usage and applicability of its low predictive cost and computer storage.

For further information about the models' performance, we have displayed the confusion matrices per each cross-validation in Fig. 7.13. We have highlighted the hit ratio in a green

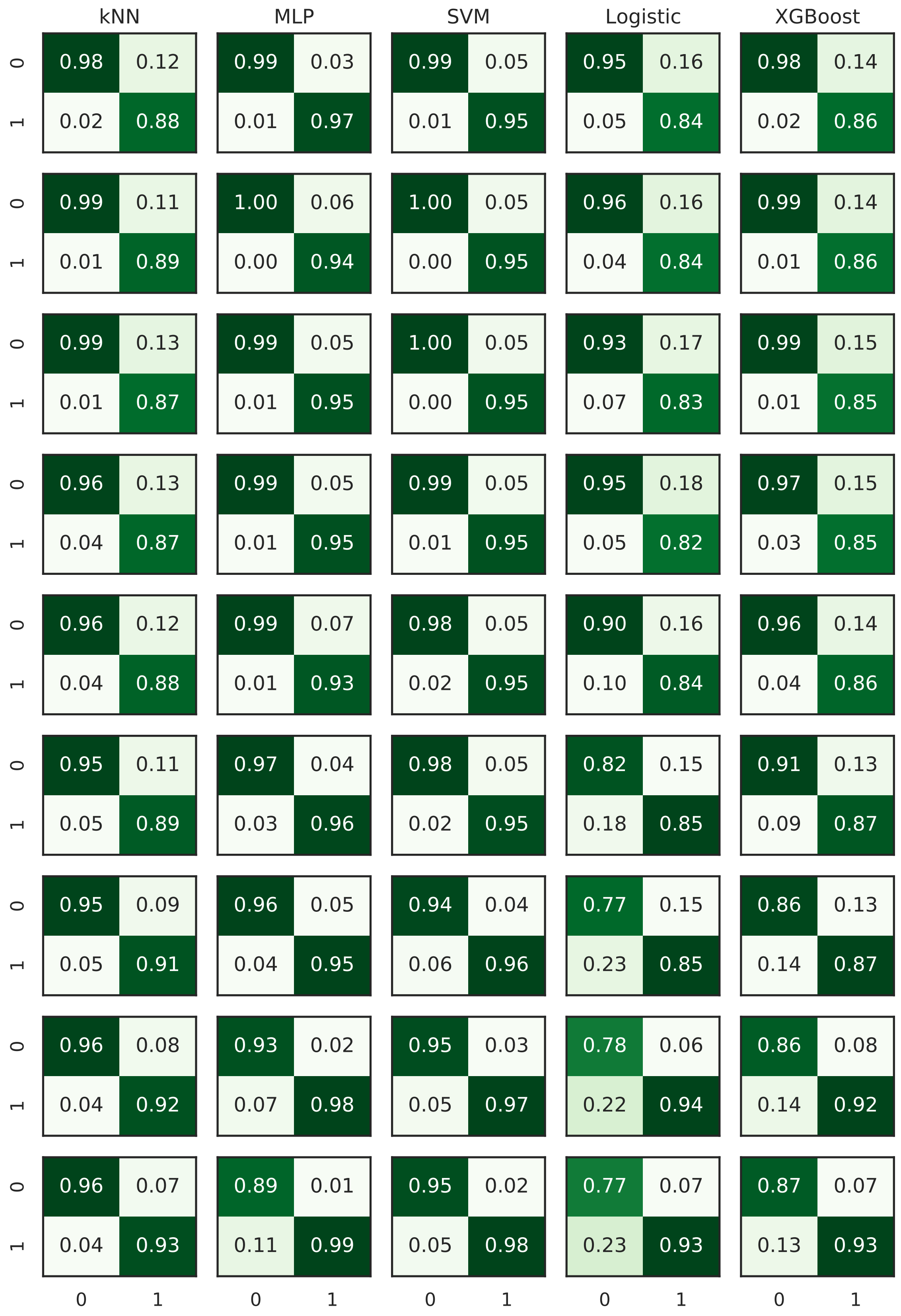


Figure 7.13: Confusion matrices as hit ratio for the different 9-fold cross-validations (rows) and models implemented (columns). The ratios are presented in a gray colormap.
Source: Own elaboration.

Table 7.11: Fuzzy decision matrix of the classification models for the CIPIC dataset regarding evaluation metrics (percentage), processing times (seconds) with five significant figures, and computational sizes (MB). The values per each criteria are presented according to Eq. 7.5 after the 9-fold cross-validation process.

	ACC				AUC				MCC				FM			
Model	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4
kNN	90.71	91.85	92.90	94.25	90.56	91.72	92.80	94.21	81.99	84.16	86.11	88.56	83.33	85.18	86.91	89.19
MLP	93.36	95.55	96.77	97.76	93.48	95.56	96.73	97.73	87.29	91.30	93.62	95.55	87.64	91.55	93.77	95.63
SVM	94.93	96.09	96.79	97.34	94.94	96.08	96.76	97.28	89.87	92.23	93.66	94.80	90.38	92.51	93.81	94.84
Logistic	80.88	84.30	86.84	88.78	81.00	84.35	86.74	88.58	62.25	69.44	74.54	78.50	69.20	73.92	77.53	80.42
XGB	86.49	89.03	90.48	91.36	86.48	88.97	90.35	91.17	72.96	78.46	81.69	83.76	76.64	80.64	82.99	84.49

	Training (s)				Predicting (s)				Saving (s)				Storage (MB)			
Model	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4
kNN	0.0076	0.0079	0.0088	0.0103	30.4348	33.5744	38.1458	43.6406	0.1443	0.1932	0.2559	0.2955	67.0974	67.0974	67.0974	67.0974
MLP	56.1889	85.2881	121.3578	144.5448	0.3209	0.4256	0.5131	0.6051	0.0146	0.0155	0.0196	0.0265	0.6295	0.6299	0.6307	0.6316
SVM	22.4708	23.5912	24.9391	25.5122	4.6086	4.9301	5.1437	5.2921	0.0180	0.0209	0.0280	0.0367	5.5942	5.8016	6.0794	6.3374
Logistic	30.1764	32.7952	35.7487	39.8029	0.0066	0.0071	0.0076	0.0080	0.0062	0.0072	0.0109	0.0171	0.0030	0.0030	0.0030	0.0030
XGB	44.7915	45.0619	45.4225	45.8712	0.0677	0.0700	0.0725	0.0744	0.0067	0.0076	0.0085	0.0096	0.0730	0.0732	0.0734	0.0735

scale in order to see the precision in each stage. In this graphic, we can also check that MLP and SVM have returned a great fit for the HRIR signals since they have reached maximum ratios of $(0.99, 1.0)$ and $(0.98, 1.0)$ for the front and back labels respectively. For KNN and XGB, their predictive power has been shown in the second trial, in which the true and negative ratios were $(0.89, 0.99)$ and $(0.86, 0.99)$ respectively.

From Table 7.11, we have conducted the decision-making stage as stated in § 7.2.3.3. Table 7.12 contains the six respective ranking systems for the FUW-MM methodology broken down per normalization function and p-norm.

Table 7.12: Score and ranking systems for the CIPIC classifiers by the FUW-MM technique disaggregated per normalization function and p-norm.

	ℓ^2 vector			**Min-Max**		
	$p = 0$	$p = 1$	$p = 2$	$p = 0$	$p = 1$	$p = 2$
KNN	0.9685 (5)	2.1956 (5)	2.6101 (5)	0.0000 (5)	2.9074 (5)	3.4982 (5)
MLP	1.0847 (2)	2.7971 (2)	3.9404 (4)	1.4114 (2)	4.4698 (1)	6.1985 (1)
SVM	1.1148 (1)	2.6294 (3)	4.0231 (1)	1.7773 (1)	4.4636 (2)	4.9211 (3)
Logistic	1.0655 (4)	2.8078 (1)	3.9926 (2)	0.9191 (4)	3.3100 (4)	4.5318 (4)
XGB	1.0840 (3)	2.5977 (4)	3.9854 (3)	1.3873 (3)	3.6370 (3)	5.0820 (2)

In this experiment, we have not obtained the best classification model unanimously. It is obvious that KNN can be considered the worst choice in terms of front-back discrimination of the HRTF signals. Although its predictive ratios are quite good, the computational task places it in the last position. For the best classification model, we can conclude that SVM is the most suitable classifier for the problem of the CIPIC's excerpts. Our decision is basically guided by two points. First, the support vector machine has been unique to reach the top position in the ranking 3 times. Second, SVM has clearly outperformed the other models according to the evaluation metrics selected. In any case, the performance of MLP is remarkable as shown in the positions achieved in Table 7.12 and its reduced differences with SVM. For the Logistic regression and XGBoost, the positions have varied within the last places of the ranking, except the first place achieved by the Logistic model for the ℓ^2 normalization and 2-mean.

7.2.6 *Conclusions*

We have presented an integrated end-to-end system for benchmarking Machine Learning models. In this manner, we have conducted an objective comparative study in which the decision-maker just has to set an appropriate number of experiments and reasonable lower and upper bounds for the criteria considered. The proposed system is in charge of performing the analysis and then returning a ranking that will lead to the final model selection by means of three stages. First, it is performed a k-fold cross-validation. Second, the k samples are transformed into LR-fuzzy numbers. Third, a novel MCDM methodology (FUW-MM) is applied to carry out the ranking, thus yielding the decision-making step.

The applicability of our system has been demonstrated in two distinct supervised tasks: classification and regression. For this purpose, we have selected two datasets associated with two complex real-life problems. In order to give a comprehensive comparison, four evaluation metrics and four computational attributes were considered. Furthermore, in both tasks five Machine Learning models were assessed. Hence, the decision phase meant a difficult issue for human benchmarking due to the conflict among criteria and their different magnitudes. Finally, six different ranking systems were given to compare and contrast the results obtained per each model.

7.2.7 *Python implementation*

The scripts utilized for this case study are presented in § C.2, where they are disaggregated per each experiment in § C.2.1 and § C.2.2 respectively. For the regression of the close price of AstraZeneca in the stock market, we have displayed the Prophet implementation (Script 18) and the *k*-fold cross-validation for time series over the XGBoost, LightGBM, CatBoost, Random Forest, and Gaussian Regressor Process (Script 17). For the front-back source localization of signal sources for the CIPIC dataset, we have shown the implementation of the supervised version of Linear Discriminant Analysis (Script 20) and the *k*-fold cross-validation for the K-Nearest Neighbours, Multi-Layer Perceptron, Support Vector Machine, Logistic Regression, and XGBoost (Script 19).

7.3 SUMMARY

In this chapter, two case studies have been approached with a combination of Multiple-Criteria Decision Making and Artificial Intelligence methodologies. In the first case (§ 7.1), we have studied the early detection of students' failure in the Universidad Industrial de Santander via a tree-based learning algorithm. Thus, we have analyzed the situation of the university and produced two interesting tools for the Colombian institution. An automatic ML model (XGBoost) that detects students at risk of academic failure and a dashboard that indicates the weaknesses of the criteria considered for the university. In the second case (§ 7.2), an integrated end-to-end system for automatic benchmarking of ML models has been presented with the aim of easing the model selection stage for decision-makers. Our system is composed of a k-fold cross-validation, a fuzzy transformation, and a new MCDM technique, named FUW-MM, that operates with LR-fuzzy numbers and without a priori weighting scheme. The robustness of our proposal has been shown in two Machine Learning supervised tasks: classification and regression.

8 CONCLUSIONS

During the development of this thesis, we deepened our research in the area of decision support systems. For this purpose, we based our work on the fields of Multiple-Criteria Decision Analysis and Artificial Intelligence to cover a large part of the problems related to decision making. Even though we have already highlighted some relevant points in the course of the book, this chapter remarks on the major contributions worth mentioning on the whole.

8.1 CONTRIBUTION 1: DESIGN AND IMPLEMENTATION OF UNWEIGHTED MULTIPLE-CRITERIA DECISION MAKING TECHNIQUES

The Chapter 3 of this thesis has associated the development of outranking unweighted techniques as one of the main goals. The study of the UW-MM, uwTOPSIS, and uwVIKOR means a breakthrough towards new ways of understanding the field of decision analysis. The main advantage is that an a priori weighting scheme is not required to implement the algorithms, instead, a flexible set of feasible solutions Ω allows us to obtain the best and worst solutions under a set of bounds $\{(l_j, u_j)\}_{j=1}^{M}$. As a result, we have a set of optimal pairs that may be combined to give a score that will yield the final ranking. Depending on the model and the approach some variants can be used to compute such a score. The selection of these models is determined by the scope of the problem performed since UW-MM offers fast computations, uwTOPSIS has high complexity attached to computing the ranking by ideal solutions, and uwVIKOR incorporates the notion of compromise solutions. Furthermore, the optimal solutions, i. e. the optimal weights (W^L, W^U), can be studied to analyze how the final scores have been calculated, giving us significant information about how decisive the criteria have been in evaluating the alternatives.

Intending to give a theoretical background to these methods, we have shown that the unweighted approaches generalize the classic ones through the Propositions 3.4.1, 3.4.2, and 3.4.3. In essence, these propositions proved that as long as the bounds satisfy $l_j = u_j = w_j^0$, per each criterion $j \in \{1, \dots, M\}$, the output of the unweighted versions of WMM, TOPSIS, and VIKOR for $\{(l_j, u_j)\}_{j=1}^{M}$ coincide with the classical one for the weighting scheme $\{w_j^0\}_{j=1}^{M}$.

The applicability and further details about similarities and differences among them have been discussed in Chapter 5. In particular, two case studies have shown the implementation of uwTOPSIS and uwVIKOR. Apart from being more flexible techniques, the solutions could be analyzed, thus giving interesting information about the attribute importance and alternative limitations.

8.2 CONTRIBUTION 2: INCORPORATION OF THE FUZZY UNWEIGHTED MULTIPLE-CRITERIA DECISION MAKING APPROACH

Another of the main goals of Chapter 3 is the incorporation of FUW-MCDM techniques. In § 3.5 is presented a routine for implementing this new methodology, where the addition of uncertainty and unweighted schemes makes this approach more complete than the classical one. The core idea is to handle the incertitude presented and help decision-makers with unique solutions because real problems cannot be considered as static systems. For applicability purposes, the Fuzzy Unweighted Mean Models method has been designed for LR-fuzzy numbers and its due computational implementation was presented in Algorithm 3.4. For theoretical purposes, Proposition 3.5.1 states that the FUW-MM is a generalization of the unweighted, fuzzy, and classic versions.

The applicability and robustness of the Fuzzy Unweighted Mean Models have been shown in the last case study § 7.2. In that experiment, the FUW-MM was part of an integrated end-to-end system for Machine Learning models benchmarking. In particular, the last stage of the process consisted of the implementation of the FUW-MM for selecting the best model according to its performance.

8.3 CONTRIBUTION 3: APPLICATION AND EVALUATION OF CUSTOMIZED ARTIFICIAL INTELLIGENCE MODELS IN COMPLEX ENVIRONMENTS

In Chapter 6 three case studies were approached from the Artificial Intelligence perspective. In every case, different methodologies were designed and implemented for achieving our target, where the customization level was considerably high. As we have shown in the case studies, all the final models have outperformed the state-of-the-art baselines, so they could be incorporated and deployed into their respective fields of work. Nevertheless, solving the problem was not the only objective of the experiments but also the evaluation of the optimal solution obtained. In general, the final solutions have been analyzed with the purpose of adding comprehensive information about the model input processing or representing other input for making a-posteriori comparisons.

8.4 CONTRIBUTION 4: COMBINATION OF THE FIELDS OF MULTIPLE-CRITERIA DECISION MAKING AND ARTIFICIAL INTELLIGENCE

This thesis is focused on the study of decision analysis. For this reason, one of our aims was to utilize the MCDM and AI fields separately and together. By applying these two subjects, we have shown that they could be combined to reinforce the objective of decision-makers. First, the scores obtained in outranking MCDM techniques offer significant information about the alternatives' performance. Then, the use of MCDM scores as a feature extractor can be very useful and beneficial in guiding the Machine Learning models. Second, we have shown that model benchmarking for Artificial Intelligence solutions can be addressed by MCDM. More specifically, FUW-MCDM can be used to aid decision markers and overcome the limitations of human-subjective decisions.

8.5 CONTRIBUTION 5: PUBLICATION OF GITHUB REPOSITORIES

This thesis is structured in theoretical and practical cases with the same level of importance. Because of this, we have put a lot of effort into programming and documenting every single detail concerning the experiments conducted. For this reason, we have published four GitHub repositories that have considerably helped the development of this work. For reproducibility and sharing purposes, all the packages are publicly available on my GitHub personal page: Aaron-AALG.

8.6 OUTLOOK AND FUTURE WORK

Given that the field of decision theory is a very prolific area of research, the continuation of this work is connected with innovative research opportunities and new market niches due to their broad applicability. For this reason, it would not be appropriate to finish this thesis without mentioning potential avenues for follow-up research. In essence, we believe that the following list of future work is worth proposing.

1. Study the properties of the unweighted versions of other Multiple-Criteria Decision Making methods. Throughout the thesis, we have focused our attention on some MCDA outranking techniques. However, this approach could be extended to other outranking techniques and/or other methods within the MCDM perspective.

2. Apply the Fuzzy Unweighted Mean Models technique in a real decision-making scenario. The purpose is to create a comprehensive example in which the importance of the p-mean and the normalization function will be studied together with the use of LR-fuzzy numbers and fuzzy bounds.

3. Design and program the integrated end-to-end system for automatic model benchmarking defined in the case study § 7.2 as an open-source solution. The core idea is to publish a library that reproduces the entire work routine for a given set of Machine Learning models to benchmark and the number of experiments to carry out during the k-fold cross-validation. Our system will automatically process the experiment with complete autonomy and will return the output ranking and the optimal solutions associated.

4. Study more possible combinations of MCDM and AI for decision-aiding. Since we have shown that ensemble approaches can be very helpful in decision-making, we must emphasize these automated computer-assisted solutions that really support decision-makers.

BIBLIOGRAPHY

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., . . . Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems [Software available from tensorflow.org]. https://www.tensorflow.org/

Abeßer, J. (2020). A review of deep learning based methods for acoustic scene classification. *Applied Sciences*, *10*(6). https://doi.org/10.3390/app10062020

Adams, A. J., & Hancock, T. (2000). Work experience as a predictor of mba performance. *College Student Journal*, *34*(2), 211–217.

Alamoodi, A. H., Zaidan, B. B., Zaidan, A. A., Albahri, O. S., Mohammed, K. I., Malik, R. Q., Almahdi, E. M., Chyad, M. A., Tareq, Z., Albahri, A. S., Hameed, H., & Alaa, M. (2021). Sentiment analysis and its applications in fighting COVID-19 and infectious diseases: A systematic review. *Expert Systems with Applications*, *167*. https://doi.org/10.1016/j.eswa.2020.114155

Albawi, S., Mohammed, T. A., & Al-Zawi, S. Understanding of a convolutional neural network. In: In *2017 international conference on engineering and technology (icet)*. 2017, 1–6. https://doi.org/10.1109/ICEngTechnol.2017.8308186.

Alberg, D., & Last, M. (2018). Short-term load forecasting in smart meters with sliding window-based arima algorithms. *Vietnam Journal of Computer Science*, *5*(3), 241–249. https://doi.org/10.1007/s40595-018-0119-7

Aldakheel, F., Satari, R., & Wriggers, P. (2021). Feed-forward neural networks for failure mechanics problems. *Applied Sciences*, *11*(14). https://doi.org/10.3390/app11146483

Alemi-Ardakani, M., Milani, A. S., Yannacopoulos, S., & Shokouhi, G. (2016). On the effect of subjective, objective and combinative weighting in multiple criteria decision making: A case study on impact optimization of composites. *Expert Systems with Applications*, *46*, 426–438. https://doi.org/10.1016/j.eswa.2015.11.003

Algazi, V., Duda, R., Thompson, D., & Avendano, C. The cipic hrtf database. In: In *Proceedings of the 2001 ieee workshop on the applications of signal processing to audio and acoustics (cat. no.01th8575)*. 2001, 99–102. https://doi.org/10.1109/ASPAA.2001.969552.

