UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE INGENIERÍA Y ARQUITECTURA

MÁSTER UNIVERSITARIO EN INGENIERÍA AERONÁUTICA

TRABAJO FIN DE MÁSTER

A MACHINE LEARNING APPROACH FOR THE PREDICTION OF AIRCRAFT VALUE DEPRECIATION

Author: Ángela Martín Milán
Tutor: Alan Domínguez Montero

April 2025

"Without data, you are just another person with an opinion"

- W. Edwards Deming

Abstract

The accurate estimation of aircraft price represents a pivotal decision-making aspect in the aviation industry, significantly impacting asset management, leasing, buy-sell transactions, and fleet maintenance. A precise price valuation of aircraft allows industry stakeholders to optimize costs, evaluate depreciation, and plan strategic renovations, while at the same time, financial risks are mitigated and long-term profitability is improved.

This dissertation is focused on the development of a predictive model capable of estimating the price of aircraft given a set of historical observations. For this purpose, diverse regression machine learning models are trained and evaluated, including Linear Regression, Decision Tree, K-Nearest Neighbors, Support Vector Regression, and Artificial Neural Network algorithms. Additionally, linear regularization approaches as the Lasso, Ridge, and Elastic Net models are explored, together with Random Forest and Gradient Boosting tree-based embedding techniques. The diversity of models considered allows the subsequent selection of the most effective approach for the prediction of aircraft prices and the derivation of relevant insights from the study.

The research includes an extensive pre-processing stage, encompassing data exploration and adjustment techniques committed to the achievement of a consistent, relevant, and meaningful dataset, which will enable the certain learning of the algorithms. Thus, a series of data processing approaches are inspected, including data-cleaning and structuring methods, exploratory data analysis, treatment of outliers and missing values, and feature engineering techniques, embracing the encoding of categorical variables, the scaling of numerical features, and transformation of attributes. Thereafter, models are constructed and optimized by implementing grid search and cross-validation methodologies, which enhance the models' performance by selecting the most optimal algorithm's hyperparameters.

The results indicate that tree-based methods achieve the highest predictive accuracy and generalization capabilities. More specifically, the Random Forest model is found to provide the best overall performance, with a MAE of 110,297, a MAPE of 23.82%, and a R-squared value of 0.9585. These outcomes reveal the suitability and potential of regression algorithms for the high-level assessment of aircraft prices, being relevant for the provision of a first estimate during the price evaluation procedure.

Despite the acceptable predictive accuracy achieved, models' performance is influenced by data constraints, mainly associated with the lack of availability of complete and detailed aircraft records, including their price and significant attributes. Future studies should tackle these limitations by the consideration of more exhaustive, comprehensive, and consistent datasets, which may be complemented with the implementation of advanced pre-processing techniques, such as more proficient feature selection procedures and imputation mechanisms. Additionally, some more sophisticated model methods are identified as enablers for the optimization of the construction of the algorithm, such as the substitution of grid search by Bayesian techniques.

Key Concepts: *Artificial intelligence, machine learning, regression model, data pre-processing, aircraft depreciation*

Resumen

La estimación precisa del precio de aeronaves representa un aspecto fundamental en la toma de decisiones para la industria de la aviación, teniendo un impacto significativo en la gestión de activos, arrendamiento, transacciones de compra-venta, y mantenimiento de flotas. Una valoración certera del precio de aronaves permite a las personas de interés de la industria optimizar costes, evaluar su depreciación, y planear renovaciones estratégicas, mientras que al mismo tiempo, se reducen riesgos financieros y se mejora la rentabilidad de estos activos a largo plazo.