Ali, M. (2019). *Pycaret: An open source, low-code machine learning library in python* [PyCaret version 2.3.10 documentation: https://pycaret.readthedocs.io/en/latest/]. https://www.pycaret.org

Ali, S., Sahoo, B., Zelikovskiy, A., Chen, P.-Y., & Patterson, M. (2022). Benchmarking machine learning robustness in covid-19 genome sequence classification. https://doi.org/10.48550/ARXIV.2207.08898

Althelaya, K. A., El-Alfy, E.-S. M., & Mohammed, S. Evaluation of bidirectional lstm for short-and long-term stock market prediction. In: In *2018 9th international conference on information and communication systems (icics)*. 2018, 151–156. https://doi.org/10.1109/IACS.2018.8355458.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175–185. https://doi.org/10.1080/00031305.1992.10475879

Aouni, B., & Kettani, O. (2001). Goal programming model: A glorious history and a promising future [Multiobjective Programming and Goal Programming]. *European Journal of Operational Research*, *133*(2), 225–231. https://doi.org/10.1016/S0377-2217(00)00294-0

Archibald, B. C., & Koehler, A. B. (2003). Normalization of seasonal factors in winters' methods. *International Journal of Forecasting*, *19*, 143–148. www.elsevier.com/locate/ijforecast

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., & Benjamins, R. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion*, *58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Awodele, O, Kasali, F., Akinsola, J. E. T., & Kuyoro, S. Performance evaluation of supervised machine learning algorithms using multi-criteria decision making techniques. In: 2020. https://publication.babcock.edu.ng/asset/docs/publications/COSC/9517/6574.pdf

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, *10*(7), 1–46. https://doi.org/10.1371/journal.pone.0130140

Bahrainian, S. A., Liwicki, M., & Dengel, A. (2014). Fuzzy subjective sentiment phrases: A context sensitive and self-maintaining sentiment lexicon. *Proceedings - 2014 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT 2014*, *1*, 361–368. https://doi.org/10.1109/WI-IAT.2014.57

Balasundaram, S., & Kapil. (2010). On lagrangian support vector regression. *Expert Systems with Applications*, *37*(12), 8784–8792. https://doi.org/10.1016/j.eswa.2010.06.028

Banerjee, A., Dunson, D. B., & Tokdar, S. T. (2012). Efficient Gaussian process regression for large datasets. *Biometrika*, *100*(1), 75–89. https://doi.org/10.1093/biomet/ass068

Bashir, D., Montañez, G. D., Sehra, S., Segura, P. S., & Lauw, J. An information-theoretic perspective on overfitting and underfitting (M. Gallagher, N. Moustafa & E. Lakshika, Eds.). In: *Ai 2020: Advances in artificial intelligence* (M. Gallagher, N. Moustafa & E. Lakshika, Eds.). Ed. by Gallagher, M., Moustafa, N., & Lakshika, E. Cham: Springer International Publishing, 2020, 347–358.

Basiri, M. E., Nemati, S., Abdar, M., Cambria, E., & Acharya, U. R. (2021). Abcdm: An attention-based bidirectional cnn-rnn deep model for sentiment analysis. *Future Generation Computer Systems*, *115*, 279–294. https://doi.org/10.1016/j.future.2020.08.005

Bebis, G., & Georgiopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials*, *13*(4), 27–31. https://doi.org/10.1109/45.329294

Behzadian, M., Kazemzadeh, R., Albadvi, A., & Aghdasi, M. (2010). Promethee: A comprehensive literature review on methodologies and applications. *European Journal of Operational Research*, *200*(1), 198–215. https://doi.org/10.1016/j.ejor.2009.01.021

Belavagi, M. C., & Muniyal, B. (2016). Performance evaluation of supervised machine learning algorithms for intrusion detection [Twelfth International Conference on Communication Networks, ICCN 2016, August 19– 21, 2016, Bangalore, India Twelfth International Conference on Data Mining and Warehousing, ICDMW 2016, August 19-21, 2016, Bangalore, India Twelfth International Conference on Image and Signal Processing, ICISP 2016, August 19-21, 2016, Bangalore, India]. *Procedia Computer Science*, *89*, 117–123. https://doi.org/10.1016/j.procs.2016.06.016

Bellman, R. E., & Zadeh, L. A. (1970). Decision-making in a fuzzy environment. *Management Science*, *17*(4), B141–B164. Retrieved October 13, 2022, from http://www.jstor.org/stable/2629367

Benayoun, R, Roy, B, & Sussman, N. (1966). Manual de reference du programme ELECTRE. *Note de synthese et Formation*, *25*, 79.

Benitez, R., & Liern, V. (2020). Unweighted TOPSIS model.

Benítez, R., & Liern, V. (2021). Unweighted TOPSIS: a new multi-criteria tool for sustainability analysis. *International Journal of Sustainable Development and World Ecology*, *28*(1), 36–48. https://doi.org/10.1080/13504509.2020.1778583

Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, *54*(3), 1937–1967. https://doi.org/10.1007/s10462-020-09896-5

Berri, D. J. (1999). Who is 'most valuable'? measuring the player's production of wins in the National Basketball Association. *Managerial and Decision Economics*, *20*, 411–427. https://doi.org/10.1002/1099-1468(199912)20:8<411::AID-MDE957>3.0.CO;2-G

Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms*. Springer New York, NY. https://doi.org/10.1007/978-1-4757-0450-1

Bezdek, J. C., Keller, J., Krisnapuram, R., & Pal, N. (1999). *Fuzzy models and algorithms for pattern recognition and image processing* (Vol. 4). Springer Science & Business Media. https://doi.org/10.1007/b106267

Bhatia, N., & Vandana. (2010). Survey of nearest neighbor techniques. https://doi.org/10.48550/ARXIV.1007.0085

Bhutto, E. S., Siddiqui, I. F., Arain, Q. A., & Anwar, M. Predicting students' academic performance through supervised machine learning. In: In *2020 international conference on information science and communication technology (icisct)*. 2020, 1–6. https://doi.org/10.1109/ICISCT49550.2020.9080033.

Biganzoli, E., Boracchi, P., Mariani, L., & Marubini, E. (1998). Feed forward neural networks for the analysis of censored survival data: A partial logistic regression approach. *Statistics in Medicine*, *17*(10), 1169–1186. https://doi.org/10.1002/(SICI)1097-0258(19980530)17:10<1169::AID-SIM796>3.0.CO;2-D

Blasco-Blasco, O., Liern-García, M., López-García, A., & Parada-Rico, S. E. (2021). An academic performance indicator using flexible multi-criteria methods. *Mathematics*, *9*(19), 2396. https://doi.org/10.3390/math9192396

Blasco-Blasco, O., Parada Rico, S. E., Liern-García, M., & López-García, A. Characterization of university students through indicators of adequacy and excellence. analysis from gender and socioeconomic status perspective. In: In *Iceri2020 proceedings*. 13th annual International Conference of Education, Research and Innovation. Online Conference: IATED, 2020, 8030–8037. ISBN: 978-84-09-24232-0. https://doi.org/10.21125/iceri.2020.1780.

Blashfield, R. K., & Aldenderfer, M. S. (1978). The literature on cluster analysis [PMID: 26821722]. *Multivariate Behavioral Research*, *13*(3), 271–295. https://doi.org/10.1207/s15327906mbr1303_2

Blauert, J. (1996). *Spatial Hearing: The Psychophysics of Human Sound Localization*. The MIT Press. https://doi.org/10.7551/mitpress/6391.001.0001

Bojadziev, G., & Bojadziev, M. (1995). *Fuzzy sets, fuzzy logic, applications*. World Scientific. https://books.google.es/books?id=At2xJDcu-3sC

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327. https://doi.org/10.1016/0304-4076(86)90063-1

Boser, B. E., Guyon, I. M., & Vapnik, V. N. A training algorithm for optimal margin classifiers. In: In *Proceedings of the fifth annual workshop on computational learning theory*. COLT '92. New York, NY, USA: Association for Computing Machinery, 1992, 144–152. ISBN: 089791497X. https://doi.org/10.1145/130385.130401.

Bottou, L., & Lin, C.-J. Support vector machine solvers (L. Bottou, O. Chapelle, D. DeCoste & J. Weston, Eds.). In: *Large scale kernel machines* (L. Bottou, O. Chapelle, D. DeCoste & J. Weston, Eds.). Ed. by Bottou, L., Chapelle, O., DeCoste, D., & Weston, J. Cambridge, MA.: MIT Press, 2007, pp. 301–320. http://leon.bottou.org/papers/bottou-lin-2006

Bouslah, K., Liern, V., Ouenniche, J., & Pérez-Gladish, B. (2022). Ranking firms based on their financial and diversity performance using multiple-stage unweighted TOPSIS. *International Transactions in Operational Research*. https://doi.org/10.1111/itor.13143

Bowman, J. S. (2000). *Columbia chronologies of asian history and culture*. Columbia University Press. https://cup.columbia.edu/book/columbia-chronologies-of-asian-history-and-culture/9780231110044

Box, G., & Jenkins, G. (1970). *Time series analysis: Forecasting and control*. Holden-Day. https://books.google.es/books?id=5BVfnXaq03oC

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2

Brans, J.-P. (1982). *L'ingénierie de la décision: L'élaboration d'instruments d'aide a la décision*. Université Laval, Faculté des sciences de l'administration.

Brans, J., & Mareschal, B. (2005). Chapter 5: PROMETHEE methods. *Multiple Criteria Decision Analysis: State of the Art Surveys*, 164–189.

Breiman, L., Friedman, J., Stone, C., & Olshen, R. (1984). *Classification and regression trees*. Taylor & Francis. https://books.google.es/books?id=JwQx-WOmSyQC

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, *78*(1), 1–3. https://doi.org/10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2

Brinkhuis, J., & Tikhomirov, V. (2005). *Optimization: Insights and applications: Insights and applications*. Princeton University Press. Retrieved May 25, 2022, from http://www.jstor.org/stable/j.ctt7s71d

Bromley, A. (1998). Charles Babbage's analytical engine, 1838. *IEEE Annals of the History of Computing*, *20*(4), 29–45. https://doi.org/10.1109/85.728228

Browne, M. W. (2000). Cross-validation methods. *Journal of Mathematical Psychology*, *44*(1), 108–132. https://doi.org/10.1006/jmps.1999.1279

Buhmann, M. D. (2000). Radial basis functions. *Acta Numerica*, *9*, 1–38. https://doi.org/10.1017/S0962492900000015

Bullen, P. S. (2003). *Handbook of means and their inequalities*. Springer Netherlands. https://doi.org/10.1007/978-94-017-0399-4

Burns, J. A., & Whitesides, G. M. (1993). Feed-forward neural networks in chemistry: Mathematical systems for classification and pattern recognition. *Chemical Reviews*, *93*(8), 2583–2601. https://doi.org/10.1021/cr00024a001

Butler, R. A., & Belendiuk, K. (1977). Spectral cues utilized in the localization of sound in the median sagittal plane. *The Journal of the Acoustical Society of America*, *61*(5), 1264–1269. https://doi.org/10.1121/1.381427

Byvatov, E., & Schneider, G. (2003). Support vector machine applications in bioinformatics. *Applied bioinformatics*, *2*(2), 67—77. http://europepmc.org/abstract/MED/15130823

Cables, E., Lamata, M., & Verdegay, J. (2016). Rim-reference ideal method in multicriteria decision making. *Information Sciences*, *337-338*, 1–10. https://doi.org/10.1016/j.ins.2015.12.011

Cam, L. L. (1986). The central limit theorem around 1935. *Statistical Science*, *1*(1), 78–91. Retrieved July 9, 2022, from http://www.jstor.org/stable/2245503

Cao, Z., Price, J., & Stone, D. F. (2011). Performance under pressure in the NBA. *Journal of Sports Economics*, *12*(3), 231–252. https://doi.org/10.1177/1527002511404785

Carrizosa, E., & Romero Morales, D. (2013). Supervised classification and mathematical optimization. *Computers & Operations Research*, *40*(1), 150–165. https://doi.org/10.1016/j.cor.2012.05.015

Cayley, A. (1857). On the theory of the analytical forms called trees. *Mathematical papers*, *3*, 242–246. https://doi.org/10.1017/CBO9780511703690.046

Cayton, L. (2005). Algorithms for manifold learning. *University of California at San Diego Tech. Rep*, *12*(1-17), 1. https://www.cs.columbia.edu/~verma/classes/ml/ref/lec8_cayton_manifolds.pdf

Chakraborty, K., Bhatia, S., Bhattacharyya, S., Platos, J., Bag, R., & Hassanien, A. E. (2020). Sentiment analysis of covid-19 tweets by deep learning classifiers—a study to show how popularity is affecting accuracy in social media. *Applied Soft Computing Journal*, *97*. https://doi.org/10.1016/j.asoc.2020.106754

Chambers, J. M. (1998). *Programming with data: A guide to the S language*. Springer Science & Business Media.

Chambers, J. M. (2008). *Software for data analysis: Programming with R* (Vol. 2). Springer. https://doi.org/10.1007/978-0-387-75936-4

Chang, J., Lee, J., Ha, A., Han, Y. S., Bak, E., Choi, S., Yun, J. M., Kang, U., Shin, I. H., Shin, J. Y., Ko, T., Bae, Y. S., Oh, B.-L., Park, K. H., & Park, S. M. (2021a). Explaining the rationale of deep learning glaucoma decisions with adversarial examples. *Ophthalmology*, *128*(1), 78–88. https://doi.org/10.1016/j.ophtha.2020.06.036

Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., & Borges, B. (2021b). *Shiny: Web application framework for R* [R package version 1.7.1]. https://CRAN.R-project.org/package=shiny

Chang, Y. C., Chang, K. H., & Wu, G. J. (2018). Application of eXtreme gradient boosting trees in the construction of credit risk assessment models for financial institutions. *Applied Soft Computing Journal*, *73*, 914–920. https://doi.org/10.1016/j.asoc.2018.09.029

Charnes, A., Cooper, W. W., & Ijiri, Y. (1963). Breakeven budgeting and programming to goals. *Journal of Accounting Research*, *1*(1), 16–43. Retrieved June 17, 2022, from http://www.jstor.org/stable/2489841

Charnes, A., Cooper, W. W., DeVoe, J., Learner, D. B., & Reinecke, W. (1968). A goal programming model for media planning. *Management Science*, *14*(8), B–423. https://doi.org/10.1287/mnsc.14.8.B423

Chattopadhay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In: In *2018 ieee winter conference on applications of computer vision (wacv)*. 2018, 839–847. https://doi.org/10.1109/WACV.2018.00097.

Chen, C.-T. (2000). Extensions of the topsis for group decision-making under fuzzy environment. *Fuzzy Sets and Systems*, *114*(1), 1–9. https://doi.org/10.1016/S0165-0114(97)00377-1

Chen, N., Lu, W., Yang, J., & Li, G. (2004). *Support vector machine in chemistry*. WORLD SCIENTIFIC. https://doi.org/10.1142/5589

Chen, S., & Donoho, D. (1994). Basis pursuit. *Conference Record - Asilomar Conference on Signals, Systems and Computers*, *1*, 41–44. https://doi.org/10.1109/ACSSC.1994.471413

Chen, S.-J., & Hwang, C.-L. (1992). *Fuzzy multiple attribute decision making - methods and applications*. Springer Berlin. https://doi.org/10.1007/978-3-642-46768-4

Chen, S.-M., & Lee, L.-W. (2010). Fuzzy multiple attributes group decision-making based on the interval type-2 topsis method. *Expert Systems with Applications*, *37*(4), 2790–2798. https://doi.org/10.1016/j.eswa.2009.09.012

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 785–794. https://doi.org/10.1145/2939672.2939785

Chen, T.-Y., & Tsao, C.-Y. (2008). The interval-valued fuzzy topsis method and experimental analysis [Theme: Fuzzy Interval Analysis]. *Fuzzy Sets and Systems*, *159*(11), 1410–1428. https://doi.org/10.1016/j.fss.2007.11.004

Chen, Z., Yeo, C. K., Lee, B. S., & Lau, C. T. Autoencoder-based network anomaly detection. In: In *2018 wireless telecommunications symposium (wts)*. IEEE. 2018, 1–5. https://doi.org/10.1109/WTS.2018.8363930.

Cheng, B., & Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. *Statistical Science*, *9*(1), 2–30. Retrieved July 27, 2022, from http://www.jstor.org/stable/2246275

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, *21*(1), 1–13. https://doi.org/10.1186/s12864-019-6413-7

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The matthews correlation coefficient (mcc) is more informative than cohen's kappa and brier score in binary classification assessment. *IEEE Access*, *9*, 78368–78381. https://doi.org/10.1109/ACCESS.2021.3084050

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Chomboon, K., Chujai, P., Teerarassammee, P., Kerdprasop, K., & Kerdprasop, N. An empirical study of distance metrics for k-nearest neighbor algorithm. In: 2015, 280–285. https://doi.org/10.12792/iciae2015.051.

Christensen, E. (2009). Methodology of diagnostic tests in hepatology. *Annals of Hepatology*, *8*(3), 177–183. https://doi.org/10.1016/S1665-2681(19)31763-6

Clark, G. Chapter 5 - the industrial revolution (P. Aghion & S. N. Durlauf, Eds.). In: *Handbook of economic growth* (P. Aghion & S. N. Durlauf, Eds.). Ed. by Aghion, P., & Durlauf, S. N. Vol. 2. Handbook of Economic Growth. Elsevier, 2014, pp. 217–262. https://doi.org/10.1016/B978-0-444-53538-2.00005-8.

Cogger, K. O., & Yu, P. L. (1985). Eigenweight vectors and least-distance approximation for revealed preference in pairwise weight ratios. *Journal of Optimization Theory and Applications*, *46*, 483–491.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, *20*(1), 37–46. https://doi.org/10.1177/001316446002000104

Cooper, S., & van Leeuwen, J. (2013). *Alan turing: His work and impact*. Elsevier Science. https://doi.org/10.1016/C2010-0-66380-2

Copeland, B. J. (1997). The Church-Turing thesis. https://plato.stanford.edu/archives/spr2019/entries/church-turing/

Corchado, J., Fyfe, C., & Lees, B. Unsupervised learning for financial forecasting. In: In *Proceedings of the ieee/iafe/informs 1998 conference on computational intelligence for financial engineering (cifer) (cat. no.98th8367)*. 1998, 259–263. https://doi.org/10.1109/CIFER.1998.690316.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, *20*(3), 273–297. https://doi.org/10.1023/A:1022627411411

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27. https://doi.org/10.1109/TIT.1967.1053964

Cutler, A., Cutler, D. R., & Stevens, J. R. (2012). Random forests. In C. Zhang & Y. Ma (Eds.), *Ensemble machine learning: Methods and applications* (pp. 157–175). Springer US. https://doi.org/10.1007/978-1-4419-9326-7_5

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303–314. https://doi.org/10.1007/BF02551274

Dahl, D. B., Scott, D., Roosen, C., Magnusson, A., & Swinton, J. (2019). *Xtable: Export tables to LaTeX or HTML* [R package version 1.8-4]. https://CRAN.R-project.org/package=xtable

Dang, L. M., Hassan, S. I., Im, S., & Moon, H. (2019). Face image manipulation detection based on a convolutional neural network. *Expert Systems with Applications*, *129*, 156–168. https://doi.org/10.1016/j.eswa.2019.04.005

Dantzig, G. B. Origins of the simplex method. In: In *A history of scientific computing*. 1990, pp. 141–151. Retrieved May 25, 2022, from https://dl.acm.org/doi/pdf/10.1145/87252.88081

Daraghmeh, M., Agarwal, A., Manzano, R., & Zaman, M. Time series forecasting using facebook prophet for cloud resource management. In: In *2021 ieee international conference on communications workshops (icc workshops)*. 2021, 1–6. https://doi.org/10.1109/ICCWorkshops50388.2021.9473607.

Das, A., & Rad, P. (2020). Opportunities and challenges in explainable artificial intelligence (XAI): A survey. *CoRR*, *abs/2006.11371*. https://arxiv.org/abs/2006.11371

Dawes, R. M., & Corrigan, B. (1974). Linear models in decision making. *Psychological Bulletin*, *81*(2), 95–106. https://doi.org/10.1037/h0037613

de Vries, J. (1994). The industrial revolution and the industrious revolution. *The Journal of Economic History*, *54*(2), 249–270. https://doi.org/10.1017/S0022050700014467

de las Heras-Pedrosa, C., Sánchez-Núñez, P., & Peláez, J. I. (2020). Sentiment analysis and emotion understanding during the covid-19 pandemic in spain and its impact on digital ecosystems. *International Journal of Environmental Research and Public Health*, *17*, 1–22. https://doi.org/10.3390/ijerph17155542

De Prisco, R., Esposito, A., Lettieri, N., Malandrino, D., Pirozzi, D., Zaccagnino, G., & Zaccagnino, R. Music plagiarism at a glance: Metrics of similarity and visualizations. In: In *2017 21st international conference information visualisation (iv)*. 2017, 410–415. https://doi.org/10.1109/iV.2017.49.

Deconinck, E., Hancock, T., Coomans, D., Massart, D., & Heyden, Y. V. (2005). Classification of drugs in absorption classes using the classification and regression trees (CART) methodology. *Journal of Pharmaceutical and Biomedical Analysis*, *39*(1), 91–103. https://doi.org/10.1016/j.jpba.2005.03.008

Dehghan-Manshadi, B., Mahmudi, H., Abedian, A., & Mahmudi, R. (2007). A novel method for materials selection in mechanical design: Combination of non-linear normalization and a modified digital logic method. *Materials and Design*, *28*, 8–15. https://doi.org/10.1016/j.matdes.2005.06.023

Delahoz-Dominguez, E., Zuluaga, R., & Fontalvo-Herrera, T. (2020). Dataset of academic performance evolution for engineering students. *Data in Brief*, *30*, 105537. https://doi.org/10.1016/j.dib.2020.105537

Delashmit, W. H., & Manry, M. T. Recent developments in multilayer perceptron neural networks. In: In *Proceedings of the seventh annual memphis area engineering and science conference, maesc*. 2005. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.318.4243

Deng, L., & Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing*, *7*(3–4), 197–387. https://doi.org/10.1561/2000000039

Devi, K. (2011). Extension of VIKOR method in intuitionistic fuzzy environment for robot selection. *Expert Systems with Applications*, *38*(11), 14163–14168. https://doi.org/10.1016/j.eswa.2011.04.227

Devroye, L. (1978). The uniform convergence of nearest neighbor regression function estimators and their application in optimization. *IEEE Transactions on Information Theory*, *24*(2), 142–151. https://doi.org/10.1109/TIT.1978.1055865

Dey, R., & Salemt, F. M. Gate-variants of Gated Recurrent Unit (GRU) neural networks. In: In *Midwest symposium on circuits and systems*. *2017-August*. Institute of Electrical; Electronics Engineers Inc., 2017, 1597–1600. ISBN: 9781509063895. https://doi.org/10.1109/MWSCAS.2017.8053243. arXiv: 1701.05923.