Esta tesis está enfocada en el desarrollo de un modelo predictivo capaz de estimar el precio de aeronaves dado un conjunto de registros históricos. Con este objetivo, diversos modelso de regresión de aprendizaje autónomo son entrenados y evaluados, incluyendo algoritmos de Regresión Lineal, Árboles de Decisión, K-Nearest-Neighbors, Vectores de Soporte, y Redes Neuronales. Adicionalmente, se exploran enfoques de regularización lineal como los modelos Lasso, Ridge, y Elastic Net, junto a técnicas embebidas basadas en árboles como Random Forest y Gradient Boosting. La diversidad de los modelos considerados permite la consecuente selección del enfoque más efectivo para la predicción de precios de aeronaves y la derivación de hallazgos relevantes del estudio.

La investigación incluye una etapa de pre-procesamiento exhaustiva, englobando la exploración de los datos y la implementación de técnicas de ajuste compremetidas a la obtención de un conjunto de datos consistente, relevante y significativo, que permita el aprendizaje seguro de los algoritmos. De este modo, se inspecciona una serie de enfoques de procesamiento, incluyendo métodos de limpieza y estructuración, análisis exploratorio de datos, tratamiento de valores atípicos y faltantes, y técnicas de ingeniería de características, considerando la codificación de variables categóricas, el escalado de los características numéricas, y la transformación de atributos. Posteriormente, se construyen y optimizan los modelos con la implementación de metodologías como grid search y validación cruzada, lo que permite el refuerzo del rendimiento de los modelos con la selección de los hiperparámetros más óptimos.

Los resultados indican que los modelos basados en árboles son los que obtienen la mayor precisión predictiva y capacidad de generalización. Más especificamente, el modelo Random Forest ha proporcionado el mejor rendimiento global, con un MAE de 110,297, un MAPE de 23.82%, y un R-squared de 0.9585. Estos resultados revelan la adecuación y el potencial de los algoritmos de regresión para una evaluación a alto nivel de los precios de aeronaves, siendo relevantes para la asignación de una primera estimación durante el proceso de evaluación de precios.

A pesar de la aceptable precisión predictiva alcanzada, el rendimiento de los modelos se ha visto influenciado por limitaciones en los datos, principalmente asociadas con la carencia de registros de aeronaves completos y detallados, incluyendo sus precios y características relevantes. Estudios futuros deberán abordar estas limitaciones mediante la consideración de conjuntos de datos más exhaustivos, íntegros, y consistentes, los cuales pueden ser complementados con la implementación de técnicas de pre-procesamiento avanzadas, como procedimientos de selección de características y mecanismos de imputación más especializados. Adicionalmente, se identifican algunos métodos más sofisticados para la optimización de la construcción de los modelos, como la sustitución de técnicas de grid search por Bayesianas.

Conceptos clave: *Inteligencia artificial, aprendizaje automático, modelo de regresión, pre-procesamiendo de datos, depreciación de aeronaves*

Acknowledgements

En primer lugar, me gustaría expresar mis más sinceros agradecimientos a mi tutor, Alan Domínguez Montero, por confiar en mí el desarrollo de este proyecto. Desde el primer momento mostrándome su profunda dedicación, cercanía y compromiso. Su orientación y apoyo han sido los pilares fundamentales que han impulsado mi inmersión en este campo de estudio.

Igualmente, me gustaría dar las gracias a la Universidad Europea de Madrid y a todos sus profesores y trabajadores, que hacen posible que estudiantes como yo continúen con su formación en un ambiente dinámico, estimulante y enriquecedor. A mis compañeros del máster, gracias por la complicidad y el buen humor en los momentos más exigentes, haciendo este viaje mucho más ameno incluso a pesar de los horarios no convencionales.

Finalmente, y con gran cariño, quiero agradecer a las personas que siempre están a mi lado, mi familia y amigos. Gracias por vuestro apoyo incondicional, entender mis ausencias, motivarme en momentos de duda y celebrar cada pequeño logro. Mención especial a mis padres, por impulsarme en la realización de este máster y su gran apoyo incluso desde la distancia.

Contents

List of Figures

List of Tables

Acronyms

AAGR	Average Annual Growth Rate
AGI	Artificial General Intelligence
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
CLT	Central Limit Theorem
CORSIA	Carbon Offsetting and Reduction Scheme for International Aviation
DL	Deep Learning
EASA	European Aviation Safety Agency
EDA	Exploratory Data Analysis
ERP	Enterprise Resource Planning
FAA	Federal Aviation Administration
GDP	Gross Domestic Product
GPU	Graphical Processing Units
ICAO	International Civil Aviation Organization
ICT	Information and Communications Technology
IFR	Instrumental Flight Rules
INE	Instituto Nacional de Estadística
IQR	Interquartile Range
KNN	K-Nearest Neighbors
LOF	Local Outlier Factor
MAD	Mean Absolute Difference
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MAR	Missing At Random
MCAR	Missing Completely At Random
MedAE	Median Average Error
MICE	Multivariate Imputation by Chained Equations
ML	Machine Learning
MNAR	Missing Not At Random

MSN	Manufacturer Serial Number
OC-SVM	One-Class Support Vector Machine
PCA	Principal Components Analysis
RBF	Radial Basis Function
REG	Registration Number
RMSE	Root Mean Squared Error
ReLu	Rectified Linear Unit
RFE	Recursive Features Elimination
SVM	Support Vector Machine
SVR	Support Vector Regression
TBO	Time Between Overhauls
TPU	Tensor Processing Units
UCI	University of California, Irvine
VFR	Visual Flight Rules
WEF	World Economic Forum

Nomenclature

Q_1	First percentile
Q_3	Third percentile
x	Predictor variable
y	Target variable
$\bar{x}$	Mean of predictor variable
$\bar{y}$	Mean of target variable
$\hat{y}$	Predicted target variable
n	Number of samples
k	Number of columns and number of splits in k-cross-validation
r	Number of rows
μ	Mean
σ	Standard deviation
χ	Chi-squared coefficient
m	IQR factor
R	Linear Pearson's coefficient
V	Cramer's V coefficient
PB	Point-Biserial coefficient
D	Mahalanobis distance
S	Covariance matrix
x'	Transformed feature
λ	Power of Box-Cox and Yeo-Johson transformations
w	Weights/coefficients in linear regression or ANN models
$L1$	Lasso penalization term
$L2$	Ridge penalization term
α_L	Lasso regularization
α_R	Ridge regularization
α_{EN}	Elastic Net regularization
l_1	Lasso-Ridge regularization ratio
max_depth	Maximum tree depth in tree-based models
$min\_samples\_leaf$	Minimum samples per leaf in tree-based models

$min\_samples\_split$	Minimum samples per node in tree-based models
$max\_features$	Number of features for divisions in Random Forest models
$n\_estimators$	Number of trees in Random Forest and GBoost models
α_{LR}	Learning rate in GBoost and ANN models
d	Minkowski distance
p	Power of Minkowski equation
k	Number of neighbors in KNN models
$weights$	Impact of closest neighbors in KNN models
$algorithm$	Method for neighbors search in KNN models
ϵ	Loss function in SVR models
C	Regularization parameter in SVR models
$kernel$	Hyperplane function in SVR models
γ	Non-linear kernel parameter in SVR models
z	Neuron input/output in ANN models
b	Neuron bias in ANN models
$activation$	Activation function in ANN models
$hidden\_layer\_sizes$	Number of hidden layers and neurons in ANN models
$solver$	Algorithm for weight adjustment in ANN models

1
Introduction

1.1. Motivation and Contextualization

1.1.1. A Glimpse into Aviation Economic Status

The origins of aviation can be traced back over two millennia, spanning from the first signs of flying objects similar to kites and daring attempts at tower jumping to supersonic and hypersonic flights. It was in 1920 when commercial flights started gaining traction, becoming aviation a relevant economic activity [2]. Now, the aviation industry represents an increasingly globalized market with a strong impact on the world economy, supporting over 4.1 trillion dollars (3.9%) of the global Gross Domestic Product (GDP) [12] and expecting an 8.5 trillion-dollar contribution by 2043.