Diakoulaki, D., Mavrotas, G., & Papayannakis, L. (1995). Determining objective weights in multiple criteria problems: The critic method. *Computers & Operations Research*, *22*(7), 763–770. https://doi.org/https://doi.org/10.1016/0305-0548(94)00059-H

Diday, E., & Simon, J. C. (1976). Clustering analysis. In K. S. Fu (Ed.), *Digital pattern recognition* (pp. 47–94). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-96303-2_3

Dino, H. I., & Abdulrazzaq, M. B. Facial expression classification based on svm, knn and mlp classifiers. In: In *2019 international conference on advanced science and engineering (icoase)*. 2019, 70–75. https://doi.org/10.1109/ICOASE.2019.8723728.

Doumpos, M., & Grigoroudis, E. (2013). *Intelligent decision support systems*. John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118522516

Doumpos, M., Grigoroudis, E., Matsatsinis, N. F., & Zopounidis, C. (2022). Preference disaggregation analysis: An overview of methodological advances and applications. In S. Greco, V. Mousseau, J. Stefanowski & C. Zopounidis (Eds.), *Intelligent decision support systems : Combining operations research and artificial intelligence - essays in honor of roman słowiński* (pp. 73–100). Springer International Publishing. https://doi.org/10.1007/978-3-030-96318-7_5

Došilović, F. K., Brčić, M., & Hlupić, N. Explainable artificial intelligence: A survey. In: In *2018 41st international convention on information and communication technology, electronics and microelectronics (mipro)*. 2018, 0210–0215. https://doi.org/10.23919/MIPRO.2018.8400040.

Dubois, D., Kerre, E., Mesiar, R., & Prade, H. Fuzzy interval analysis. In: In *Fundamentals of fuzzy sets*. Springer, 2000, pp. 483–581. https://doi.org/10.1007/978-1-4615-4429-6_11.

Dubois, D., & Prade, H. (1978). Operations on fuzzy numbers. *International Journal of Systems Science*, *9*(6), 613–626. https://doi.org/10.1080/00207727808941724

Dunn, J. C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, *3*(3), 32–57. https://doi.org/10.1080/01969727308546046

Duran, B., & Odell, P. (2013). *Cluster analysis: A survey*. Springer Berlin Heidelberg. https://books.google.es/books?id=OHnnCAAAQBAJ

Edwards, W. (1977). How to use multiattribute utility measurement for social decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics*, *7*, 326–340. https://doi.org/10.1109/TSMC.1977.4309720

Edwards, W., & Barron, F. (1994). Smarts and smarter: Improved simple methods for multiattribute utility measurement. *Organizational Behavior and Human Decision Processes*, *60*(3), 306–325. https://doi.org/https://doi.org/10.1006/obhd.1994.1087

Eisner, B., Rocktäschel, T., Augenstein, I., Bošnjak, M., & Riedel, S. (2016). Emoji2vec: Learning emoji representations from their description.

Elman, J. L. (1990). *Finding Structure in Time* (tech. rep.). https://doi.org/10.1207/s15516709cog1402_1

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, *50*, 987–1007. https://doi.org/10.2307/1912773

Esposito, F., Malerba, D., Semeraro, G., & Kay, J. (1997). A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(5), 476–491. https://doi.org/10.1109/34.589207

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: In *Proceedings of the second international conference on knowledge discovery and data mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, 226–231.

European Commission. Joint Research Centre. & Organisation for Economic Co-operation and Development. (2008). *Handbook on constructing composite indicators : methodology and user guide.* OECD. https://doi.org/10.1787/533411815016

Fan, C., Xiao, F., & Zhao, Y. (2017). A short-term building cooling load prediction method using deep learning algorithms. *Applied Energy*, *195*, 222–233. https://doi.org/10.1016/j.apenergy.2017.03.064

Fang, W.-X., Lan, P.-C., Lin, W.-R., Chang, H.-C., Chang, H.-Y., & Wang, Y.-H. Combine facebook prophet and lstm with bpnn forecasting financial markets: The morgan taiwan index. In: In *2019 international symposium on intelligent signal processing and communication systems (ispacs)*. 2019, 1–2. https://doi.org/10.1109/ISPACS48206.2019.8986377.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Fearnhead, P., & Taylor, B. M. (2011). On estimating the ability of NBA players. *Journal of Quantitative Analysis in Sports*, *7*(3). https://doi.org/doi:10.2202/1559-0410.1298

Federgruen, A., & Simchi-Levi, D. Chapter 4 analysis of vehicle routing and inventory-routing problems. In: In *Network routing*. Vol. 8. Handbooks in Operations Research and Management Science. Elsevier, 1995, pp. 297–373. https://doi.org/10.1016/S0927-0507(05)80108-2.

Fernández-Gavilanes, M., Juncal-Martínez, J., García-Méndez, S., Costa-Montenegro, E., & González-Castaño, F. J. (2018). Creating emoji lexica from unsupervised sentiment analysis of their descriptions. *Expert Systems with Applications*, *103*, 74–91. https://doi.org/10.1016/j.eswa.2018.02.043

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In F. Hutter, L. Kotthoff & J. Vanschoren (Eds.), *Automated machine learning: Methods, systems, challenges* (pp. 3–33). Springer International Publishing. https://doi.org/10.1007/978-3-030-05318-5_1

Fienberg, S. E. (1992). A brief history of statistics in three and one-half chapters: A review essay. *Statistical Science*, *7*(2), 208–225. Retrieved July 3, 2022, from http://www.jstor.org/stable/2246307

Figueira, J., & Roy, B. (2002). Determining the weights of criteria in the electre type methods with a revised simos' procedure [EURO XVI: O.R. for Innovation and Quality of Life]. *European Journal of Operational Research*, *139*(2), 317–326. https://doi.org/https://doi.org/10.1016/S0377-2217(01)00370-8

Fischer, H. (2011). *A history of the central limit theorem: From classical to modern probability theory*. Springer New York. https://doi.org/10.1007/978-0-387-87857-7

Fishburn, P. C. (1967). Additive utilities with incomplete product sets: Application to priorities and assignments. *Operations Research*, *15*(3), 537–542. https://doi.org/10.1287/opre.15.3.537

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, *7*(2), 179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

Fix, E., & Hodges, J. L. (1951). Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, *57*(3), 238–247. Retrieved September 16, 2022, from http://www.jstor.org/stable/1403797

Föllmer, H., & Kabanov, Y. M. (1997). Optional decomposition and lagrange multipliers. *Finance and Stochastics*, *2*(1), 69–81. https://doi.org/10.1007/s007800050033

Fowlkes, E. B., & Mallows, C. L. (1983). A Method for Comparing Two Hierarchical Clusterings. *Journal of the American Statistical Association*, *78*(383), 553–569. https://doi.org/10.1080/01621459.1983.10478008

Franc, V., & Hlavac, V. Multi-class support vector machine. In: In *2002 international conference on pattern recognition*. *2*. 2002, 236–239 vol.2. https://doi.org/10.1109/ICPR.2002.1048282.

Freudenberg, M. (2003). Composite indicators of country performance. https://doi.org/10.1787/405566708255

Freund, R., Wilson, W., & Sa, P. (2006). *Regression analysis*. Elsevier Science. https://books.google.es/books?id=Us4YE8lJVYMC

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232. Retrieved May 24, 2022, from http://www.jstor.org/stable/2699986

Friedman, J. H. (2002). Stochastic gradient boosting [Nonlinear Methods and Data Mining]. *Computational Statistics & Data Analysis*, *38*(4), 367–378. https://doi.org/10.1016/S0167-9473(01)00065-2

Fuglede, B., & Topsoe, F. Jensen-Shannon divergence and Hilbert space embedding. In: In *International symposium oninformation theory, 2004. isit 2004. proceedings.* 2004, 31–. https://doi.org/10.1109/ISIT.2004.1365067.

Fushiki, T. (2011). Estimation of prediction error by using k-fold cross-validation. *Statistics and Computing*, *21*(2), 137–146. https://doi.org/10.1007/s11222-009-9153-8

Gamil, Y., A. Abdullah, M., Abd Rahman, I., & Asad, M. M. (2020). Internet of things in construction industry revolution 4.0. *Journal of Engineering, Design and Technology*, *18*(5), 1091–1102. https://doi.org/10.1108/JEDT-06-2019-0164

Gan, G., Ma, C., & Wu, J. (2007). *Data clustering: Theory, algorithms, and applications*. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9780898718348

Garlapati, A., Krishna, D. R., Garlapati, K., Srikara Yaswanth, N. m., Rahul, U., & Narayanan, G. Stock price prediction using facebook prophet and arima models. In: In *2021 6th international conference for convergence in technology (i2ct)*. 2021, 1–7. https://doi.org/10.1109/I2CT51068.2021.9418057.

Gass, S. I., & Assad, A. A. (2014). History of operations research. In *Transforming research into action* (pp. 1–14). https://doi.org/10.1287/educ.1110.0084

Gass, S., & Assad, A. (2005). *An annotated timeline of operations research: An informal history*. Springer New York, NY. Retrieved May 25, 2022, from https://link.springer.com/book/9781402081125

Gentleman, R. (2008). *R programming for bioinformatics*. Chapman; Hall/CRC. https://doi.org/10.1201/9781420063684

Gondara, L. Medical image denoising using convolutional denoising autoencoders. In: In *2016 ieee 16th international conference on data mining workshops (icdmw)*. IEEE. 2016, 241–246. https://doi.org/10.1109/ICDMW.2016.0041.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. \url{http://www.deeplearningbook.org}

Gooijer, J. G. D., & Hyndman, R. J. (2006). 25 years of time series forecasting. *International Journal of Forecasting*, *22*, 443–473. https://doi.org/10.1016/j.ijforecast.2006.01.001

Gul, M., Celik, E., Aydin, N., Taskin Gumus, A., & Guneri, A. F. (2016). A state of the art literature review of vikor and its fuzzy extensions on applications. *Applied Soft Computing*, *46*, 60–89. https://doi.org/10.1016/j.asoc.2016.04.040

Gumus, M., & Kiran, M. S. Crude oil price forecasting using xgboost. In: In *2017 international conference on computer science and engineering (ubmk)*. IEEE. 2017, 1100–1103. https://doi.org/10.1109/UBMK.2017.8093500.

Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., & Yang, G.-Z. (2019). XAI—Explainable artificial intelligence. *Science robotics*, *4*(37), eaay7120. https://doi.org/10.1126/scirobotics.aay7120

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. An kNN model-based approach and its application in text categorization (A. Gelbukh, Ed.). In: *Computational linguistics and intelligent text processing* (A. Gelbukh, Ed.). Ed. by Gelbukh, A. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, 559–570. ISBN: 978-3-540-24630-5. https://doi.org/10.1007/978-3-540-24630-5_69.

Hacking, I. (2006). *The emergence of probability: A philosophical study of early ideas about probability, induction and statistical inference*. Cambridge University Press. https://books.google.es/books?id=ewqzOGDKYscC

Hall, P., Park, B. U., & Samworth, R. J. (2008). Choice of neighbor order in nearest-neighbor classification. *The Annals of Statistics*, *36*(5), 2135 –2152. https://doi.org/10.1214/07-AOS537

Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, *57*, 357–384. Retrieved August 15, 2022, from http://www.jstor.org/stable/1912559

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal*, *29*(2), 147–160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x

Hamsa, H., Indiradevi, S., & Kizhakkethottam, J. J. (2016). Student academic performance prediction model using decision tree and fuzzy genetic algorithm [1st Global Colloquium on Recent Advancements and Effectual Researches in Engineering, Science and Technology - RAEREST 2016 on April 22nd & 23rd April 2016]. *Procedia Technology*, *25*, 326–332. https://doi.org/10.1016/j.protcy.2016.08.114

Hariadhy, R. P., Sutoyo, E., & Pratiwi, O. N. Application of k-nearest neighbor algorithm for prediction of television advertisement rating (J. H. Abawajy, K.-K. R. Choo & H. Chiroma, Eds.). In: *International conference on emerging applications and technologies for industry 4.0 (eati'2020)* (J. H. Abawajy, K.-K. R. Choo & H. Chiroma, Eds.). Ed. by Abawajy, J. H., Choo, K.-K. R., & Chiroma, H. Cham: Springer International Publishing, 2021, 82–91. ISBN: 978-3-030-80216-5. https://doi.org/10.1007/978-3-030-80216-5_7.

Harrell, F. E., Califf, R. M., Pryor, D. B., Lee, K. L., & Rosati, R. A. (1982). Evaluating the yield of medical tests. *Jama*, *247*(18), 2543–2546.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hasan, R., Palaniappan, S., Raziff, A. R. A., Mahmood, S., & Sarker, K. U. Student academic performance prediction by using decision tree algorithm. In: In *2018 4th international conference on computer and information sciences (iccoins)*. 2018, 1–5. https://doi.org/10.1109/ICCOINS.2018.8510600.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. Springer New York. https://doi.org/10.1007/978-0-387-84858-7

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *Proceedings of the International Joint Conference on Neural Networks*, 1322–1328. https://doi.org/10.1109/IJCNN.2008.4633969

He, H., Zhang, W., & Zhang, S. (2018). A novel ensemble method for credit scoring: Adaption of different imbalance ratios. *Expert Systems with Applications*, *98*, 105–117. https://doi.org/10.1016/j.eswa.2018.01.012

He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition. In: In *2016 ieee conference on computer vision and pattern recognition (cvpr)*. 2016, 770–778. https://doi.org/10.1109/CVPR.2016.90.

Hebrank, J., & Wright, D. (1974). Spectral cues used in the localization of sound sources on the median plane. *The Journal of the Acoustical Society of America*, *56*(6), 1829–1834. https://doi.org/10.1121/1.1903520

Hidayah, I., Permanasari, A. E., & Ratwastuti, N. Student classification for academic performance prediction using neuro fuzzy in a conventional classroom. In: In *2013 international conference on information technology and electrical engineering (icitee)*. 2013, 221–225. https://doi.org/10.1109/ICITEED.2013.6676242.

Ho, T. K. Random decision forests. In: In *Proceedings of 3rd international conference on document analysis and recognition*. *1*. 1995, 278–282 vol.1. https://doi.org/10.1109/ICDAR.1995.598994.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hoerl, A. E., Hoerl, A. E., & Hoerl, C. Application of ridge analysis to regression problems. In: 1962.

Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.-C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H., & Le, Q. Searching for MobileNetV3. In: In *2019 ieee/cvf international conference on computer vision (iccv)*. 2019, 1314–1324. https://doi.org/10.1109/ICCV.2019.00140.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. https://doi.org/10.48550/ARXIV.1704.04861

Howson, C., & Urbach, P. (2006). *Scientific reasoning: The bayesian approach*. Open Court Publishing. https://books.google.es/books?id=3JusAwAAQBAJ

Hsu, P. L., & Robbins, H. (1947). Complete convergence and the law of large numbers. *Proceedings of the National Academy of Sciences*, *33*(2), 25–31. https://doi.org/10.1073/pnas.33.2.25

Huang, Y., Chen, C. H., & Huang, C. J. (2019). Motor fault detection and feature extraction using rnn-based variational autoencoder. *IEEE Access*, *7*, 139086–139096. https://doi.org/10.1109/ACCESS.2019.2940769

Huber, P. J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, *35*(1), 73–101. https://doi.org/10.1214/aoms/1177703732

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Hussain, S., Dahan, N. A., Ba-Alwib, F. M., & Ribata, N. (2018). Educational data mining and analysis of students' academic performance using weka. *Indonesian Journal of Electrical Engineering and Computer Science*, *9*(2), 447–459. https://doi.org/10.11591/ijeecs.v9.i2.pp447-459

Hwang, C.-L., & Yoon, K. (1981). *Multiple Attribute Decision Making* (Vol. 186). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-48318-9

Ihaka, R., & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, *5*(3), 299–314. https://doi.org/10.1080/10618600.1996.10474713

Iida, K., & Ishii, Y. (2011). Individualization of the head-related transfer functions on the basis of the spectral cues for sound localization. In *Principles and applications of spatial hearing* (pp. 159–178). https://doi.org/10.1142/9789814299312_0013

Imandoust, S., & Bolandraftar, M. (2013). Application of k-nearest neighbor (KNN) approach for predicting economic events theoretical background. *Int J Eng Res Appl*, *3*, 605–610. https://www.semanticscholar.org/paper/Application-of-K-Nearest-Neighbor-(KNN)-Approach-Imandoust-Bolandraftar/72007430928c7c99111ddf9bc2ad498470aad75b

Imran, M., Latif, S., Mehmood, D., & Shah, M. S. (2019). Student academic performance prediction using supervised learning techniques. *International Journal of Emerging Technologies in Learning*, *14*(14). https://doi.org/10.3991/ijet.v14i14.10310

Inuiguchi, M, Ichihashi, H, & Tanaka, H. (1990). Fuzzy programming: A survey of recent developments. *Stochastic versus fuzzy approaches to multiobjective mathematical programming under uncertainty*, 45–68. https://doi.org/10.1007/978-94-009-2111-5_4

Ishizaka, A., & Labib, A. (2011). Review of the main developments in the analytic hierarchy process. *Expert systems with applications*, *38*(11), 14336–14345. https://doi.org/10.1016/j.eswa.2011.04.143

Ivanova, I., Arcelus, F. J., & Srinivasan, G. (1999). An assessment of the measurement properties of the human development index. *Social indicators research*, *46*(2), 157–179. https://doi.org/10.1023/A:1006839208067

Jaccard, P. (1912). The distribution of the flora in the alpine zone. 1. *New phytologist*, *11*(2), 37–50. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x

Jacquet-Lagreze, E., & Siskos, J. (1982). Assessing a set of additive utility functions for multicriteria decision-making, the UTA method. *European journal of operational research*, *10*(2), 151–164. https://doi.org/10.1016/0377-2217(82)90155-2

Jacquet-Lagreze, E., & Siskos, Y. (2001). Preference disaggregation: 20 years of MCDA experience. *European Journal of Operational Research*, *130*(2), 233–245. https://doi.org/10.1016/S0377-2217(00)00035-7

Jahan, A., Mustapha, F., Sapuan, S. M., Ismail, M. Y., & Bahraminasab, M. (2012). A framework for weighting of criteria in ranking stage of material selection process. *International Journal of Advanced Manufacturing Technology*, *58*, 411–420. https://doi.org/10.1007/s00170-011-3366-7

Jain, A. K. (2010). Data clustering: 50 years beyond k-means [Award winning papers from the 19th International Conference on Pattern Recognition (ICPR)]. *Pattern Recognition Letters*, *31*(8), 651–666. https://doi.org/10.1016/j.patrec.2009.09.011

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc. https://dl.acm.org/doi/book/10.5555/46712

James, I. (2009). Claude Elwood Shannon 30 april 1916—24 february 2001. *55*, 257–265. https://doi.org/10.1098/rsbm.2009.0015

Jana, R. K., Ghosh, I., & Wallin, M. W. (2022). Taming energy and electronic waste generation in bitcoin mining: Insights from facebook prophet and deep neural network. *Technological Forecasting and Social Change*, *178*, 121584. https://doi.org/10.1016/j.techfore.2022.121584

Jiang, L., Cai, Z., Wang, D., & Jiang, S. Survey of improving k-nearest-neighbor for classification. In: In *Fourth international conference on fuzzy systems and knowledge discovery (fskd 2007)*. *1*. 2007, 679–683. https://doi.org/10.1109/FSKD.2007.552.

Johnson, S. G. (2020). The NLopt nonlinear-optimization package.

Joo, H.-T., & Kim, K.-J. Visualization of deep reinforcement learning using grad-cam: How ai plays atari games? In: In *2019 ieee conference on games (cog)*. 2019, 1–2. https://doi.org/10.1109/CIG.2019.8847950.

Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. *Advances in Psychology*, *121*, 471–495. https://doi.org/10.1016/S0166-4115(97)80111-2

Joyce, J. M. (2011). Kullback-leibler divergence. In M. Lovric (Ed.), *International encyclopedia of statistical science* (pp. 720–722). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04898-2_327

Kao, C. (2010). Weight determination for consistently ranking alternatives in multiple criteria decision analysis. *Applied Mathematical Modelling*, *34*(7), 1779–1787. https://doi.org/10.1016/j.apm.2009.09.022

Kaufman, L., & Rousseeuw, P. (1990). *Finding groups in data: An introduction to cluster analysis*. https://doi.org/10.2307/2532178

Kaya, E., Agca, M., Adiguzel, F., & Cetin, M. (2019). Spatial data analysis with r programming for environment. *Human and Ecological Risk Assessment: An International Journal*, *25*(6), 1521–1530. https://doi.org/10.1080/10807039.2018.1470896

Ke, G., Meng, Q., Finely, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. LightGBM: A highly efficient gradient boosting decision tree. In: In *Advances in neural information processing systems 30 (nip 2017)*. 2017. https://www.microsoft.com/en-us/research/publication/lightgbm-a-highly-efficient-gradient-boosting-decision-tree/

Keser, S. B., & Aghalarova, S. (2022). Hela: A novel hybrid ensemble learning algorithm for predicting academic performance of students. *Education and Information Technologies*, *27*(4), 4521–4552. https://doi.org/10.1007/s10639-021-10780-0

Khoo, C. S., & Johnkhan, S. B. (2018). Lexicon-based sentiment analysis: Comparative evaluation of six sentiment lexicons. *Journal of Information Science*, *44*, 491–511. https://doi.org/10.1177/0165551517703514

Kim, K., Lee, D., & Essa, I. Gaussian process regression flow for analysis of motion trajectories. In: In *2011 international conference on computer vision*. 2011, 1164–1171. https://doi.org/10.1109/ICCV.2011.6126365.

Kim, K. J. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*, 307–319. https://doi.org/10.1016/S0925-2312(03)00372-2

Kim, T., Lee, J., & Nam, J. Sample-level cnn architectures for music auto-tagging using raw waveforms. In: In *2018 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE. 2018, 366–370.

Kingma, D. P., & Lei Ba, J. (2014). ADAM: A Method For Stochastic Optimization. *arXiv*. http://arxiv.org/abs/1412.6980v9

Kohzadi, N., Boyd, M. S., Kermanshahi, B., & Kaastra, I. (1996). A comparison of artificial neural network and time series models for forecasting commodity prices [Financial Applications, Part I]. *Neurocomputing*, *10*(2), 169–181. https://doi.org/10.1016/0925-2312(95)00020-8

Kolmogorov, A. N. (1950). *Foundations of the theory of probability*. New York, USA: Chelsea Pub. Co. https://archive.org/details/foundationsofthe00kolm/page/n9/mode/2up

Kou, G., LU, Y., Peng, Y., & Shi, Y. (2012). Evaluation of classification algorithms using MCDM and rank correlation. *International Journal of Information Technology & Decision Making*, *11*(01), 197–225. https://doi.org/10.1142/S0219622012500095

Koutník, J., Greff, K., Gomez, F., & Schmidhuber, J. (2014). A Clockwork RNN. https://doi.org/10.48550/ARXIV.1402.3511

Kraus, M. W., Huang, C., & Keltner, D. (2010). Tactile communication, cooperation, and performance: an ethological study of the NBA. *Emotion*, *10*(5), 745. https://doi.org/10.1037/a0019382

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500. *European Journal of Operational Research*, *259*, 689–702. https://doi.org/10.1016/j.ejor.2016.10.031

Krizhevsky, A., Sutskever, I., & Hinton, G. E. Imagenet classification with deep convolutional neural networks (F. Pereira, C. Burges, L. Bottou & K. Weinberger, Eds.). In: *Advances in neural information processing systems* (F. Pereira, C. Burges, L. Bottou & K. Weinberger, Eds.). Ed. by Pereira, F., Burges, C., Bottou, L., & Weinberger, K. *25*. Curran Associates, Inc., 2012. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Kuhn, M., & Wickham, H. (2020). *Tidymodels: A collection of packages for modeling and machine learning using tidyverse principles.* https://www.tidymodels.org

Kumar, R., Singh, S., Bilga, P. S., Jatin, Singh, J., Singh, S., Scutaru, M.-L., & Pruncu, C. I. (2021). Revealing the benefits of entropy weights method for multi-objective optimization in machining operations: A critical review. *Journal of Materials Research and Technology*, *10*, 1471–1492. https://doi.org/https://doi.org/10.1016/j.jmrt.2020.12.114

Kuo, M.-S., & Liang, G.-S. (2011). Combining VIKOR with GRA techniques to evaluate service quality of airports under fuzzy environment. *Expert Systems with Applications*, *38*(3), 1304–1312. https://doi.org/10.1016/j.eswa.2010.07.003

Kwak, S. G., & Kim, J. H. (2017). Central limit theorem: The cornerstone of modern statistics. *kja*, *70*(2), 144–156. https://doi.org/10.4097/kjae.2017.70.2.144

Lachtermacher, G., & Fuller, J. D. (1995). Back propagation in time-series forecasting. *Journal of Forecasting*, *14*(4), 381–393. https://doi.org/10.1002/for.3980140405

Laporte, G., & Martello, S. (1990). The selective travelling salesman problem. *Discrete Applied Mathematics*, *26*(2), 193–207. https://doi.org/10.1016/0166-218X(90)90100-Q

Laudani, A., Lozito, G. M., Fulginei, F. R., & Salvini, A. (2015). On training efficiency and computational costs of a feed forward neural network: A review. *Intell. Neuroscience*, *2015*. https://doi.org/10.1155/2015/818243

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, *1*(4), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Leekwijck, W. V., & Kerre, E. E. (1999). Defuzzification: Criteria and classification. *Fuzzy Sets and Systems*, *108*(2), 159–178. https://doi.org/10.1016/S0165-0114(97)00337-0

León, T., Liern, V., Ruiz, J., & Sirvent, I. (2003). A fuzzy mathematical programming approach to the assessment of efficiency with DEA models. *Fuzzy sets and systems*, *139*(2), 407–419. https://doi.org/10.1016/S0165-0114(02)00608-5

Lepenioti, K., Bousdekis, A., Apostolou, D., & Mentzas, G. (2020). Prescriptive analytics: Literature review and research challenges. *International Journal of Information Management*, *50*, 57–70. https://doi.org/10.1016/j.ijinfomgt.2019.04.003

Li, L., Zheng, N.-N., & Wang, F.-Y. (2019). On the crossroad of artificial intelligence: A revisit to alan turing and norbert wiener. *IEEE Transactions on Cybernetics*, *49*(10), 3618–3626. https://doi.org/10.1109/TCYB.2018.2884315

Li, Q., & Liny, N. (2010). The Bayesian elastic net. *Bayesian Analysis*, *5*(1), 151–170. https://doi.org/10.1214/10-BA506

Li, S., Li, W., Cook, C., Zhu, C., & Gao, Y. (2018). Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN. *arXiv*. http://arxiv.org/abs/1803.04831

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21. https://doi.org/10.1109/TNNLS.2021.3084827

Liern, V., Parada-Rico, S., & Blasco-Blasco, O. (2020a). Construction of quality indicators based on pre-established goals: Application to a colombian public university. *Mathematics*, *8*, 1075. https://doi.org/https://doi.org/10.3390/math8071075

Liern, V., & Pérez-Gladish, B. (2020). Multiple criteria ranking method based on functional proximity index: un-weighted TOPSIS. *Annals of Operations Research 2020*, 1–23. https://doi.org/10.1007/S10479-020-03718-1

Liern, V. (2005). Fuzzy tuning systems: The mathematics of musicians. *Fuzzy sets and systems*, *150*(1), 35–52. https://doi.org/10.1016/j.fss.2004.04.002

Liern, V., Parada-Rico, S. E., & Blasco-Blasco, O. (2020b). Construction of quality indicators based on pre-established goals: Application to a colombian public university. *Mathematics*, *8*(7). https://doi.org/10.3390/math8071075

Liern, V., & Pérez-Gladish, B. (2021). Building composite indicators with unweighted-topsis. *IEEE Transactions on Engineering Management*. https://doi.org/10.1109/TEM.2021.3090155

Liu, T., Moore, A. W., & Gray, A. (2003). New algorithms for efficient high dimensional non-parametric classification. *Journal of Machine Learning Research*, *7*, 2006. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.89.738

Ljung, L. (2002). Prediction error estimation methods. *Circuits, Systems and Signal Processing*, *21*(1), 11–21. https://doi.org/10.1007/BF01211648

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137. https://doi.org/10.1109/TIT.1982.1056489

Loh, W.-Y. (2014). Fifty years of classification and regression trees. *International Statistical Review*, *82*(3), 329–348. https://doi.org/10.1111/insr.12016

López-García, A. (2021a). uwTOPSIS: Unweighted TOPSIS method [Last commit version: c15ff5931c9a8857c1778f39e0b0a6e0a8cdc6fd].

López-García, A. (2021b). uwVIKOR: Unweighted VIKOR method [Last commit version: 8a96fe446f6356430aa10d226d832c50d6a6bfa2].

López-García, A. (2022a). GPMM: Collection of generalized p-mean models with classic, fuzzy and un-weighted approach [Last commit version: cec0dbf5659d66c647185ae2330a00117adc0531].

López-García, A. (2022b). SpectroMap: peak detection algorithm that computes the constellation map for a given signal [Last commit version: 12d29136f2c8c327b81b2a6d366ed342703fdfb6].

López-García, A., Martínez-Rodríguez, B., & Liern, V. A proposal to compare the similarity between musical products. one more step for automated plagiarism detection? (M. Montiel, O. A. Agustín-Aquino, F. Gómez, J. Kastine, E. Lluis-Puebla & B. Milam, Eds.). In: *Mathematics and computation in music* (M. Montiel, O. A. Agustín-Aquino, F. Gómez, J. Kastine, E. Lluis-Puebla & B. Milam, Eds.). Ed. by Montiel, M., Agustín-Aquino, O. A., Gómez, F., Kastine, J., Lluis-Puebla, E., & Milam, B. Cham: Springer International Publishing, 2022, 192–204. ISBN: 978-3-031-07015-0. https://doi.org/10.1007/978-3-031-07015-0_16.

Lu, X., Tsao, Y., Matsuda, S., & Hori, C. Speech enhancement based on deep denoising autoencoder. In: In *Proc. interspeech 2013*. *2013*. 2013, 436–440. https://doi.org/10.21437/Interspeech.2013-130.

Lundberg, S. M., & Lee, S.-I. A unified approach to interpreting model predictions (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett, Eds.). In: *Advances in neural information processing systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett, Eds.). Ed. by Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., & Garnett, R. *30*. Curran Associates, Inc., 2017. https://doi.org/10.48550/ARXIV.1705.07874.

López-García, A. (2022). Spectromap: Peak detection algorithm for audio fingerprinting. https://doi.org/10.48550/ARXIV.2211.00982

López-García, A., & Benítez, R. (2022). Reputation analysis of news sources in twitter: Particular case of spanish presidential election in 2019. https://doi.org/10.48550/ARXIV.2201.08609

MacQueen, J. Some methods for classification and analysis of multivariate observations. In: In *5th berkeley symp. math. statist. probability*. 1967, 281–297. https://www.cs.cmu.edu/~bhiksha/courses/mlsp.fall2010/class14/macqueen.pdf

Madhulatha, T. S. (2012). An overview on clustering methods. *CoRR*, *abs/1205.1117*. http://arxiv.org/abs/1205.1117

Majumdar, A. (2018). Blind denoising autoencoder. *IEEE transactions on neural networks and learning systems*, *30*(1), 312–317. https://doi.org/10.1109/TNNLS.2018.2838679

Malakar, P., Balaprakash, P., Vishwanath, V., Morozov, V., & Kumaran, K. Benchmarking machine learning methods for performance modeling of scientific applications. In: In *2018 ieee/acm performance modeling, benchmarking and simulation of high performance computer systems (pmbs)*. 2018, 33–44. https://doi.org/10.1109/PMBS.2018.8641686.

Malik, R. Q., Zaidan, A. A., Zaidan, B. B., Ramli, K. N., Albahri, O. S., Kareem, Z. H., Ameen, H. A., Garfan, S. S., Mohammed, A., Zaidan, R. A., & Salih, M. M. (2022). Novel roadside unit positioning framework in the context of the vehicle-to-infrastructure communication system based on ahp — entropy for weighting and borda — vikor for uniform ranking. *International Journal of Information Technology & Decision Making*, *21*(04), 1233–1266. https://doi.org/10.1142/S0219622021500061

Mardani, A., Zavadskas, E. K., Govindan, K., Amat Senin, A., & Jusoh, A. (2016). VIKOR technique: A systematic review of the state of the art literature on methodologies and applications. *Sustainability*, *8*(1), 37. https://www.mdpi.com/2071-1050/8/1/37

Mareschal, B. (1988). Weight stability intervals in multicriteria decision aid. *European Journal of Operational Research*, *33*(1), 54–64. https://doi.org/https://doi.org/10.1016/0377-2217(88)90254-8

Mareschal, B., Brans, J. P., & Vincke, P. (1984). PROMETHEE: A new family of outranking methods in multicriteria analysis, 477–490. https://ideas.repec.org/p/ulb/ulbeco/2013-9305.html

Martínez-Rodríguez, B., & Liern, V. A fuzzy-clustering based approach for measuring similarity between melodies. In: In *International conference on mathematics and computation in music*. Springer. 2017, 279–290. ISBN: 978-3-319-71827-9. https://doi.org/10.1007/978-3-319-71827-9_21.

Martínez-Rodríguez, B., & Liern, V. Mercury: A software based on fuzzy clustering for computer-assisted composition. In: In *International conference on mathematics and computation in music*. Springer. 2019, 236–247. ISBN: 978-3-030-21392-3. https://doi.org/10.1007/978-3-030-21392-3_19.

Marzban, C. (2004). The ROC curve and the area under it as performance measures. *Weather and Forecasting*, *19*(6), 1106–1114. https://doi.org/10.1175/825.1

Matthews, B. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, *405*(2), 442–451. https://doi.org/10.1016/0005-2795(75)90109-9

McCloskey, J. F. (1987). Or forum—the beginnings of operations research: 1934–1941. *Operations Research*, *35*(1), 143–152. https://doi.org/10.1287/opre.35.1.143

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biophysics, vol. 5. *Journal of Symbolic Logic*, *9*(2), 49–50. https://doi.org/10.2307/2268029

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*, 1093–1113. https://doi.org/10.1016/j.asej.2014.04.011

Messenger, R., & Mandell, L. (1972). A modal search technique for predictibe nominal scale multivariate analys. *Journal of the American Statistical Association*, *67*(340), 768–772. Retrieved July 18, 2022, from http://www.jstor.org/stable/2284634

Miao, Y., Gowayyed, M., & Metze, F. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In: In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 2015, 167–174. https://doi.org/10.1109/ASRU.2015.7404790.

Middlebrooks, J. C. (1999). Virtual localization improved by scaling nonindividualized external-ear transfer functions in frequency. *The Journal of the Acoustical Society of America*, *106*(3), 1493–1510. https://doi.org/10.1121/1.427147

Mikołajec, K., Maszczyk, A., & Zając, T. (2013). Game indicators determining sports performance in the NBA. *Journal of human kinetics*, *37*, 145. https://doi.org/10.2478/hukin-2013-0035

Miraz, M. H., Ali, M., Excell, P. S., & Picking, R. A review on internet of things (iot), internet of everything (ioe) and internet of nano things (iont). In: In *2015 internet technologies and applications (ita)*. 2015, 219–224. https://doi.org/10.1109/ITechA.2015.7317398.

Mishev, K., Gjorgjevikj, A., Vodenska, I., Chitkushev, L. T., & Trajanov, D. (2020). Evaluation of sentiment analysis in finance: From lexicons to transformers. *IEEE Access*, *8*, 131662–131682. https://doi.org/10.1109/ACCESS.2020.3009626

Mittal, K., Jain, A., Vaisla, K. S., Castillo, O., & Kacprzyk, J. (2020). A comprehensive review on type 2 fuzzy logic applications: Past, present and future. *Engineering Applications of Artificial Intelligence*, *95*, 103916. https://doi.org/10.1016/j.engappai.2020.103916

Mohammed, M. A., Abdulkareem, K. H., Al-Waisy, A. S., Mostafa, S. A., Al-Fahdawi, S., Dinar, A. M., Alhakami, W., BAZ, A., Al-Mhiqani, M. N., Alhakami, H., Arbaiy, N., Maashi, M. S., Mutlag, A. A., García-Zapirain, B., & De La Torre Díez, I. D. L. T. (2020). Benchmarking Methodology for Selection of Optimal COVID-19 Diagnostic Model Based on Entropy and TOPSIS Methods. *IEEE Access*, *8*, 99115–99131. https://doi.org/10.1109/ACCESS.2020.2995597

Morgan, J. N., & Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, *58*(302), 415–434. https://doi.org/10.1080/01621459.1963.10500855

Mosier, C. I. (1951). The need and means of cross validation. i. problems and designs of cross-validation. *Educational and Psychological Measurement*, *11*(1), 5–11. https://doi.org/10.1177/001316445101100101

Murtagh, F. (1991). Multilayer perceptrons for classification and regression. *Neurocomputing*, *2*(5), 183–197. https://doi.org/10.1016/0925-2312(91)90023-5

Mustaqeem & Kwon, S. (2020). A cnn-assisted enhanced audio signal processing for speech emotion recognition. *Sensors*, *20*(1). https://doi.org/10.3390/s20010183

Nanda, S. J., & Panda, G. (2014). A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary Computation*, *16*, 1–18. https://doi.org/10.1016/j.swevo.2013.11.003

Nandy, A., & Singh, P. K. (2021). Application of fuzzy dea and machine learning algorithms in efficiency estimation of paddy producers of rural eastern india. *Benchmarking: An International Journal*, *28*(1), 229–248. https://doi.org/10.1108/BIJ-01-2020-0012

Naranjo-Alcazar, J., Perez-Castanos, S., Lopez-Garcia, A., Zuccarello, P., Cobos, M., & Ferri, F. J. (2021). Squeeze-excitation convolutional recurrent neural networks for audio-visual scene classification. https://doi.org/10.48550/ARXIV.2107.13180

Nasseri, A. A., Tucker, A., & Cesare, S. D. (2015). Quantifying stocktwits semantic terms' trading behavior in financial markets: An effective application of decision tree algorithms. *Expert Systems with Applications*, *42*, 9192–9210. https://doi.org/10.1016/j.eswa.2015.08.008

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, *7*, 21. https://doi.org/10.3389/fnbot.2013.00021

Nemes, L., & Kiss, A. (2021). Social media sentiment analysis based on covid-19. *Journal of Information and Telecommunication*, *5*, 1–15. https://doi.org/10.1080/24751839.2020.1790793

Nutt, P. C. (1980). Comparing methods for weighting decision criteria. *Omega*, *8*(2), 163–172. https://doi.org/https://doi.org/10.1016/0305-0483(80)90020-1

Nădăban, S., Dzitac, S., & Dzitac, I. (2016). Fuzzy topsis: A general view [Promoting Business Analytics and Quantitative Management of Technology: 4th International Conference on Information Technology and Quantitative Management (ITQM 2016)]. *Procedia Computer Science*, *91*, 823–831. https://doi.org/10.1016/j.procs.2016.07.088

Odu, G. (2019). Weighting methods for multi-criteria decision making technique. *Journal of Applied Sciences and Environmental Management*, *23*(8), 1449–1457.

Ogunleye, A., & Wang, Q.-G. (2020). Xgboost model for chronic kidney disease diagnosis. *IEEE/ACM transactions on computational biology and bioinformatics*, *17*(6), 2131–2140. https://doi.org/10.1109/TCBB.2019.2911071

Omeiza, D., Speakman, S., Cintas, C., & Weldermariam, K. (2019). Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models. https://doi.org/10.48550/ARXIV.1908.01224

Opricovic, S. (1998). Multicriteria optimization of civil engineering systems. *Faculty of civil engineering, Belgrade*, *2*(1), 5–21.