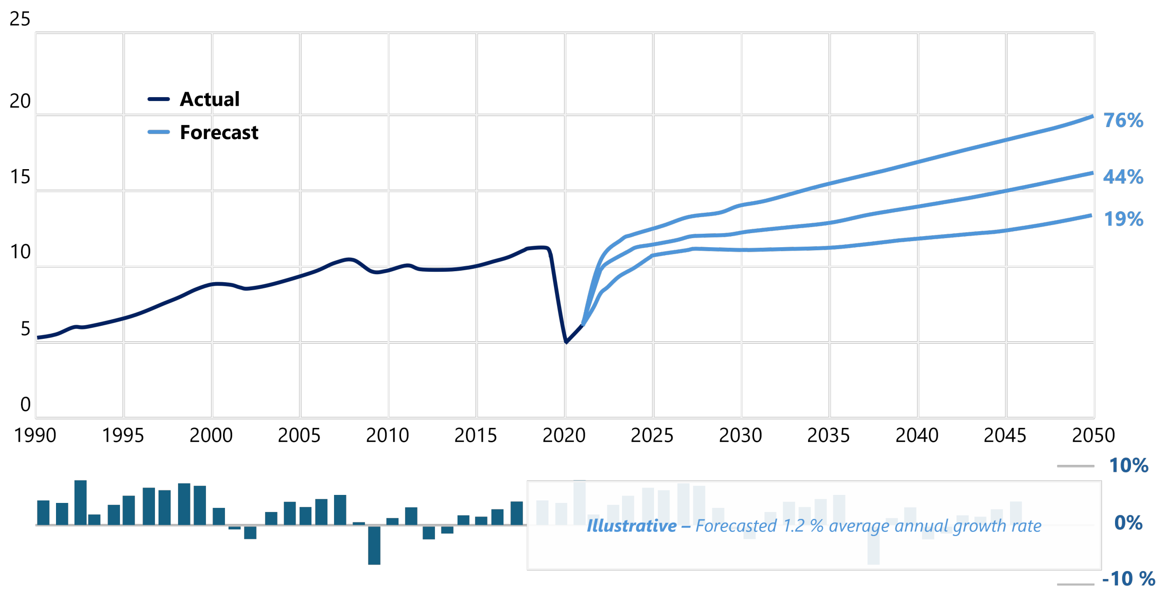


Figure 1.1: Evolution of the number of flights in Europe from 1990 until current times, including high, base, and low forecast scenarios to estimate its tendency until 2050. The bar chart below the main graph indicates the actual and expected Average Annual Growth Rate (AAGR) for the period considered.

As forecasted by Eurocontrol's General Director, Eamoon Breannen, in its last annual report [21]: *"There will most probably be 16 million flights in 2050, increasing demand by a 44% over 2019 at an average annual rise of 1.2% per year"*. Although this increase in aviation activity seems to be significant, the reality is that this estimation is 10-year lagged compared to the last long-term forecast provided by Eurocontrol in 2018 [20]. The reasons behind these degraded new estimations are mainly associated with the impact of COVID-19 pandemic, required investments to reach 2050 environmental expectations, and unstable global politics, which inevitably imply a limitation on aviation growth.

While substantial advancements in the aviation industry are anticipated in the coming decades, the path forward is unlikely to be straightforward, compelling market stakeholders to confront numerous challenges and threats. As today, the aviation industry can turn the page on the detrimental situation generated by the COVID-19 pandemic, catching 2019 traffic and returning to profitability. Although this positive tendency is willing to persist, the pace of improvement is likely to slow due to the challenging global stability, which can significantly affect an industry generally known by its slim margins [109].