Opricovic, S. (2007). A fuzzy compromise solution for multicriteria problems. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *15*(03), 363–380. https://doi.org/10.1142/S0218488507004728

Opricovic, S. (2011). Fuzzy vikor with an application to water resources planning. *Expert Systems with Applications*, *38*(10), 12983–12990. https://doi.org/10.1016/j.eswa.2011.04.097

Opricovic, S., & Tzeng, G.-H. (2002). Multicriteria planning of post-earthquake sustainable reconstruction. *Computer-Aided Civil and Infrastructure Engineering*, *17*(3), 211–220. https://doi.org/10.1111/1467-8667.00269

Opricovic, S., & Tzeng, G.-H. (2003b). Fuzzy multicriteria model for postearthquake land-use planning. *Natural Hazards Review*, *4*(2), 59–64. https://doi.org/10.1061/(ASCE)1527-6988(2003)4:2(59)

Opricovic, S., & Tzeng, G.-H. (2003a). Defuzzification within a multicriteria decision model. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *11*(05), 635–652. https://doi.org/10.1142/S0218488503002387

Opricovic, S., & Tzeng, G.-H. (2004). Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS. *European Journal of Operational Research*, *156*(2), 445–455. https://doi.org/10.1016/S0377-2217(03)00020-1

Opricovic, S., & Tzeng, G.-H. (2007). Extended VIKOR method in comparison with outranking methods. *European Journal of Operational Research*, *178*(2), 514–529. https://doi.org/https://doi.org/10.1016/j.ejor.2006.01.020

Ore, Ø. (1953). *Cardano: The gambling scholar* (Vol. 4972). Princeton University Press. Retrieved August 13, 2022, from http://www.jstor.org/stable/j.ctv39x7h7

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. https://doi.org/10.48550/ARXIV.1511.08458

Ouenniche, J., Pérez-Gladish, B., & Bouslah, K. (2018). An out-of-sample framework for TOPSIS-based classifiers with application in bankruptcy prediction. *Technological Forecasting and Social Change*, *131*, 111–116. https://doi.org/10.1016/j.techfore.2017.05.034

Paliwal, M., & Kumar, U. A. (2009). A study of academic performance of business school graduates using neural network and statistical techniques. *Expert Systems with Applications*, *36*(4), 7865–7872. https://doi.org/https://doi.org/10.1016/j.eswa.2008.11.003

Pamucar, D., Stevic, Z., & Sremac, S. (2018). A new model for determining weight coefficients of criteria in mcdm models: Full consistency method (fucom). *Symmetry*, *10*(9). https://doi.org/10.3390/sym10090393

Pandala, S. R. (2019). *Lazy predict* [Lazy Predict version 0.2.10 documentation: https://lazypredict.readthedocs.io/en/latest/]. https://github.com/shankarpandala/lazypredict

Parada, S. E., Blasco-Blasco, O., & Liern, V. (2019). Adequacy Indicators Based on Pre-established Goals: An Implementation in a Colombian University. *Social Indicators Research*, *143*(1). https://doi.org/10.1007/s11205-018-1979-z

Parada, S., Blasco-Blasco, O., & Liern, V. (2017). Construcción de indicadores basada en medidas de similitud con ideales. Una aplicación al cálculo de índices de adecuación y de excelencia. *Recta*, *18*, 119–135. https://doi.org/https://doi.org/10.24309/recta.2017.18.2.02

Park, J. H., Cho, H. J., & Kwun, Y. C. (2011). Extension of the VIKOR method for group decision making with interval-valued intuitionistic fuzzy information. *Fuzzy Optimization and Decision Making*, *10*(3), 233–253. https://doi.org/10.1007/s10700-011-9102-9

Pasero, E., Raimondo, G., & Ruffa, S. (2010). Mulp: A multi-layer perceptron application to long-term, out-of-sample time series prediction. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6064 LNCS*, 566–575. https://doi.org/10.1007/978-3-642-13318-3_70

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., . . . Chintala, S. Pytorch: An imperative style, high-performance deep learning library (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Garnett, Eds.). In: *Advances in neural information processing systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Garnett, Eds.). Ed. by Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., & Garnett, R. Curran Associates, Inc., 2019, pp. 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572. https://doi.org/10.1080/14786440109462720

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Peirce, C. S., & Eisele, C. (1976). *The new elements of mathematics* (Vol. 4). Mouton The Hague. https://books.google.es/books?id=iY3wxwEACAAJ

Pejlare, J., & Bråting, K. (2019). Writing the history of mathematics: Interpretations of the mathematics of the past and its relation to the mathematics of today. In B. Sriraman (Ed.), *Handbook of the mathematics of the arts and sciences* (pp. 1–26). Springer International Publishing. https://doi.org/10.1007/978-3-319-70658-0_63-1

Peng, R. D. (2016). *R programming for data science*. Leanpub Victoria, BC, Canada. https://bookdown.org/rdpeng/rprogdatascience/

Peng, Y., Kou, G., Shi, Y., & Chen, Z. (2008). A multi-criteria convex quadratic programming model for credit data analysis. *Decision Support Systems*, *44*(4), 1016–1030. https://doi.org/10.1016/j.dss.2007.12.001

Pinciroli Vago, N. O., Milani, F., Fraternali, P., & da Silva Torres, R. (2021). Comparing cam algorithms for the identification of salient image features in iconography artwork analysis. *Journal of Imaging*, *7*(7). https://doi.org/10.3390/jimaging7070106

Pollack, J. B. (1990). Recursive Distributed Representations. *Artificial Intelligence*, *46*(1–2), 77–105. https://doi.org/10.1016/0004-3702(90)90005-K

Power, D., Heavin, C, McDermott, J, & Daly, M. (2018). Defining business analytics: An empirical approach. *Journal of Business Analytics*, *1*(1), 40–53. https://doi.org/10.1080/2573234X.2018.1507605

Pradhan, A. (2012). Support vector machine-a survey. *International Journal of Emerging Technology and Advanced Engineering*, *2*(8), 82–85. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.366.905

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2017). Catboost: Unbiased boosting with categorical features. https://doi.org/10.48550/ARXIV.1706.09516

Pulliam, H. R. (1975). Diet optimization with nutrient constraints. *The American Naturalist*, *109*(970), 765–768. https://doi.org/10.1086/283041

Ramson, S. J., Vishnu, S., & Shanmugam, M. Applications of internet of things (IoT) – an overview. In: In *2020 5th international conference on devices, circuits and systems (icdcs)*. 2020, 92–95. https://doi.org/10.1109/ICDCS48716.2020.243556.

Ramík, J., & ímánek, J. (1985). Inequality relation between fuzzy numbers and its use in fuzzy optimization. *Fuzzy Sets and Systems*, *16*(2), 123–138. https://doi.org/10.1016/S0165-0114(85)80013-0

Rebentrost, P., Mohseni, M., & Lloyd, S. (2014). Quantum support vector machine for big data classification. *Phys. Rev. Lett.*, *113*, 130503. https://doi.org/10.1103/PhysRevLett.113.130503

Rekabsaz, N., Lupu, M., Baklanov, A., Hanbury, A., Dür, A., Anderson, L., & Wien, T. (2017). Volatility prediction using financial disclosures sentiments with word embedding-based ir models. *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, *1*, 1712–1721. https://doi.org/10.18653/v1/P17-1157

Renault, T. (2020). Sentiment analysis and machine learning in finance: A comparison of methods and models on one million messages. *Digital Finance*, *2*, 1–13. https://doi.org/10.1007/s42521-019-00014-x

Ribeiro, M. T., Singh, S., & Guestrin, C. " why should i trust you?" explaining the predictions of any classifier. In: In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, 1135–1144. https://doi.org/10.48550/ARXIV.1602.04938.

Rokach, L., & Maimon, O. (2005). Clustering methods. In O. Maimon & L. Rokach (Eds.), *Data mining and knowledge discovery handbook* (pp. 321–352). Springer US. https://doi.org/10.1007/0-387-25465-X_15

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton*. Cornell Aeronautical Laboratory, Buffalo, New York. https://books.google.es/books?id=P_XGPgAACAAJ

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, *65 6*, 386–408. https://doi.org/10.1037/H0042519

Rosenblatt, F. (1963). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. *American Journal of Psychology*, *76*, 705. https://doi.org/10.2307/1419730

Roszkowska, E. (2013). Rank ordering criteria weighting methods–a comparative overview. *Optimum. Studia Ekonomiczne*, (5 (65)), 14–33.

Roy, B. (1968). Classement et choix en présence de points de vue multiples. *Revue française d'informatique et de recherche opérationnelle*, *2*(8), 57–75. http://eudml.org/doc/104443

Roy, B. The outranking approach and the foundations of ELECTRE methods (C. A. Bana e Costa, Ed.). In: *Readings in multiple criteria decision aid* (C. A. Bana e Costa, Ed.). Ed. by Bana e Costa, C. A. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, 155–183. ISBN: 978-3-642-75935-2. https://doi.org/10.1007/978-3-642-75935-2_8.

Roy, B. (1996). *Multicriteria methodology for decision aiding* (Vol. 12). Springer Science & Business Media. https://doi.org/10.1007/978-1-4757-2500-1

Ruder, S. (2016). An overview of gradient descent optimization algorithms, 1–14. http://arxiv.org/abs/1609.04747

Saaty, R. (1987). The analytic hierarchy process—what it is and how it is used. *Mathematical Modelling*, *9*(3), 161–176. https://doi.org/https://doi.org/10.1016/0270-0255(87)90473-8

Saaty, T. L. (1977). A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, *15*(3), 234–281. https://doi.org/https://doi.org/10.1016/0022-2496(77)90033-5

Saaty, T. (1980). *The Analytic Hierarchy Process*. McGraw-Hill, New York.

Saaty, T., & Vargas, L. (2012). *Models, methods, concepts and applications of the analytic hierarchy process* (Vol. 175). Springer. https://doi.org/10.1007/978-1-4614-3597-6

Sackman, H. (1974). *Delphi assessment: Expert opinion, forecasting, and group process*. RAND Corporation. https://www.rand.org/pubs/reports/R1283.html

Sahu, D. A., Sahu, N. K., & Sahu, A. (2015). Benchmarking CNC machine tool using hybrid-fuzzy methodology:: A multi-indices decision making (MCDM) approach. *International Journal of Fuzzy System Applications*, *4*, 28–46. https://doi.org/10.4018/IJFSA.2015040103

Saisana, M., Saltelli, A., & Tarantola, S. (2005). Uncertainty and sensitivity analysis techniques as tools for the quality assessment of composite indicators. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, *168*(2), 307–323. https://doi.org/10.1111/j.1467-985X.2005.00350.x

Salih, M. M., Al-Qaysi, Z. T., Shuwandy, M. L., Ahmed, M. A., Hasan, K. F., & Muhsen, Y. R. (2022). A new extension of fuzzy decision by opinion score method based on fermatean fuzzy: A benchmarking covid-19 machine learning methods [3]. *Journal of Intelligent & Fuzzy Systems*, *43*, 3549–3559. https://doi.org/10.3233/JIFS-220707

Salih, M. M., Zaidan, B., Zaidan, A., & Ahmed, M. A. (2019). Survey on fuzzy topsis state-of-the-art between 2007 and 2017. *Computers & Operations Research*, *104*, 207–227. https://doi.org/10.1016/j.cor.2018.12.019

Samoly, K. (2012). The history of the abacus. *Ohio Journal Of School Mathematics*, (65). http://hdl.handle.net/1811/78206

Sanayei, A., Farid Mousavi, S., & Yazdankhah, A. (2010). Group decision making process for supplier selection with vikor under fuzzy environment. *Expert Systems with Applications*, *37*(1), 24–30. https://doi.org/10.1016/j.eswa.2009.04.063

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: In *2018 ieee/cvf conference on computer vision and pattern recognition*. 2018, 4510–4520. https://doi.org/10.1109/CVPR.2018.00474.

Sapiezynski, P., Kassarnig, V., & Wilson, C. Academic performance prediction in a gender-imbalanced environment. In: In *Fatrec workshop on responsible recommendation proceedings*. 2017, 49–58. https://doi.org/10.18122/B20Q5R.

Sasaki, Y. (2007). The truth of the F-measure. *Teach tutor mater*, *1*(5), 1–5. https://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf

Sayadi, M. K., Heydari, M., & Shahanaghi, K. (2009). Extension of vikor method for decision making problem with interval numbers. *Applied Mathematical Modelling*, *33*(5), 2257–2262. https://doi.org/10.1016/j.apm.2008.06.002

Sazli, M. H. (2006). A brief review of feed-forward neural networks. *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, *50*(01). https://doi.org/10.1501/commua1-2_0000000026

Schetinin, V., Fieldsend, J. E., Partridge, D., Coats, T. J., Krzanowski, W. J., Everson, R. M., Bailey, T. C., & Hernandez, A. (2007). Confident interpretation of bayesian decision tree ensembles for clinical applications. *IEEE Transactions on Information Technology in Biomedicine*, *11*(3), 312–319. https://doi.org/10.1109/TITB.2006.880553

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Schmidt, F. L. (1971). The relative efficiency of regression and simple unit predictor weights in applied differential psychology. *Educational and Psychological Measurement*, *31*(3), 699–714. https://doi.org/10.1177/001316447103100310

Scholkopf, B., Mika, S., Burges, C., Knirsch, P., Muller, K.-R., Ratsch, G., & Smola, A. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, *10*(5), 1000–1017. https://doi.org/10.1109/72.788641

Schulz, E., Speekenbrink, M., & Krause, A. (2018). A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, *85*, 1–16. https://doi.org/10.1016/j.jmp.2018.03.001

Schumaker, R. P., & Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems*, *27*. https://doi.org/10.1145/1462198.1462204

Schuster, M., & Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. https://doi.org/10.1109/78.650093

Seabold, S., & Perktold, J. Statsmodels: Econometric and statistical modeling with Python. In: In *9th python in science conference*. 2010. https://www.statsmodels.org/stable/index.html

Searle, C. L., Braida, L. D., Davis, M. F., & Colburn, H. S. (1976). Model for auditory localization. *The Journal of the Acoustical Society of America*, *60*(5), 1164–1175. https://doi.org/10.1121/1.381219

Seeger, M. (2004). Gaussian processes for machine learning. *International journal of neural systems*, *14*(02), 69–106. https://doi.org/10.1142/S0129065704001899

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In: In *2017 ieee international conference on computer vision (iccv)*. 2017, 618–626. https://doi.org/10.1109/ICCV.2017.74.

Selvin, S., Vinayakumar, R, Gopalakrishnan, E. A, Menon, V. K., & Soman, K. P. Stock price prediction using LSTM, RNN and CNN-sliding window model. In: In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2017, 1643–1647. https://doi.org/10.1109/ICACCI.2017.8126078.

Seneta, E. (2013). A Tricentenary history of the Law of Large Numbers. *Bernoulli*, *19*(4), 1088 –1121. https://doi.org/10.3150/12-BEJSP12

Shakor, S. M. (2022). Benchmarking machine learning methods COVID-19 classification using MCDM technique. *International Journal of Science and Research (IJSR)*, *11*(04), 678–681. https://www.ijsr.net/get_abstract.php?paper_id=SR22411222004

Shanian, A., Milani, A., Carson, C., & Abeyaratne, R. (2008). A new application of electre iii and revised simos' procedure for group material selection under weighting uncertainty. *Knowledge-Based Systems*, *21*(7), 709–720. https://doi.org/https://doi.org/10.1016/j.knosys.2008.03.028

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Shinde, P. P., & Shah, S. A review of machine learning and deep learning applications. In: In *2018 fourth international conference on computing communication control and automation (iccubea)*. 2018, 1–6. https://doi.org/10.1109/ICCUBEA.2018.8697857.

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019). A comparison of arima and lstm in forecasting time series. *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, 1394–1401. https://doi.org/10.1109/ICMLA.2018.00227

Sidey-Gibbons, J. A. M., & Sidey-Gibbons, C. J. (2019). Machine learning in medicine: A practical introduction. *BMC Medical Research Methodology*, *19*(1), 64. https://doi.org/10.1186/s12874-019-0681-4

Sigler, K. J., & Sackley, W. H. (2000). NBA players: are they paid for performance? *Managerial finance*. https://doi.org/10.1108/03074350010766783

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. https://doi.org/10.48550/ARXIV.1409.1556

Singh, S., & Gupta, P. (2014). Comparative study id3, cart and c4. 5 decision tree algorithm: A survey. *International Journal of Advanced Information Science and Technology (IJAIST)*, *27*(27), 97–103. https://doi.org/10.14569/SpecialIssue.2014.040203

Siskos, E., & Tsotsolas, N. (2015). Elicitation of criteria importance weights through the simos method: A robustness concern. *European Journal of Operational Research*, *246*, 543–553. https://doi.org/10.1016/j.ejor.2015.04.037

Siskos, Y., Grigoroudis, E., Zopounidis, C., & Saurais, O. (1998). Measuring customer satisfaction using a collective preference disaggregation model. *Journal of Global Optimization*, *12*(2), 175–195. https://doi.org/10.1023/A:1008262411587

Smailović, J., Grčar, M., Lavrač, N., & Žnidaršič, M. Predictive sentiment analysis of tweets: A stock market application (A. Holzinger & G. Pasi, Eds.). In: *Human-computer interaction and knowledge discovery in complex, unstructured, big data* (A. Holzinger & G. Pasi, Eds.). Ed. by Holzinger, A., & Pasi, G. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, 77–88. ISBN: 978-3-642-39146-0. https://doi.org/10.1007/978-3-642-39146-0_8.

Smilkstein, G. (1978). The family APGAR: a proposal for a family function test and its use by physicians. *The Journal of Family Practice*, *6*(6), 1231–9. https://psycnet.apa.org/record/1979-26481-001

Solymosi, T., & Dombi, J. (1986). A method for determining the weights of criteria: The centralized weights [Second EURO Summer Institute]. *European Journal of Operational Research*, *26*(1), 35–41. https://doi.org/https://doi.org/10.1016/0377-2217(86)90157-8

Sproull, R. F. (1991). Refinements to nearest-neighbor searching ink-dimensional trees. *Algorithmica*, *6*(1), 579–589. https://doi.org/10.1007/BF01759061

Srinivasan, V., & Shocker, A. D. (1973). Linear programming techniques for multidimensional analysis of preferences. *Psychometrika*, *38*, 337–369. https://doi.org/10.1007/BF02291658

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958. https://dl.acm.org/doi/abs/10.5555/2627435.2670313

Stigler, S. M. (1981). Gauss and the Invention of Least Squares. *The Annals of Statistics*, *9*(3), 465 –474. https://doi.org/10.1214/aos/1176345451

Sujatha, P., & Mahalakshmi, K. Performance evaluation of supervised machine learning algorithms in prediction of heart disease. In: In *2020 ieee international conference for innovation in technology (inocon)*. 2020, 1–7. https://doi.org/10.1109/INOCON50539.2020.9298354.

Sun, A., Lim, E.-P., & Ng, W.-K. Web classification using support vector machine. In: In *Proceedings of the 4th international workshop on web information and data management*. WIDM '02. New York, NY, USA: Association for Computing Machinery, 2002, 96–99. ISBN: 1581135939. https://doi.org/10.1145/584931.584952.

Suresh, V., Janik, P., Rezmer, J., & Leonowicz, Z. (2020). Forecasting solar pv output using convolutional neural networks with a sliding window algorithm. *Energies*, *13*(3). https://doi.org/10.3390/en13030723

Suthaharan, S. (2016). Support vector machine. In *Machine learning models and algorithms for big data classification: Thinking with examples for effective learning* (pp. 207–235). Springer US. https://doi.org/10.1007/978-1-4899-7641-3_9

Swets, J. A. (1988). Measuring the accuracy of diagnostic systems. *Science*, *240*(4857), 1285–1293. https://doi.org/10.1126/science.3287615

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational Linguistic*, *37*, 267–307. http://direct.mit.edu/coli/article-pdf/37/2/267/1798865/coli_a_00049.pdf

Takeda, E., Cogger, K., & Yu, P. (1987). Estimating criterion weights using eigenvectors: A comparative study. *European Journal of Operational Research*, *29*(3), 360–369. https://doi.org/https://doi.org/10.1016/0377-2217(87)90249-9

Tamiz, M., Jones, D. F., & El-Darzi, E. (1995). A review of goal programming and its applications. *Annals of Operations Research*, *58*(1), 39–53. https://doi.org/10.1007/BF02032309

Tanaka, H. (1984). A formulation of fuzzy linear programming problem based on comparison of fuzzy numbers. *Control and cybernetics*, *13*, 185–194. https://www.semanticscholar.org/paper/FORMULATION-OF-FUZZY-LINEAR-PROGRAMMING-PROBLEM-ON-Tanaka-Ichihashi/eea8b4aef1bc060310115f693a82b10d57388521

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, *72*(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

Thai-Nghe, N., Busche, A., & Schmidt-Thieme, L. Improving academic performance prediction by dealing with class imbalance. In: In *2009 ninth international conference on intelligent systems design and applications*. 2009, 878–883. https://doi.org/10.1109/ISDA.2009.15.

Thuillier, E., Gamper, H., & Tashev, I. J. Spatial audio feature discovery with convolutional neural networks. In: In *2018 ieee international conference on acoustics, speech and signal processing (icassp)*. 2018, 6797–6801. https://doi.org/10.1109/ICASSP.2018.8462315.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267–288. Retrieved May 24, 2022, from http://www.jstor.org/stable/2346178

Tippmann, S. (2015). Programming tools: Adventures with R. *Nature*, *517*(7532), 109–110. https://doi.org/10.1038/517109a

Tong, H., & Lim, K. S. (1980). Threshold autoregression, limit cycles and cyclical data. *Journal of the Royal Statistical Society: Series B (Methodological)*, *42*(3), 245–268. https://doi.org/10.1111/j.2517-6161.1980.tb01126.x

Triantaphyllou, E. (2000). *Multi-Criteria Decision Making Methods: A Comparative Study* (Vol. 44). https://doi.org/10.1007/978-1-4757-3157-6

Triantaphyllou, E., Kovalerchuk, B., Mann, L., & Knapp, G. M. (1997). Determining the most important criteria in maintenance decision making. *Journal of Quality in Maintenance Engineering*, *3*(1), 16–28. https://doi.org/10.1108/13552519710161517

Tryon, R. C., & Bailey, D. E. (1971). Cluster analysis. *Psychological Medicine*, *1*(2), 184–184. https://doi.org/10.1017/S0033291700000234

Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., & Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, *36*(10), 11994–12000. https://doi.org/10.1016/j.eswa.2009.05.029

Tweedie, M. C. An index which distinguishes between some important exponential families. In: In *Statistics: Applications and new directions: Proc. indian statistical institute golden jubilee international conference*. *579*. 1984, 579–604. https://www.academia.edu/29154095/Recent_Developments_in_Multilayer_Perceptron_Neural_Networks

Van Horn, K. S. (2003). Constructing a logic of plausible inference: A guide to cox's theorem. *International Journal of Approximate Reasoning*, *34*(1), 3–24. https://doi.org/10.1016/S0888-613X(03)00051-3

Van Rossum, G., & Drake Jr, F. L. (1995). *Python tutorial* (Vol. 620). Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands. https://bugs.python.org/file30394/tutorial.pdf

Vapnik, V. N. (1992). Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 831–838. http://papers.nips.cc/paper/506-principles-of-risk-minimization-for-learning-theory

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, *10*(5), 988–999. https://doi.org/10.1109/72.788640

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. https://doi.org/10.48550/ARXIV.1706.03762

Verma, U., Garg, C., Bhushan, M., Samant, P., Kumar, A., & Negi, A. Prediction of students' academic performance using machine learning techniques. In: In *2022 international mobile and embedded technology conference (mecon)*. 2022, 151–156. https://doi.org/10.1109/MECON53876.2022.9751956.