At a universal level, the vast majority of the world population went to the polls in over 70 countries by the end of 2024. Many of these elections were characterized by a strongly polarized setting, which can result in significant policy changes. This will happen in a global market that is still very tight, therefore contributing to keeping sticky prices and high inflation levels that will take longer to decrease than the target foreseen by central banks. All of this is being translated into policy interest rates that will be maintained high, especially in the US, where higher rates for longer periods will appreciate the value of the US dollar against most of the other world currencies. A characteristic that will hinder growth for most countries and adds costs to all debt and invoices denominated in US dollars for organizations that are not dollar-based [55].

Considering industry specifications, for aeronautical stakeholders, such as airliners, manufacturers, and consumers, the global financial forecast will be heightened by some additional factors, including:

- Exposure to countries' restrictions concerning international trade, which is a consequence of the changing policy forecast that can involve increasing tariffs.
- Geopolitical conflicts that generate global instability, which affects a wide variety of markets; however, the aviation industry is directly impacted by the closure of the airspaces of the involved states. This supposes a limitation for route optimization, increasing flight trajectories and therefore, associated fuel costs.
- Increase in the price of oil since the outbreak of the war in the Middle East, spreading the gap between jet fuel and crude oil prices.
- Organization's resources provision to absorb new potential passenger and cargo traffic, meaning high investment in the acquisition of aircraft to absorb the increasing demand.
- Stakeholders' commitment to delivering net-zero CO_2 emissions by 2050 will require an investment in favoring fuel-efficient and quieter vehicles, including costs for aircraft design, fuel development, and infrastructure adaptation. Additional fees are charged by the Carbon Offsetting and Reduction Scheme for International Aviation (CORSIA), a global market-based measure established by ICAO to limit carbon emissions [19].

All of these threats will not alleviate the industry's profitability, requiring short margins to absorb their associated mitigation costs, impeding the strengthening of companies' balance sheets that are still facing an excess of debt since the COVID-19 pandemic. This situation obliges aviation stakeholders to monitor economic and strategic business decisions thoroughly.

In that sense, the aviation industry dedicates a large part of its effort to the optimum management of its main assets, aircraft. The high price associated with these resources, along with their price change over time, makes aircraft a fundamental pivotal point for economic decisions.

The subsequent section will deep-dive into the importance of aircraft valuation, the causes and reasons behind their depreciation over time, and the main mechanisms to evaluate this price decrease. This explanation will allow the understanding of the key principles behind the efficient management of fleets, differentiating between profitable operations and those with high financial costs.

1.1.2. Aviation Assets Management: Aircraft Price Evaluation

As outlined in the previous section, the aviation sector can be categorized as a capital-intensive market where the accounting for assets significantly influences financial results. More specifically, aircraft constitute the basis of this asset accounting, being high-cost resources that normally require payments in advance of the acquisition, which may result in substantial financial interest. Additionally, transactions in the aviation market are commonly designated in US Dollars, harming non-US stakeholders [56].

Another consideration of particular importance is that aircraft correspond to long-life assets consisting of multiple individual components that tend to deteriorate, experiencing a price change over time. In this way, contrary to some other resources which are prone to appreciate, increasing their purchase cost (e.g., the housing market), aircraft display a depreciation tendency starting from a maximum value at the time of the acquisition until reaching a residual value at the end of their life. The overall depreciation of an aircraft is estimated by categorizing and analyzing each of its components individually based on their defined useful life, resulting in a range of individual depreciations that are subsequently weighted.

To completely understand aircraft depreciation, it is crucial to explore its causes, the methodologies used for its estimation, and its implications in the industry, which will be covered in the following paragraphs.