Vicens-Colom, J., Holles, J., & Liern, V. (2021). Measuring Sustainability with Unweighted TOPSIS: An Application to Sustainable Tourism in Spain. *Sustainability 2021, Vol. 13, Page 5283*, *13*(9), 5283. https://doi.org/10.3390/SU13095283

Vidal, R., Bruna, J., Giryes, R., & Soatto, S. (2017). Mathematics of deep learning. https://doi.org/10.48550/ARXIV.1712.04741

Vincke, P. (1992). Multicriteria decision-aid. *Mathematical Social Sciences*, *25*(2), 204–204. https://doi.org/10.1002/mcda.4020030208

Vinogradova, I., Podvezko, V., & Zavadskas, E. K. (2018). The recalculation of the weights of criteria in mcdm methods using the bayes approach. *Symmetry*, *10*(6). https://doi.org/10.3390/sym10060205

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

von Neumann, J. (1993). First draft of a report on the edvac. *IEEE Annals of the History of Computing*, *15*(4), 27–75. https://doi.org/10.1109/85.238389

Wang, A. An industrial strength audio search algorithm. In: In *Ismir. 2003*. Citeseer. 2003, 7–13. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.217.8882

Wang, F., & Sun, J. (2015). Survey on distance metric learning and dimensionality reduction in data mining. *Data Mining and Knowledge Discovery*, *29*(2), 534–564. https://doi.org/10.1007/s10618-014-0356-z

Wang, Y., Wang, S., & Lai, K. (2005). A new fuzzy support vector machine to evaluate credit risk. *IEEE Transactions on Fuzzy Systems*, *13*(6), 820–831. https://doi.org/10.1109/TFUZZ.2005.859320

Wang, Y.-J. (2014). A fuzzy multi-criteria decision-making model by associating technique for order preference by similarity to ideal solution with relative preference relation [New Sensing and Processing Technologies for Hand-based Biometrics Authentication]. *Information Sciences*, *268*, 169–184. https://doi.org/10.1016/j.ins.2014.01.029

Wasserman, P., & Schwartz, T. (1988). Neural networks. ii. what are they and why is everybody so interested in them now? *IEEE Expert*, *3*(1), 10–15. https://doi.org/10.1109/64.2091

Wes McKinney. Data Structures for Statistical Computing in Python (Stéfan van der Walt & Jarrod Millman, Eds.). In: *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt & Jarrod Millman, Eds.). Ed. by Stéfan van der Walt & Jarrod Millman. 2010, 56 –61. https://doi.org/10.25080/Majora-92bf1922-00a.

Wickham, H. (2016). *Ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., . . . Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Widodo, A., & Yang, B.-S. (2007). Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing*, *21*(6), 2560–2574. https://doi.org/10.1016/j.ymssp.2006.12.007

Williams, G., Baxter, R., He, H., Hawkins, S., & Gu, L. (2002). A comparative study of rnn for outlier detection in data mining. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 709–712. https://doi.org/10.1109/icdm.2002.1184035

Wilson, E. B. (1923). First and second laws of error. *Journal of the American Statistical Association*, *18*(143), 841–851. https://doi.org/10.1080/01621459.1923.10502116

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., . . . Rush, A. Transformers: State-of-the-art natural language processing. In: In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations*. Online: Association for Computational Linguistics, 2020, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6.

Wolters, W., & Mareschal, B. (1995). Novel types of sensitivity analysis for additive mcdm methods. *European Journal of Operational Research*, *81*(2), 281–290. https://doi.org/https://doi.org/10.1016/0377-2217(93)E0343-V

Wrigley, E. A. (2018). Reconsidering the Industrial Revolution: England and Wales. *The Journal of Interdisciplinary History*, *49*(1), 9–42. https://doi.org/10.1162/jinh_a_01230

Wu, J. (2017). Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, *5*(23), 495. https://cs.nju.edu.cn/wujx/paper/CNN.pdf

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd) [ISBN 978-1498716963]. Chapman; Hall/CRC. https://yihui.org/knitr/

Xie, Y., Allaire, J. J., & Grolemund, G. (2018). *R markdown: The definitive guide*. Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown/

Yang, Y. (2007). Consistency of cross validation for comparing regression procedures. *The Annals of Statistics*, *35*(6), 2450 –2473. https://doi.org/10.1214/009053607000000514

Yetilmezsoy, K., Sihag, P., Kıyan, E., & Doran, B. (2021). A benchmark comparison and optimization of gaussian process regression, support vector machines, and m5p tree model in approximation of the lateral confinement coefficient for cfrp-wrapped rectangular/square rc columns. *Engineering Structures*, *246*, 113106. https://doi.org/10.1016/j.engstruct.2021.113106

Yim, O., & Ramdeen, K. T. (2015). Hierarchical cluster analysis: Comparison of three linkage measures and application to psychological data. *The Quantitative Methods for Psychology*, *11*(1), 8–21. https://doi.org/10.20982/tqmp.11.1.p008

Yoon, K., Hwang, K., Yoon, P., Hwang, C., SAGE. & Sage Publications, i. (1995). *Multiple attribute decision making: An introduction*. SAGE Publications. https://books.google.es/books?id=gIU5DQAAQBAJ

Youden, W. J. (1950). Index for rating diagnostic tests. *Cancer*, *3*(1), 32–35. https://doi.org/10.1002/1097-0142(1950)3:1<32::AID-CNCR2820030106>3.0.CO;2-3

Yuan, W., Liu, J., & Zhou, H.-B. An improved knn method and its application to tumor diagnosis. In: In *Proceedings of 2004 international conference on machine learning and cybernetics (ieee cat. no.04ex826)*. *5*. 2004, 2836–2841 vol.5. https://doi.org/10.1109/ICMLC.2004.1378515.

Yule, G. U. (1927). On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London*, *226*, 267–298. https://www.jstor.org/stable/91170

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353. https://doi.org/10.1016/S0019-9958(65)90241-X

Zadeh, L. A. (1996). Fuzzy sets. In *Fuzzy sets, fuzzy logic, and fuzzy systems* (pp. 394–432). https://doi.org/10.1142/9789814261302_0021

Zavadskas, E. K., & Podvezko, V. (2016). Integrated determination of objective criteria weights in mcdm. *International Journal of Information Technology and Decision Making*, *15*, 267–283. https://doi.org/10.1142/S0219622016500036

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. http://arxiv.org/abs/1212.5701

Zhang, G. (2000). Neural networks for classification: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *30*(4), 451–462. https://doi.org/10.1109/5326.897072

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, *14*, 35–62. https://doi.org/10.1016/S0169-2070(97)00044-7

Zhang, J.-S., & Xiao, X.-C. (2000). Predicting chaotic time series using recurrent neural network. *Chinese Physics Letters*, *17*, 88–90. https://doi.org/10.1088/0256-307x/17/2/004

Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*. https://doi.org/10.1002/widm.1253

Zhang, L., Zhou, W., & Jiao, L. (2004). Wavelet support vector machine. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *34*(1), 34–39. https://doi.org/10.1109/TSMCB.2003.811113

Zheng, B. (1993). An axiomatic characterization of the Watts poverty index. *Economics Letters*, *42*(1), 81–86. https://doi.org/10.1016/0165-1765(93)90177-E

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2015). Learning deep features for discriminative localization. https://doi.org/10.48550/ARXIV.1512.04150

Zhou, Z., Cheng, S., & Hua, B. (2000). Supply chain optimization of continuous process industries with sustainability considerations. *Computers & Chemical Engineering*, *24*(2), 1151–1158. https://doi.org/10.1016/S0098-1354(00)00496-8

Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, *23*(4), 550–560. https://doi.org/10.1145/279232.279236

Zhu, M., Shahnawaz, M., Tubaro, S., & Sarti, A. Hrtf personalization based on weighted sparse representation of anthropometric features. In: In *2017 international conference on 3d immersion (ic3d)*. 2017, 1–7. https://doi.org/10.1109/IC3D.2017.8251901.

Zieliński, S. K., Antoniuk, P., Lee, H., & Johnson, D. (2022). Automatic discrimination between front and back ensemble locations in hrtf-convolved binaural recordings of music. *EURASIP J. Audio Speech Music Process.*, *2022*(1). https://doi.org/10.1186/s13636-021-00235-2

Zimmermann, H.-J. (1978). Fuzzy programming and linear programming with several objective functions. *Fuzzy Sets and Systems*, *1*(1), 45–55. https://doi.org/10.1016/0165-0114(78)90031-3

Zimmermann, H.-J. (2001). *Fuzzy set theory — and its applications*. Springer Dordrecht. https://doi.org/10.1007/978-94-010-0646-0

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, *67*(2), 301–320. Retrieved May 24, 2022, from http://www.jstor.org/stable/3647580

A APPENDIX 1: MULTIPLE-CRITERIA DECISION MAKING COMPUTATIONAL IMPLEMENTATIONS

A.1 IMPLEMENTATION OF CLASSIC TOPSIS AND VIKOR METHODS

One of the main contribution of this thesis is the publication of our GitHub repositories: uwTOPSIS (López-García, 2021a) and uwVIKOR (López-García, 2021b). Both repositories contain the Python modules to implement both unweighted techniques in the Python programming language. As we discussed in § 3.4, there exists a generalization from the unweighted approach to the classic one, being this demonstrated in Propositions 3.4.2 and 3.4.3.

Script 1: Example of Python script for applying TOPSIS and uwTOPSIS over a dataset.

```python
# Required packages
import pandas as pd
import numpy as np
from uwTOPSIS.uwTOPSIS import *

# Import the dataset
data = pd.read_csv("dataset.csv")
M = data.shape[1]

# TOPSIS parameters
directions = np.repeat("max", M)
L = np.repeat(1/(M+1), M)
U = np.repeat(1/(M-1), M)
weights = np.repeat(1/M, M)
norm = "euclidean"
p = 2
alpha = 0.5
forceideal = False
epsilon = np.finfo(float).eps

# Compute TOPSIS methods
uwT = uwTOPSIS(x, directions, L, U, norm, p, weights, alpha, forceideal)
try:
    T = uwTOPSIS(data, directions, L = weights, U = weights + epsilon)
except:
    T = uwTOPSIS(data, directions, L = weights - epsilon, U = weights)
```

Script 2: Example of Python script for applying VIKOR and uwVIKOR over a dataset.

```python
# Required packages
import pandas as pd
import numpy as np
from uwVIKOR.uwVIKOR import *

```

```
# Import the dataset
data = pd.read_csv("dataset.csv")
M = data.shape[1]

# VIKOR parameters
directions = np.repeat("max", M)
L = np.repeat(1/(M+1), M)
U = np.repeat(1/(M-1), M)
weights = np.repeat(1/M, M)
v = 0.5
epsilon = np.finfo(float).eps

# Compute TOPSIS methods
uwV = uwVIKOR(x, directions, L, U, v, forceideal)
try:
    V = uwVIKOR(data, directions, L = weights, U = weights + epsilon)
except:
    V = uwVIKOR(data, directions, L = weights - epsilon, U = weights)
```

A.2 SCRIPT FOR AN ACADEMIC PERFORMANCE INDICATOR USING FLEXIBLE MULTI-CRITERIA METHODS

This subsection is particularly dedicated to the Python implementation of uwTOPSIS to obtain the results for the students of Universidad Industrial de Santander.

Script 3: Implementation of the uwTOPSIS method for the entire SEA-UIS dataset.

```
import pandas as pd
import numpy as np
from uwTOPSIS.uwTOPSIS import *

# Definition of the eta normalization
def eta(x, A, a, b, B, k1, k2):
    def f(x, v1, v2, k):
        return (1-np.exp(k*(x-v2)/(v1-v2)))/(1-np.exp(k))
    if A <= x and x < a:
        z = f(x, a, A, k1)
    elif a < x and x < b:
        z = 1
    elif b < x and x <= B:
        z = f(-x, -b, -B, k2)
    else:
        z = 0
    return z

# Definition of the xi normalization
def xi(x, A, a, b, B):
    def f(x, v1, v2):
        return (x-v2)/(v1-v2)
    if A <= x and x < a:
        z = f(x, a, A)
    elif a < x and x < b:
        z = 1
    elif b < x and x <= B:
        z = f(-x, -b, -B)
    else:
        z = 0
    return z

# Data preparation
path_to_data = 'path/to/data.xlsx'
data = pd.read_excel(path_to_data)
directions = np.repeat("max", 5)
L = np.repeat(0.1, 5)
U = np.repeat(0.5, 5)
norm = "none"
p = 2

# Data normalization
data['Academic'] = data['Academic'].apply(lambda x: eta(x,1,6,7,7,1,0))
data['Cognitive'] = data['Cognitive'].apply(lambda x: eta(x,1,6,7,7,-1,0))
data['Economic'] = data['Economic'].apply(lambda x: xi(x,0,0.8,1,1))
data['Health'] = data['Health'].apply(lambda x: xi(x,0,0.65,0.65,0.65))
data['Social'] = data['Social'].apply(lambda x: xi(x,0.1,0.7,1,1))

# Application of unweighted TOPSIS method
x = uwTOPSIS(data, directions, L, U, norm, p, forceideal=True)
```

A.3 SCRIPT FOR UWVIKOR: AN UNWEIGHTED MULTI-CRITERIA DECISION MAKING APPROACH FOR COMPROMISE SOLUTION

Similarly to the uwTOPSIS algorithm, the implementation of uwVIKOR is coded with the same libraries and follows the same guidelines. Then, this section is just focused to show the minimal code to compute all the experiments. First, we describe how to conduct the data analysis and how to perform the unweighted algorithm.

Script 4: Data preprocessing to clean and prepare the data.

```
# Required packages
import os
import glob
import pandas as pd
import numpy as np

# Set the working directory
path_to_files = os.path.join('MVP_nominees', '*.csv')
# Get just CSV format files
files = glob.glob(path_to_files)
# List the attributes to change as float
attributes = ['G','MP','PTS',
              'TRB','AST','STL',
              'BLK','FG%','3P%',
              'FT%','WS','WS/48']

NBA_dataset = []
for year in files:
    data = pd.read_csv(year)
    # Drop useless columns
    data.drop(['Unnamed: 0', 'X'], axis=1, inplace=True)
    data.columns = data.loc[0]
    # Drop useless rows
    data.drop(0, axis=0, inplace=True)
    # Modify format of the attributes
    data[attributes] = data[attributes].astype(float)
    # Store the data
    NBA_dataset.append(data)
```

Script 5: Implementation of the uwVIKOR algorithm for the entire dataset.

```
# Import uwVIKOR library
from uwVIKOR.uwVIKOR import *

# Set parameters
J = NBA_dataset[0].shape[1]
directions = np.repeat('max', J)
L  = np.repeat(1/(J+1), J)
U  = np.repeat(1/(J-1), J)
v = 0.5

# Compute uwVIKOR per each year
DF = []
for data in NBA_dataset:
    x = uwVIKOR(data, directions, L, U, v)
    DF.append(x)
```

Once we have executed the code showed before, now we are able to carry out the additional programs that allow us to analyze the whole execution and the outcome. The following scripts contain a basic code instance that return the ranking positions, the uwVIKOR-efficiency, and the evaluation of weights.

Script 6: Ranking output for the three different measures $p \in \{0, 1, 2\}$.

```
RANKING = []
for i in range(N):
  # Get Q_L and Q_R ranks
  QL = DF[i][0].Q_Min
  QR = DF[i][0].Q_Max
  # Compute our three means
  AM = 0.5*(QL+QR)
  QM = (0.5*(QL**2+QR**2))**0.5
  GM = (QL*QR)**0.5
  # Transform ranking to cardinal order
  AM_rank = np.concatenate([np.argsort(AM), np.repeat([np.nan], 18-len(AM))])
  QM_rank = np.concatenate([np.argsort(QM), np.repeat([np.nan], 18-len(QM))])
  GM_rank = np.concatenate([np.argsort(GM), np.repeat([np.nan], 18-len(GM))])
  RANKING.append([AM_rank, QM_rank, GM_rank])
  # Print MVP with scores
  print('Year: {} | MVP {:22s} | Scores: {:.3f};{:.3f};{:.3f}'.format(
        2001+i, DF[i][0].Player[0], GM[0], AM[0], QM[0]))
  # Print its cardinality
  n_times_best = int(GM[0]==GM.min()) + int(AM[0]==AM.min()) + int(QM[0]==QM.min())
  print('{}| Best {:7.0f}{:7.0f}{:7.0f} | Higher? {};{};{} ({})\n'.format(
        11*' ',
        GM_rank[0], AM_rank[0], QM_rank[0],
        GM[0]==GM.min(), AM[0]==AM.min(), QM[0]==QM.min(),
        n_times_best))
```

Script 7: Tables of the uwVIKOR-efficiency broken down by year.

```
N = len(DF)
```

```
# Count the efficient and non-efficient elements
eff  = [(DF[i][0].Q_Min == 0).sum() for i in range(N)]
Neff = [(DF[i][0].Q_Max == 1).sum() for i in range(N)]
# Compute them as percentage
eff_p  = [(DF[i][0].Q_Min == 0).sum()/len(DF[i][0].Q_Min) for i in range(N)]
Neff_p = [(DF[i][0].Q_Max == 1).sum()/len(DF[i][0].Q_Max)  for i in range(N)]
# Present it as table
df_efficiency =pd.DataFrame([eff, eff_p, Neff, Neff_p],
                            index = ['Total', 'Percentage', 'Total', 'Percentage',],
                            columns=[2001 + i for i in range(20)])
print(df_efficiency.T.to_latex(float_format="%.4f"))
```

Script 8: Evaluations of weights via trapezoidal fuzzy functions.

```
# Define the trapezoidal fuzzy evaluation function
low = 0.1
up = 0.4
def merit_function(X, l=low, u=up):
  # Calculate the two superior breaks of the trapezoid
  stop1 = l + (u-l)/3
  stop2 = l + 2*(u-l)/3
  merit_vector = []
  for x in X:
    # Outside the trapezoid
    if x <= l or x >= u:
      merit = 0
    # Left spread
    elif x > l and x <= stop1:
      merit = (x - l)/(stop1 - l)
    # Fuzzy core
    elif x >= stop1 and x <= stop2:
      merit = 1
    # Right spread
    elif x >= stop2 and x < u:
      merit = (x - stop2)/(u -stop2)
    merit_vector.append(merit)
  return merit_vector

# Compute the evaluation of weights
merit_DF = []
merit_metrics = []
for i in range(N):
    merit_min = DF[i][1].apply(merit_function, axis=0) # min
    merit_max = DF[i][2].apply(merit_function, axis=0) # MAX
    merit_DF.append([merit_min,
                     merit_max])
    merit_metrics.append([merit_min.mean(axis=1),
                          merit_max.mean(axis=1)])
```

B APPENDIX 2: ARTIFICIAL INTELLIGENCE COMPUTATIONAL IMPLEMENTATIONS

B.1 SCRIPT FOR A PROPOSAL TO COMPARE THE SIMILARITY BETWEEN MUSICAL PRODUCTS. ONE MORE STEP FOR AUTOMATED PLAGIARISM DETECTION?

In our GitHub repository López-García (2022b), we designed an audio fingerprinting algorithm by means of two different manners: from raw audio (Script 9) or from a given spectrogram (Script 10).

Script 9: An example to apply SpectroMap over a raw signal.

```
import numpy as np
from spectromap.functions.spectromap import spectromap

y = np.random.rand(44100)
kwargs = {'fs': 22050, 'nfft': 512, 'noverlap':64}

# Instantiate the SpectroMap object
SMap = spectromap(y, **kwargs)

# Get the spectrogram representation plus its time and frequency bands
f, t, S = SMap.get_spectrogram()

# Extract the topological prominent elements from the spectrogram.
# Get (time, freq) coordinates of the peaks and the matrix with just these peaks.
fraction = 0.15 # Fraction of spectrogram to compute local comparisons
condition = 2   # Axis to analyze (0: Time, 1: Frequency, 2: Time+Frequency)
id_peaks, peaks = SMap.peak_matrix(fraction, condition)

# Get the peaks coordinates as as (s, Hz, dB)-array.
extraction_t_f_dB = SMap.from_peaks_to_array()
```

Script 10: An example to apply SpectroMap over a computed spectrogram.

```
from spectromap.functions.spectromap import peak_search

fraction = 0.05 # Fraction of spectrogram to compute local comparisons
condition = 2   # Axis to analyze (0: Time, 1: Frequency, 2: Time+Frequency)
id_peaks, peaks = peak_search(spectrogram, fraction, condition)
```

B.2 SCRIPT FOR MULTIVARIATE TIME SERIES PREDICTION BASED ON STOCK MARKET AND SENTIMENT ANALYSIS REGRESSORS

In § 6.2.7, we already mentioned the detailed description of the library versions. Then, this section is exclusively dedicated to showing the Python implementation for this paper. It is easy to realize that the procedure has to be automatized in order to ease the computational complexity of designing and training eight different RNNs. Then, we developed a bash script that sequentially executes the Script 11. The Python code needed to compute every single task for the achievement of results is written in the Script 12.

Script 11: Bash code to automate the execution of the models in a computer with Ubuntu 20.04 Focal Fossa as OS.

```
#!/bin/sh
printf "\n [-] Environment\n\n"
printenv
printf "\n [-] Host\n\n"
hostname
printf "\n [-] Current directory\n\n"
pwd
printf "\n [-] User\n\n"
whoami
printf "\n [-] Space Left\n\n"
df -h
printf "\n [-] Conda environment activation\n\n"
conda activate tf_LSTM
conda info
printf "\n [-] NVIDIA info\n\n"
nvidia-smi
printf "\n [-] Python execution\n\n"
python3 az_stateful_model.py -r all
python3 az_stateful_model.py -r none
python3 az_stateful_model.py -r prices
python3 az_stateful_model.py -r sentiment
python3 az_stateful_model.py -r all -m RNN
python3 az_stateful_model.py -r none -m RNN
python3 az_stateful_model.py -r prices -m RNN
python3 az_stateful_model.py -r sentiment -m RNN
```

Script 12: Script to perform the training of RNN models for predicting the stock prices of AstraZeneca.

```
# Package importation
# Basic packages
import os
import argparse
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
# TF library
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM, SimpleRNN
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
# Sklearn library
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import median_absolute_error, r2_score, accuracy_score

# Sentiment
```

```python
AZ_sent = pd.read_csv(os.path.join('Datasets', 'AstraZenca_sentiment.csv'))
AZ_sent.drop(['Unnamed: 0'], axis=1, inplace=True)
# Price
AZ_price = pd.read_csv(os.path.join('Datasets', 'AstraZenca_prices.csv'))
date_price = pd.to_datetime(AZ_price['Exchange Date'])
AZ_price.drop(['Unnamed: 0', 'Exchange Date'], axis=1, inplace=True)

# Parser object
parser = argparse.ArgumentParser(description = 'Model generation and training\nTwo needed parameters: Model and Regressor')
parser.add_argument('-m', '--model',
                    help = 'Indicate whether the model is "LSTM" or "RNN".',
                    default="LSTM")
parser.add_argument('-r', '--regressor',
                    help = 'Indicate if the regressor is "none", "sentiment", "prices" or "all".',
                    default="sentiment")
args = parser.parse_args()

units = 2**7
model_type = args.model
regressor = args.regressor

if regressor == 'sentiment':
    df = AZ_sent
    df.drop(df.index[-1], axis=0, inplace=True)
elif regressor == 'prices':
    df = AZ_price
elif regressor == 'none':
    df = AZ_price.loc[:, ['Close']]
else:
    AZ_price.drop(['Close'], axis=1, inplace=True)
    df = pd.concat([AZ_sent,AZ_price], axis=1)
    df.drop(df.index[-1], axis=0, inplace=True)

# Data standarization
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)

# Preprocess and prepare train/validation sets
def data_to_sequences(data, steps_back = 5, steps_forward = 1, start = 'end'):
    '''
    Prepare the temporal sequences of our data
    INPUT:
        data: Data Frame. Contains the predictor and the regressor
        steps_back: Integer. Number of needed elements to make a prediction
        steps_forward: Integer. Number of predictions for a given output
        start: Either "beginning" or "end" to indicate at which point the loop sequence starts
    OUTPUT:
        x: Array with the X-dataset sequenced [(N-steps_forward)/steps_back, steps_back]-shaped
        y: Array with the Y-dataset sequenced [(N-steps_forward)/steps_back, steps_forward]-shaped
    '''
    # Split data into the n of steps back as input(X) and get the following steps forward as output (Y)
    x = []
    y = []
    N = data.shape[0]
    if start == 'beginning':
        for i in range(steps_back, N-steps_forward+1):
            x.append(data[i-steps_back : i, 0:data.shape[1]])
            y.append(data[i+steps_forward-1 : i+steps_forward, 0])
    elif start == 'end':
        for i in range(N-steps_back-steps_forward+1):
            x.append(data[N-i-steps_back-steps_forward : N-i-steps_forward, 0:data.
                shape[1]])
            y.append(data[N-i-steps_forward : N-i-steps_forward+1, 0])
        # Since we started from the last value, now we rearrange from N to 0.
        y = [y[i] for i in range(len(x)-1, -1, -1)]
        x = [x[i] for i in range(len(x)-1, -1, -1)]
    return np.array(x), np.array(y)

# Sequentalization of the data
from_2021 = 108
batch_size = 4
X, y = data_to_sequences(df_scaled)
X_train, Y_train = X[:-from_2021], y[:-from_2021]
X_val, Y_val = X[-from_2021:], y[-from_2021:]

print('{}'.format(20*'-'))
print('Training sets\nX-shape: {}\nY-shape: {}\n{}'.format(X_train.shape, Y_train.
    shape, 20*'-'))
print('Validation sets\nX-shape: {}\nY-shape: {}\n{}'.format(X_val.shape, Y_val.shape
    , 20*'-'))
print('Data-shape: {}\n{}'.format(df.shape, 20*'-'))

# Fitting
Stop = EarlyStopping(mode = 'min', patience = 30, verbose=1, min_delta = 1e-4)
LRed = ReduceLROnPlateau(factor=1/2, patience=10, verbose=1, min_lr=1e-6)

def RNNs(model_type, layer, units, shape, regressor):
    '''
    Model generation of LSTM-RNN.

    # Parameters
        model_type: Type of model, either LSTM or RNN.
        layer: Type of layer arch, either bidirectional, simple or stacked.
        units: Number of neurons for the recursive blocks.
        shape: Batch input shape for stateful RNNs.
        regressor: Name of the regressors.

    # Return
        model: Recursive Keras model.
    '''
    # Parameters
    to_drop = 1e-2*shape[-1]
    reg = l2(l2 = 1e-4*shape[-1])
    # 1st: Select the recursive model
    if model_type == 'LSTM':
        RNN = LSTM
    elif model_type == 'RNN':
        RNN = SimpleRNN
    # 2nd: Select the layer architecture
    model = Sequential()
    if layer == 'bidirectional':
        model.add(Bidirectional(RNN(units,
                                    batch_input_shape=shape,
                                    stateful=True,
                                    dropout=to_drop,
                                    recurrent_regularizer=reg),
                                    name='Bidirectional'))
    elif layer == 'simple':
        model.add(RNN(units,
                      batch_input_shape=shape,
                      stateful=True,
                      dropout=to_drop,
                      recurrent_regularizer=reg,
                      name='Simple'))
    elif layer == 'stacked':
        model.add(RNN(units,
                      batch_input_shape=shape,
                      stateful=True,
                      dropout=to_drop,
                      return_sequences=True,
                      name='Stack0'))
        model.add(RNN(units,
                      batch_input_shape=shape,
                      stateful=True,
                      recurrent_regularizer=reg,
                      name='Stack1'))
    # 3rd: Fully connected layer
    model.add(Dense(units, name = 'FC'))
    model.add(Dense(1, name = 'Dense_Out'))
    # 5th: Compile
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model

# Model generation
inp_shape = [batch_size, X_train.shape[1], X_train.shape[2]]
model_list = []
for arch in ['bidirectional', 'simple', 'stacked']:
    model = RNNs(model_type = model_type,
                 layer = arch,
                 units = units,
                 shape = inp_shape,
                 regressor = regressor)
    model._name = '{}_{}_{}'.format(arch, model_type, units)
    model_list.append(model)

history_list = []
model_arch = ['bidirectional', 'simple', 'stacked']
for i, model in enumerate(model_list):
    print('\nFitting {}\n{}'.format(model._name, 22*'-'))
    history = model.fit(X_train, Y_train,
                        validation_data = (X_val, Y_val),
                        epochs=350,
                        batch_size=batch_size,
                        verbose=2,
                        shuffle=False,
                        callbacks=[Stop, LRed],
                        )
    model.reset_states()
    model.save('{}_{}.hdf5'.format(regressor, model._name))
    history_list.append(history)
# Plot train-validation loss curves
fig, axs = plt.subplots(1, len(history_list), figsize=(8,4), sharey=True)
for i, history in enumerate(history_list):
    axs[i].plot(history.history['loss'], label ='Loss')
    axs[i].plot(history.history['val_loss'], label = 'Val-loss')
    axs[i].set(xlabel='Epochs')
    axs[i].set_title(model_list[i]._name)
    axs[i].legend()
plt.yscale('log')
plt.tight_layout()
plt.savefig('train_curves_{}_{}.pdf'.format(model_type, regressor))

# Scores and predictions
predictions = []
for model in model_list:
    pred = model.predict(X, batch_size=batch_size)
    pred = scaler.inverse_transform(pred.repeat(X.shape[2], axis=-1))[:,0]
    model.reset_states()
    predictions.append(pred)

# Evaluation
y_real = scaler.inverse_transform(y.repeat(X.shape[2], axis=-1))[:,0]

measures = []
for preds in predictions:
    # Train
    y_train = y_real[:-from_2021]
    pred_train = preds[:-from_2021]
    MAE_train = np.mean(np.abs(y_train - pred_train))
    MAPE_train = 100*np.mean(np.abs((y_train-pred_train) / y_train))
    R2_train = r2_score(y_train,pred_train)
    # Validation
    y_val = y_real[-from_2021:]
    pred_val = preds[-from_2021:]
    MAE_val = np.mean(np.abs(y_val - pred_val))
    MAPE_val = 100*np.mean(np.abs((y_val-pred_val) / y_val))
    R2_val = r2_score(y_val, pred_val)
    # Results
    dic = {'MAE':[MAE_train, MAE_val],
           'WMAPE': [MAPE_train, MAPE_val],
           'R2': [R2_train, R2_val],
            }
    measures.append(dic)

# Store evaluations as CSV
eval_list = []
for m in measures:
    ev = pd.DataFrame(m, index = ['Train', 'Val'])
    eval_list.append(ev)
    ev.round(4).to_csv('evaluations_{}_{}.csv'.format(model_list[i]._name, regressor)
        )
pd.concat(eval_list).to_csv('evaluations_{}_{}.csv'.format(regressor, units))

# Generate the data splits per architecture
dates_prediction = date_price[5:]
DATE = [dates_prediction,
        dates_prediction[:-from_2021],
        dates_prediction[-from_2021:]]
Y_REAL = [y_real, y_train, y_val]
PREDICTION = [[p, p[:-from_2021], p[-from_2021:]] for p in predictions]
MODEL_ARCH = ['bidirectional', 'simple', 'stacked']

# Plot predictions
_lw = 0.9
fig, axs = plt.subplots(3, 3, figsize=(20,8))
for k, pred in enumerate(PREDICTION):
    for i in range(3):
        axs[k, i].plot(DATE[i], Y_REAL[i], label = 'Real', lw=_lw)
        axs[k, i].plot(DATE[i], pred[i], label = 'Prediction', lw=_lw)
        if i == 0:
            axs[k, i].axvline(pd.to_datetime('2021-1-1'),
                              linestyle='--',
                              linewidth=_lw)
        if i == 1:
            axs[k, i].legend()
            axs[k, i].set_title(model_list[k]._name)
        if i != 2:
            axs[k, i].xaxis.set_major_locator(mdates.MonthLocator(bymonth=(1, 7)))
        else:
            axs[k, i].xaxis.set_major_locator(mdates.MonthLocator(bymonth=(1, 2, 3,
                4, 5, 6)))
plt.tight_layout()
plt.savefig('predictions_{}_{}.pdf'.format(model_type, regressor))
```

B.3 SCRIPT FOR ON THE APPLICATION OF EXPLAINABLE ARTIFICIAL INTELLIGENCE TECHNIQUES ON HRTF DATA

This section outlines the Python code needed to perform some parts of the experimental case. It is noteworthy that the whole script is quite long to be attached here, so we decide to cover just the important steps that concern the data procedure and model implementation.

Script 13: Data preparation and preprocess required from the import to pre-fitting phase.

```
# Data import
import numpy as np
npzfile = np.load('cipic.npz')

# Break down each component of the numpy file
sr = 44100
X_hrir = npzfile['arr_0'].astype('float')/(2**15)   # HRIR data normalized
subj_indx = list(npzfile['arr_1'])                   # Subjects
az_indx = list(npzfile['arr_2'])                     # Azimuth angles
elev_indx = list(npzfile['arr_3'])                   # Elevation angles

# Reshape the data to merge all the responses for all the subjects
X_hrir_s = np.reshape(X_hrir, (200, 45, -1, 2))
# Get elevation an azimuth indexes
y_elevation = np.asarray(list(range(50)) * 25)
y_azimuth = np.repeat(list(range(25)), 50)
y = np.vstack([y_azimuth, y_elevation])

# Defining directional sectors for classification
def get_direction_class(elev, azi):
  if elev >= -90 and elev <= -20 and azi >= -60 and azi <= +60:
    cl = ’front_down’
  if elev > -20 and elev <= +20 and azi >= -60 and azi <= +60:
    cl = ’front_level’
  if elev > +20 and elev <= +70 and azi >= -60 and azi <= +60:
    cl = ’front_up’
  if elev > +70 and elev <= +110 and azi >= -60 and azi <= +60:
    cl = ’up’
  if elev > +110 and elev <= +160 and azi >= -60 and azi <= +60:
    cl = ’back_up’
  if elev > +160 and elev <= +200 and azi >= -60 and azi <= +60:
    cl = ’back_level’
  if elev > +200 and elev <= +270 and azi >= -60 and azi <= +60:
    cl = ’back_down’
  if np.abs(azi)>60 and elev >= 0 and elev <= 180:
    cl = ’lateral_up’
  if np.abs(azi)>60 and (elev < 0 or elev >= 180):
    cl = ’lateral_down’
  return cl
# Disaggregate spatial components into classes
y_class = [get_direction_class(elev_indx[k[1]], az_indx[k[0]]) for k in y.T]
class_dict = {’front_down’: 0, ’front_level’: 1, ’front_up’: 2,
              ’up’: 3, ’back_up’: 4, ’back_level’: 5,
              ’back_down’: 6, ’lateral_up’: 7, ’lateral_down’: 8 }
y_class = [class_dict[k] for k in y_class]

# HRTF transformation
X = np.reshape(X_hrir_s, (200,-1,2))
X = np.swapaxes(X, 0, 1)
# Compute FFT
NFFT = 512
X_l = X[:, :, 0]
X_r = X[:, :, 1]
Xf_l = np.fft.rfft(X_l, NFFT, axis=1)
Xf_r = np.fft.rfft(X_r, NFFT, axis=1)
Xf = np.stack([Xf_l, Xf_r], axis=2)
Xf = np.abs(Xf)
Nbins = Xf.shape[1]

# Track original angles
y_sj = np.repeat(list(range(45)), 25*50)
y_az = np.asarray(list(y_azimuth)*45)
y_el = np.asarray(list(range(50))*45*25)
y = np.stack([y_sj, y_az, y_el], axis=1)
yc = np.asarray(y_class*45)

# Change channels
y_az_ang = np.asarray([az_indx[k] for k in y_az])
y_contra = np.nonzero(y_az_ang > 0)[0]
Xf2 = np.copy(Xf)
Xf2[y_contra, :, 0] = Xf[y_contra, :, 1]
Xf2[y_contra, :, 1] = Xf[y_contra, :, 0]

# Mel-scale frequency warping
def Hz_to_Mel(f_hz):
  return (2595 * np.log10(1 + f_hz/700))

def Mel_to_Hz(f_mel):
  return (700* (10**(f_mel/2595)-1))
# Prepare transformation
f_min = 0
f_max = sr/2
f_min_mel = Hz_to_Mel(f_min)
f_max_mel = Hz_to_Mel(f_max)
f_mels = np.linspace(f_min_mel, f_max_mel, Nbins)
# Transform
f_Hz = Mel_to_Hz(f_mels)
mel_bins = np.round(f_Hz/(sr/Nbins)).astype(int)
Xm = np.log10(np.abs(Xf2[:, mel_bins, :]) + np.finfo(float).eps)

# Train-Validation splits
X_train, y_train = Xm[0:36*1250, :], yc[0:36*1250]
X_val, y_val = Xm[36*1250:, :], yc[36*1250:]
```

Script 14: Implementation of the 1D-CNN architecture for input of 2 channels as described in Table 6.12.

```
# TensorFlow import
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping,
    ReduceLROnPlateau
from pathlib import Path

# Basic parameters
CONV_KERNEL = 16
CONV_FILTERS = 16
POOL_KERNEL = 2

N_classes = len(np.unique(yc))
model = models.Sequential(name='CIPIC_2Channels')
# 1st block
model.add(layers.Conv1D(CONV_FILTERS*4, CONV_KERNEL,
                        activation='relu', padding='same',
                        input_shape=X_train.shape[1:], name='Conv1'))
model.add(layers.MaxPooling1D(POOL_KERNEL, name='MaxP1'))
# 2nd block
model.add(layers.Conv1D(CONV_FILTERS*2, CONV_KERNEL,
                        activation='relu', padding='same', name='Conv2'))
model.add(layers.MaxPooling1D(POOL_KERNEL, name='MaxP2'))
# 3rd block
model.add(layers.Conv1D(CONV_FILTERS*2, int(CONV_KERNEL/2),
                        activation='relu', padding='same', name='Conv3'))
model.add(layers.MaxPooling1D(POOL_KERNEL, name='MaxP3'))
# 4th block
model.add(layers.Conv1D(CONV_FILTERS, int(CONV_KERNEL/2),
                        activation='relu', padding='same', name='Conv4'))
# Output
model.add(layers.GlobalAveragePooling1D(name='GlobPool'))
model.add(layers.Dense(N_classes, activation='softmax', name='Prediction'))
model.summary()

# Create callbacks
checkpoint_name = 'HRTFModel'
checkpoint_dir  = './checkpoints'
checkpoint_path = str(Path(checkpoint_dir)/(checkpoint_name+".hdf5"))
```

```
logs_dir  = './logs'
logs_path = str(Path(logs_dir)/'{}.csv'.format(checkpoint_name))

# Create folders if they don't exist
Path(checkpoint_dir).mkdir(parents=True, exist_ok=True)
Path(logs_dir).mkdir(parents=True, exist_ok=True)

# Create the callback list
callbacks = []
# Save best checkpoints
callbacks.append(
    ModelCheckpoint(filepath = checkpoint_path,
                    monitor = 'val_loss',
                    mode = 'min',
                    save_best_only = True,
                    save_weights_only = False,
                    verbose = True)

)
# Save training history in csv
callbacks.append(
    CSVLogger(filename = logs_path, append = False)
)
# Early stopping
callbacks.append(
    EarlyStopping(monitor = 'val_loss',
                 patience = 25,
                 verbose = True)
)
#Reduce learning rate
callbacks.append(
    ReduceLROnPlateau(monitor = 'val_loss',
                      factor = 0.5,
                      patience = 15,
                      verbose = True)
)
# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
# Fit the model
history = model.fit(X_train,
                    y_train,
                    validation_data = (X_val, y_val),
                    callbacks = callbacks,
                    batch_size = 16,
                    epochs=1000)
```

C APPENDIX 3: MULTIPLE-CRITERIA DECISION MAKING & ARTIFICIAL INTELLIGENCE COMPUTATIONAL IMPLEMENTATIONS

C.1 SCRIPT FOR EARLY DETECTION OF STUDENTS' FAILURE USING MACHINE LEARNING TECHNIQUES

The entire development for this case study has been implemented in Google Colab. Then, the Python version utilized was 3.7.13 with Pandas 1.3.5 and Numpy 1.21.6. In this case, we have implemented machine learning algorithms and evaluations from the Sklearn 1.0.2 library with the XGBoost 0.90 package for the main tree-gradient algorithm used for classification. The ADASYN technique was applied with the imblearn 0.8.1 package.

The following Script 15, contains the Python process for training the dataset and obtaining the results shown in § 7.1.4.1. It is assumed that this script follows the code and results computed in the Script 3, and so it is known the dataset X.

Script 15: Implementation of the tree-based methods for the SEA-UIS dataset, assuming a continuation from Script 3.

```python
# Data packages
import pandas as pd
import numpy as np
# Classification models
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
# Additionals for ML
import imblearn
from collections import Counter
from sklearn import metrics
from sklearn.preprocessing import minmax_scale
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.model_selection import train_test_split, RepeatedStratifiedKFold

# Main default parameters
RANDOM = 13
N_ESTIMATORS = 500
LEARNING_RATE = 0.1
DEPTH = 10

# Extraction of success-failure labels
df = pd.read_excel('datosUIS.xlsx')
df['Pass'] = 0.5*(df['Calculus'] + df['Algebra']).to_numpy() < 2.5
y = df['Pass'].to_numpy().astype('int')

# Generate the train-test splits
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=RANDOM)
```

```

# Generate ADASYN-oversampling sets (so far we do not used them)
oversample = imblearn.over_sampling.ADASYN(random_state=RANDOM)
X_adasyn, y_adasyn = oversample.fit_resample(X, y)
X_adasyn_train, X_adasyn_test, y_adasyn_train, y_adasyn_test = train_test_split(
    X_adasyn_norm, y_adasyn)

# Function that jointly evaluates the model output
def get_scores(y_pred, y_real, y_prob):
    PRF1 = metrics.precision_recall_fscore_support(y_real, y_pred, average='binary')
    acc = metrics.accuracy_score(y_real, y_pred)
    auc = 1 - metrics.roc_auc_score(y_real, y_prob)
    return np.array([PRF1[0], PRF1[1], PRF1[2], acc, auc])

# Model definition
models = [XGBClassifier(max_depth = DEPTH,
                        n_estimators = N_ESTIMATORS,
                        learning_rate = LEARNING_RATE,
                        random_state = RANDOM),
          GradientBoostingClassifier(max_depth = DEPTH,
                                     n_estimators = N_ESTIMATORS,
                                     learning_rate = LEARNING_RATE,
                                     random_state = RANDOM),
          DecisionTreeClassifier(max_depth = DEPTH,
                                 random_state = RANDOM),
          RandomForestClassifier(max_depth = DEPTH,
                                 n_estimators = N_ESTIMATORS,
                                 random_state = RANDOM),
          ]
# Model names
model_names = ['XGBoost',
               'GBoost',
               'Decision_Tree',
               'Random_Forest',
               ]
# Definition of the vector outputs to store
feat_importance = []
predictions = []
scores = []
dec_matrix = []

# Loop to train each model
for i, model in enumerate(models):
    # Training
    model.fit(X_train, y_train)
    # Get feature weights
    feat_importance.append(model.feature_importances_)
    # Get decision and probability vectors
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test).T[0]
    predictions.append(y_pred)
    # Compute scores and decision matrices
    scores.append(get_scores(y_pred, y_test, y_prob))
    dec_matrix.append(metrics.confusion_matrix(y_test, y_pred))
```

Script 16: Implementation of the hyperparameter tuning for the XGBoost model.

```
# Create the computational grid
```

```
param_grid = {
    "max_depth": [i+5 for i in range(5)],
    "n_estimators": [200 + 10*i for i in range(11)],
    "gamma": [0.075, 0.1, 0.125, 0.15],
    "learning_rate": [0.1, 0.05, 0.02, 0.01],
    "reg_lambda": [0.2, 0.25, 0.3, 0.5],
    "random_state": [RANDOM],
}

# Initialize the XGBoost classifier
xgb_binary = XGBClassifier()

# Init Grid Search
grid_binary = GridSearchCV(xgb_binary,
                           param_grid,
                           n_jobs=-1,
                           cv=3,
                           scoring="roc_auc",
                           verbose=1)
# Compute the grid search
grid_binary.fit(X_over_train, y_over_train)

# Print the results
print('Tunned parameters')
for _key in grid_binary.best_params_:
    print('{:<14s}:{:7.2f}'.format(_key, grid_binary.best_params_[_key]))

# Print model
grid_binary.best_estimator_
```

C.2 SCRIPT FOR MULTIPLE-CRITERIA DECISION MAKING APPROACH FOR AN IN-DEPTH BENCHMARKING OF SUPERVISED MACHINE LEARNING MODELS

The implementation of the code presented in this subsection is entirely related to our GitHub repository López-García (2022a). Due to the different settings required per implementation, we have divided this section into two subsections. This first one concerns the topics related to the AstraZeneca regression task and the second one with the issues related to the CIPIC classification task.