Factors affecting aircraft depreciation

In general terms, depreciation is caused by the deterioration of the aircraft due to their exposure to wear and tear, technological obsolescence, or shifts in market tendencies. These factors may significantly impact the useful life of the asset and boost the rate at which depreciation occurs [6]:

- Age and condition: Similar to many other assets, aircraft are subjected to physical deterioration due to their operation. As aircraft get older, they accumulate flight hours, take-off and landing cycles, and any other damaging conditions derived from the execution of a flight. This wear and tear increases the need for proper maintenance or component upgrades, decreasing their market price.
- Technology obsolescence: The aviation industry presents a wide dynamism and adoption of new technologies, being in constant research for the development of more advanced and efficient models. As aircraft with more cutting-edge features are introduced, those older models experience a decrease in their value, although still operational. This depreciation is especially relevant for the technology that constitutes navigation and communication systems aboard aircraft.
- Regulatory demands: Aeronautical activities are highly regulated by international organizations such as the Federal Aviation Administration (FAA) in the US, the European Aviation Safety Agency (EASA) in Europe, or the International Civil Aviation Organization (ICAO) at a global level. These institutions establish the requirements that have to be fulfilled by aircraft in terms of security, carbon and noise emissions, maintenance, and certifications in order to be suitable for flying. Any emerging regulatory policies may significantly impact aircraft depreciation when the aircraft can not be adapted to the new requirements.
- Market tendency: Customers' needs and preferences may leverage specific types and models of aircraft. The last tendencies show an increase in the demand for low-cost flights with more efficient propulsion systems. Aircraft fulfilling the requirements to operate according to these demands will experience a more limited depreciation in comparison to those aircraft designed for different purposes.
- Economic conditions: The global economic status can significantly influence aircraft demand. While high inflation, instabilities, or fluctuations in fuel prices may produce a retirement of customers and investors, reducing demand and increasing depreciation rates, a stabilized economic market with low interest rates will boost demand, reducing depreciation.

Depreciation estimation methods

Once the depreciation concept is understood and the causes behind aircraft devaluation have been explored, the determination of its value is crucial for the guidance of stakeholders' decisions. Traditionally, three main depreciation methods have been used for its estimation [66]:

1. Straight-Line depreciation:
   Corresponding to the most straightforward and broadened extended method, the Straight-Line depreciation assumes a constant depreciation during the useful life of the aircraft, meaning that the original cost is evenly reduced over the years until reaching its final residual value. Although easy to implement, this method does not reflect the loss of value in terms of aircraft usage or obsolescence.

$$\textit{Straight-Line depreciation} = (\textit{Original cost} - \textit{Residual value}) \ / \ \textit{Useful life} \qquad (1.1)$$

2. Declining Balance Depreciation:
   In this case, depreciation is also distributed over the useful life of the aircraft, however, it considers that assets often depreciate more rapidly in their early years. This method is classified as an accelerated one, applying a constant rate of devaluation to the declining book value [1] of the asset each year.

   $$\textit{Declining Balance depreciation} = \textit{Previous year's book value} \times \textit{Depreciation rate} \qquad (1.2)$$

   The Declining Balance technique offers a more representative behavior for aircraft, which normally suffer their largest devaluation during their first years. Additionally, as this approach results in higher depreciation expenses in the initial stages, smaller income is reported, leading to reduced applicable taxes during the early years.

3. Units of Production:
   This method escalates depreciation according to the actual usage of the aircraft, normally represented by hours of flight or cycles. This approach is the most tied to the actual state of the aircraft. However, it requires a close tracking of usage and maintenance of data, which can result in a non-homogeneous and difficult-to-predict depreciation.

Impact of depreciation on the aeronautical industry

As outlined in the previous paragraphs, the accurate estimation of depreciation is critical for the proper financial analysis of aeronautical corporations, allowing for the optimum decision-making mechanisms. More specifically, depreciation can have a significant impact on stakeholders when exploring the areas itemized below [93]:

- Financial reporting: Accounting standards make mandatory the inclusion of aircraft depreciation into the annual financial reports, which may have an important impact on the income statement and illustrate the actual profitability of aircraft as assets.

- Fleet replacement: The aviation industry must determine the appropriate time to replace aircraft, not only by considering their efficiency and required maintenance, but also by taking into consideration their market value and potential for revenue generation.

- Resale value: Understanding how aircraft depreciate is determinant for the estimation of its purchase-sale price in the secondary market. In this way, buyers and sellers need to acknowledge both the aircraft price at the time of the deal and its capacity to generate future income. The same reasoning is used for the estimation of leasing costs.