C.2.1 *Case one: Forecast of the AstraZeneca close price with the use of stock market regressors*

Script 17: Implementation of the fitting stage for the XGB, LGB, CatB, RF, and GPR regressor models.

```
# Basics
import os
import time
from datetime import datetime
import numpy as np
import pandas as pd
from joblib import dump, load

# Remove warnings
import warnings
warnings.filterwarnings('ignore')
```

```python
warnings.simplefilter('ignore')

# Statistics
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import r2_score, mean_squared_error, median_absolute_error,
    mean_absolute_error
from sklearn.model_selection import TimeSeriesSplit

# Machine learning
# Gaussian process
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, DotProduct, WhiteKernel
from sklearn.gaussian_process.kernels import Matern, RationalQuadratic
# Regressors
from sklearn.ensemble import AdaBoostRegressor, RandomForestRegressor
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
from catboost import CatBoostRegressor
from prophet import Prophet

# Graphics
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
plt.rcParams['figure.figsize'] = [8, 2.5]
plt.rcParams['lines.linewidth'] = 0.75
plt.rcParams['lines.markersize'] = 1.5

# Constants of the problem
RANDOM = 13
N_ESTIMATORS = 1000
STOPPING = 20
TREE_VERBOSE = 1

# Preprocess and prepare train/validation sets
def data_to_sequences(data, steps_back = 30, steps_forward = 1, start = 'end'):
    '''
    Prepare the temporal sequences of our data

    INPUT:
        data: Data Frame. Contains the predictor and the regressor
        steps_back: Integer. Number of needed elements to make a prediction
        steps_forward: Integer. Number of predictions for a given output
        start: Either "beginning" or "end" to indicate at which point the loop
            sequence starts

    OUTPUT:
        x: Array with the X-dataset sequenced [(N-steps_forward)/steps_back,
            steps_back]-shaped
        y: Array with the Y-dataset sequenced [(N-steps_forward)/steps_back,
            steps_forward]-shaped
    '''
    # Split data into the number of steps back as input(X) and get the following
        steps forward as output (Y)
    x = []
    y = []
    for i in range(len(data)-steps_back-1):
        x.append(data[i : (i+steps_back), :])
        y.append(data[(i+steps_back), 0])
    return np.array(x), np.array(y)



# Evaluation of the predictions
def model_evaluation(y_real, y_predicted):
    '''
    Set of evaluation functions

    INPUT:
        y_real: Array with real values of the time series
        y_predicted: Array with the forecast values of a model

    OUTPUT:
        List with the scores: MSE, RMSE, MAE, MedAE, MAPE, WMAPE, Rsquare
    '''
    MSE     =  mean_squared_error(y_real, y_predicted)
    RMSE    = np.sqrt(MSE)
    MedAE   = median_absolute_error(y_real, y_predicted)
    MAPE    = np.sum(np.abs((y_real - y_predicted) / y_real))/len(y_real) * 100
    Rsquare = r2_score(y_real, y_predicted)

    return [RMSE, MedAE, MAPE, Rsquare]

metrics_names = ["RMSE", "MedAE", "MAPE", "R2"]

# Load AZ dataset
regressor_type = 'prices'
df = pd.read_csv(os.path.join('AstraZenca_{}.csv'.format(regressor_type)), index_col
    =0)
N = df.shape[0]

# We convert the data structures into float32
df.index = pd.to_datetime(df['Exchange Date'])
df.drop(df.columns[0], axis=1, inplace=True)
df = df.astype('float32')

# Scale the data for fitting
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)

# Sequentialize data series
STEPS_BACK = 30
X, Y = data_to_sequences(df_scaled, STEPS_BACK)

# Get sizes to split data
from_2021 = np.sum(df.index > '2021-01-01')

# Split the dataframe as train and test
X_train, Y_train = X[ : -from_2021, ], Y[ : -from_2021, ]
X_val, Y_val = X[-from_2021 : , ], Y[-from_2021 : , ]
print('Train shapes: {}, {}\nVal   shapes: {}, {}'.format(X_train.shape, Y_train.
    shape, X_val.shape, Y_val.shape))

# Data preparison for Tree-Based models: Concatenate all the regressors into a single
     vector
def regressor_flatten(data):
```

```python
    '''
    Per each component in data, it flattens the next components into a single vector.\n
    It returns a matrix with shape:\n
        (data.shape[0], data.shape[1] * ... * data.shape[N_regressors])
    '''
    data_flatten = []
    for i in range(data.shape[0]):
        data_flatten.append(data[i].flatten())
    return np.array(data_flatten)

X_flat = regressor_flatten(X)
X_train_flat = regressor_flatten(X_train)
X_val_flat = regressor_flatten(X_val)

print('New shapes for flat datasets')
print('Entire: {}\nTrain:  {}\nTest:   {}'.format(X_flat.shape,
                                                X_train_flat.shape,
                                                X_val_flat.shape))

y_real = np.expand_dims(Y, -1)
y_real = y_real.repeat(df.shape[1], axis=-1)
y_real = scaler.inverse_transform(y_real)[:,0]
y_real.shape

# Create new working directory
AZ_DIR = 'AZ_FINAL_EXPERIMENT'
os.chdir('./{}'.format(AZ_DIR))

N_SPLITS = 9
SERIES_GAP = 5
TEST_SIZE  = from_2021 - N_SPLITS * SERIES_GAP
TRAIN_SIZE = X.shape[0] - from_2021 - N_SPLITS * SERIES_GAP

def Time_Series_CV(timeseries, N_SPLITS = 9, GAP = 8):
    TRAIN_SIZE = 700
    TEST_SIZE  = 70
    train_test_splits = []
    for n in range(N_SPLITS):
        train_split = timeseries[n*GAP : n*GAP + TRAIN_SIZE]
        test_split  = timeseries[n*GAP + TRAIN_SIZE : n*GAP + TRAIN_SIZE + TEST_SIZE]
        train_test_splits.append([train_split, test_split])
    return train_test_splits

TB_model_names = [
                  'XGB',
                  'LGB',
                  'CatB',
                  'RF',
                  'GPR',
                  ]

TB_setting_times = []
TB_fitting_times = []
TB_saving_times = []
TB_predictions = []
TB_evaluations = []
TB_feat_importance = []
```

```
TB_indexes = []
# TB_eval_curves = []

# iteration = 1
# for train_index, test_index in TS_split.split(X, Y):
TS_split = Time_Series_CV(np.arange(X.shape[0]))
for iteration, [train_index, test_index] in enumerate(TS_split):
    print('({}) Cross-Validation number {}\n{}'.format(datetime.now().strftime('%H:%M
        :%S'), iteration, 70*'='))
    start_time = time.time()
    # Get sets from indexes
    X_train, X_val = X[train_index], X[test_index]
    Y_train, Y_val = Y[train_index], Y[test_index]
    TB_indexes.append([train_index, test_index])
    # Flat X-components
    X_flat = regressor_flatten(X)
    X_train_flat = regressor_flatten(X_train)
    X_val_flat = regressor_flatten(X_val)

    RANDOM = 13 * iteration
    TB_models = [
            XGBRegressor(n_estimators = N_ESTIMATORS,
                    objective = 'reg:squarederror',
                    # early_stopping_rounds = STOPPING,
                    random_state=RANDOM,
                    ),

            LGBMRegressor(n_estimators = N_ESTIMATORS,
                    objective = 'mse',
                #     early_stopping_round = STOPPING,
                    random_state=RANDOM,
                    ),

            CatBoostRegressor(n_estimators = N_ESTIMATORS,
                    loss_function = 'RMSE',
                    silent = True,
                #     early_stopping_rounds = STOPPING,
                    random_state=RANDOM,
                    ),

            RandomForestRegressor(n_estimators = N_ESTIMATORS,
                    criterion = 'squared_error',
                #     early_stopping_rounds = STOPPING,
                    random_state=RANDOM,
                    ),

            GaussianProcessRegressor(kernel = Matern(),
                                    n_restarts_optimizer = 10,
                                    random_state = RANDOM)

            ]
    st = []
    sv = []
    ft = []
    pr = []
    ev = []
    fi = []
    for i, model in enumerate(TB_models):
        print('({}) {:<4s} model'.format(datetime.now().strftime('%H:%M:%S'),
                                              TB_model_names[i]),
                                              end='')
        # Training phase
        t0 = time.time()
        if TB_model_names[i] == 'GPR' or 'RF':
            model.fit(X_train_flat, Y_train)
        else:
            model.fit(X_train_flat,
                      Y_train,
                      eval_set = [(X_train_flat, Y_train), (X_val_flat, Y_val)],
                      verbose=False)
        ft.append(time.time() - t0)
        print(' | Fit: {:5.2f}"'.format(ft[i]), end='')

        # Predictive phase
        t1 = time.time()
        pr.append(model.predict(regressor_flatten(X)))
        st.append(time.time() - t1)
        print(' | Predict: {:5.2f}"'.format(st[i]), end='')

        # Save model
        t2 = time.time()
        dump(model, '{}_CV{}.joblib'.format(TB_model_names[i], iteration))
        sv.append(time.time()-t2)
        print(' | Save: {:5.2f}"'.format(sv[i]))

        # Feat importance
        if TB_model_names[i] != 'GPR':
            fi.append(model.feature_importances_)

    print('({}) [{}] Results saved!'.format(datetime.now().strftime('%H:%M:%S'), '\
        u2713'))
    TB_fitting_times.append(ft)
    TB_setting_times.append(st)
    TB_saving_times.append(sv)
    TB_feat_importance.append(fi)
    TB_predictions.append(pr)
    TB_evaluations.append(ev)
    print('({}) Cross-Validation performed in {:6.2f} seconds\n{}\n\n'.format(
        datetime.now().strftime('%H:%M:%S'), time.time()-start_time, 70*'='))

# Save results
np.save("TB_fitting_times.npy",   TB_fitting_times)
np.save("TB_setting_times.npy",   TB_setting_times)
np.save("TB_saving_times.npy",    TB_saving_times)
np.save("TB_feat_importance.npy", TB_feat_importance)
np.save("TB_predictions.npy",     TB_predictions)
np.save("TB_evaluations.npy",     TB_evaluations)
```

Script 18: Prophet regression for the close price of the AstraZeneca company.

```
from prophet import Prophet
from prophet.utilities import regressor_coefficients
from prophet.plot import plot, plot_components_plotly

m = Prophet()

```

```python
# Create customized Prophet-dataset
prophet_structure = {
    'ds'      : df.index[: -from_2021],
    'y'       : df.Close[: -from_2021],
    'Open'    : df.Open[: -from_2021],
    'Net'     : df.Net[: -from_2021],
    'Volume'  : df.Volume[: -from_2021],
}

df_prophet = pd.DataFrame(prophet_structure)
df_prophet.index = np.arange(df_prophet.shape[0])

# Adding the set of regressors
for reg in df_prophet.columns[2:]:
    m.add_regressor(reg)

# Fitting Prophet model
m.fit(df_prophet)

# Create a test-set with its regressors
future = m.make_future_dataframe(periods=from_2021)
for reg in df_prophet.columns[2:]:
    future[reg] = df[reg].to_numpy()

# Make future predictions
forecast = m.predict(future)
m_eval_train = model_evaluation(df['Close'].to_numpy()[:-from_2021],
                                forecast.yhat[:-from_2021])

m_eval_val   = model_evaluation(df['Close'].to_numpy()[-from_2021:],
                                forecast.yhat[-from_2021:])

print(pd.DataFrame([m_eval_train, m_eval_val], columns=metrics_names, index=['Train',
     'Val']).to_latex())

# Save results
forecast.to_csv('prophet_forecast.csv')
# Plot results
plt.figure()
plt.plot(df.index[-from_2021 :], df['Close'].to_numpy()[-from_2021 :], label='Real')
plt.plot(df.index[-from_2021 :], forecast['yhat'].to_numpy()[-from_2021 :], linestyle
    ='-.', label='Predicted')
plt.fill_between(df.index[-from_2021 :],
                        forecast['yhat_lower'][-from_2021 :],
                        forecast['yhat_upper'][-from_2021 :],
                        color = 'tab:orange',
                        alpha = 0.2,
                        label = 'Uncertainty')
plt.title('Prophet test forecasting')
plt.legend()
plt.tight_layout()
plt.savefig('Prophet_validation.pdf')
```

C.2.2 *Case two: Front-Back sound discrimination on HRTF data*

Script 19: Implementation of the fitting stage for the KNN, MLP, SVM, Linear regression, and XGB classification models.

```
import time
from datetime import datetime
from joblib import dump, load
import pandas as pd

from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix,
    ConfusionMatrixDisplay
from sklearn import neighbors, linear_model, ensemble, svm
from sklearn import metrics

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier

def from_classes_to_binary(X_train, X_val, y_train, y_val):
  # SELECT NON-LATERAL CLASSES
  IDX_binary_train = np.array([__y in [0,1,2,4,5,6] for __y in y_train])
  IDX_binary_val  = np.array([__y in [0,1,2,4,5,6] for __y in y_val])
  # REMOVE LATERAL CLASSES
  X_binary_train = X_train[IDX_binary_train,:,:]
  X_binary_val   = X_val[IDX_binary_val,:,:]
  Y_binary_train = y_train[IDX_binary_train]
  Y_binary_val  = y_val[IDX_binary_val]
  # CONVERSION TO BINARY
  for i, __y in enumerate(Y_binary_train):
    if __y in [0, 1, 2]:
      Y_binary_train[i] = 0
    else:
      Y_binary_train[i] = 1

  for i, __y in enumerate(Y_binary_val):
    if __y in [0, 1, 2]:
      Y_binary_val[i] = 0
    else:
      Y_binary_val[i] = 1
  return X_binary_train, X_binary_val, Y_binary_train, Y_binary_val

COLUMNS = ['Precision', 'Recall',
           'F1-score', 'Accuracy',
           'AUC', 'p-AUC',
           'Brier', 'Jaccard',
           'MCC', 'FM', 'Log-Loss']

def get_scores(y_pred, y_real, y_prob):
    acc = metrics.accuracy_score(y_real, y_pred)
    auc = metrics.roc_auc_score(y_real, y_pred)
    mcc = metrics.matthews_corrcoef(y_real, y_pred)
    fm = metrics.fowlkes_mallows_score(y_real, y_pred)

    return np.array(acc, auc, mcc, fm])

# PERFORM 9-FOLD CROSS-VALIDATION
X_idx = np.arange(Xm.shape[0])
```

```

model_names = ['kNN',
               'MLP',
               'SVM',
               'Logistic',
               'XGBoost',
               ]
fitting_times = []
setting_times = []
saving_times = []
feat_importance = []
predictions = []
scores = []
dec_matrix = []

N_cross_validations = 9
for n in range(N_cross_validations):
  print('({}) Cross-Validation number {}\n{}'.format(datetime.now().strftime('%H:%M:%
      S'), n+1, 80*'='))
  start_time = time.time()
  # Make a 36:9 of 45 subjects of split
  cv_idx = np.arange(n*1250, (n+36)*1250)
  data_split = np.in1d(X_idx, cv_idx)

  # Data split for train-validation
  X_train = Xm[data_split, :, :]
  y_train = yc[data_split]
  X_val = Xm[~data_split, :, :]
  y_val = yc[~data_split]
  X_binary_train, X_binary_val, Y_binary_train, Y_binary_val = from_classes_to_binary
      (X_train, X_val, y_train, y_val)

  # Sum the Ipsilateral/Contralateral components
  X_binary_train_1channel = X_binary_train.sum(axis=-1)
  X_binary_val_1channel = X_binary_val.sum(axis=-1)

  RANDOM = n*13
  models = [KNeighborsClassifier(n_neighbors=6, n_jobs=-1),
            MLPClassifier(random_state = RANDOM),
            SVC(random_state = RANDOM),
            LogisticRegression(random_state = RANDOM, max_iter=250, n_jobs=-1),
            XGBClassifier(random_state = RANDOM, n_jobs=-1),
            ]
  ft = []
  st = []
  sv = []
  fi = []
  pr = []
  sc = []
  dm = []
  for i, model in enumerate(models):
      print('({}) {:<8s} model'.format(datetime.now().strftime('%H:%M:%S'),
                                                model_names[i]),
                                                end='')
      t0 = time.time()
      model.fit(X_binary_train_1channel, Y_binary_train)
      ft.append(time.time()-t0)
      print(' | Fit: {:6.2f}"'.format(ft[i]), end='')
      # Just store the feature importance of the possible models
      if i < 3:
        fi.append(np.zeros(X_binary_train_1channel.shape[1]))
      elif i == 3:
        lr_intercept = model.intercept_
        fi.append(model.coef_)
      else:
        fi.append(model.feature_importances_)
      t1 = time.time()
      y_pred = model.predict(X_binary_val_1channel)
      if i == 2:
        y_prob = y_pred
      else:
        y_prob = model.predict_proba(X_binary_val_1channel).T[0]
      st.append(time.time()-t1)
      print(' | Predict: {:5.2f}"'.format(st[i]), end='')
      # Save model
      t2 = time.time()
      dump(model, '{}_CV{}.joblib'.format(model_names[i], n))
      sv.append(time.time()-t2)
      print(' | Save: {:5.2f}"'.format(sv[i]))
      # Save resultant objects
      pr.append(y_pred)
      sc.append(get_scores(y_pred, Y_binary_val, y_prob))
      dm.append(metrics.confusion_matrix(Y_binary_val, y_pred))

  # Save results for the Cross-Validation
  print('({}) [{}] Results saved!'.format(datetime.now().strftime('%H%M%S'), '\
      u2713'))
  fitting_times.append(ft)
  setting_times.append(st)
  saving_times.append(sv)
  feat_importance.append(fi)
  predictions.append(pr)
  scores.append(sc)
  dec_matrix.append(dm)
  print('({}) Cross-Validation performed in {:6.2f} seconds\n{}\n\n'.format(datetime.
      now().strftime('%H%M%S'), time.time()-start_time, 80*'='))

# Saver results
np.save("fitting_times.npy",   fitting_times)
np.save("setting_times.npy",   setting_times)
np.save("saving_times.npy",    saving_times)
np.save("feat_importance.npy", feat_importance)
np.save("predictions.npy",     predictions)
np.save("scores.npy",          scores)
np.save("dec_matrix.npy",      dec_matrix)
```

Script 20: Linear Discriminant Analysis classification strategy for the HRTF signals of the CIPIC dataset.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

clf = LinearDiscriminantAnalysis()
clf.fit(X_binary_train_1channel, Y_binary_train)
clf.score(X_binary_val_1channel, Y_binary_val)

# X-y values
```

```
y_LDA = clf.transform(X_binary_val_1channel)
x_LDA = np.arange(len(y_LDA))

# Graphic representation
plt.figure(figsize=(8, 5))
marker_LDA = ['^', 'o']
color_LDA = ['tab:red', 'tab:green']
label_LDA = ['Front', 'Back']
for p in range(2):
    sns.scatterplot(x=x_LDA[Y_binary_val == p],
                    y=y_LDA[:,0][Y_binary_val==p],
                    marker=marker_LDA[p],
                    color = color_LDA[p],
                    alpha = 0.5,
                    edgecolor = color_LDA[p],
                    label = label_LDA[p])
plt.xlim(0-10, len(y_LDA)+10)
plt.title('Linear Discriminant Analysis for 1-channel HRTF sources')
plt.legend(loc = 'upper right')
plt.tight_layout()
plt.savefig('CIPIC_LDA_1channel.pdf')
```

COLOPHON

I want to conclude by highlighting the leading websites that helped me develop my thesis. The document that yields this thesis has been compiled with LaTeX. The edition, storage, and online compilation have been in the collaborative cloud-based Overleaf. We would like to mention that our thesis style is provided by the author of the `ltu-thesis` GitHub repository. Then, we are very thankful for the beauty and elegance of their template. The reference managing has been carried out via the Mendeley Desktop. Regarding the graphics presented throughout the thesis, all the "Own elaboration" sources have been drawn by utilizing the following sources: Matplotlib for scientific plots, SlidesGo for standard graphics, and Inkscape for image edition.