- Tax deductions: The reduction of assets‘ value positively influences companies’ tax liability, leading to substantial tax deductions as depreciation is claimed as an expense, reducing the taxed income. In fact, as seen before, the method used to calculate depreciation can affect the timing of the deductions.

- Insurance and financing: The rate at which an aircraft depreciates can significantly influence insurance premiums and financing terms, as they are normally based on the aircraft's current market value. For this reason, the regular update of the aircraft price is fundamental for preventing the overpayment for coverage/finance of an asset that exceeds its actual worth.

This section has provided a detailed definition of the depreciation concept, emphasizing its importance within the aeronautical market where its accurate estimation may lead to more informed business decisions in an industry with quite small margins. This overview provides a contextualization of the reasons that have boosted the development of this study, which approach and objectives are detailed in the following sections.

[1] In financial terms, the book value corresponds to the original cost of an asset minus its accumulated depreciation over time, or at a global level, the company's total tangible assets minus its liabilities.

1.2. Study Approach and Scope

Under the context described in the previous section, the author of this study identifies an opportunity to explore the suitability of Machine Learning (ML) algorithms as tools for accurately estimating aircraft prices. In this way, considering the possibility of turning the page on traditional depreciation methods, and establishing the pillars for the potential evolution towards a more precise, exploratory, and non-labor-intensive approach.

With this purpose, this dissertation will be focused on the development of a predictive model capable of estimating the price of aircraft given a certain set of characterizing attributes. The implementation of the tool will be based on the training of an algorithm with a historical dataset including descriptive information about the status and technical features of aircraft, together with their price. In this way, the model will be able to recognize relationships and patterns within the data, which will allow the evaluation of newly introduced aircraft.

The development of the model will require a prior analysis and treatment of the training data, whose purpose is to ensure that the model is fed with consistent and meaningful information. Additionally, different machine learning algorithms and optimization techniques will be explored, allowing the selection of the approach providing the best results, i.e., the model that best estimates aircraft prices.

Finally, the development of this thesis will place equal emphasis on the identification of the most relevant features that influence an aircraft's market value, providing insights into the key aspects driving aircraft depreciation following a data-driven approach.

1.3. Outlined Objectives

This section presents a detailed view of the objectives of the study, which have been formulated to ensure the guidance and execution of the research. In addition, the definition of these targets will provide a clear and structured framework for the evaluation of the posed problem. In this way, the main objectives of the study are outlined below:

- Development of an organized, succinct, and clear project report to guide the development of this thesis, helping not-so-experienced people get an idea of the reasons, results, and conclusions obtained.
- Definition and understanding of the key principle behind Artificial Intelligence (AI) and Machine Learning (ML), highlighting its relevance in the prediction and data-analysis fields, and establishing the theoretical and computational framework for the implementation of predictive models within the context.
- Revision of the complete machine learning implementation roadmap, paying special attention to the stages involving the pre-processing of the training data and the algorithm's implementation.
- Exploration of usual pre-processing techniques, including a thorough study of general cleaning and inspection approaches, data analysis procedures, outliers treatments, mechanisms for the handling of missing values, and further feature engineering techniques, for the adjustment of the dataset to the most suitable learning of the algorithm.
- Application of studied pre-processing approaches to the specific training dataset.
- Identification of suitable machine learning algorithms for the problem under concern, exploring the hyperparameters that define them, and any possible optimization mechanism that could enhance their performance.
- Revision of machine learning evaluation metrics, understanding their significance and applicability.
- Implementation of baseline algorithms for all the models under study, serving as a comparison reference for the subsequent implementation of refinement mechanisms.
- Iteration of different feature engineering approaches, assessing their impact on the model's performance and allowing the identification of the most optimal pre-processing techniques.

- Optimization of the models with the selection of the most suitable feature engineering mechanisms, and the performance of any other advanced technique for the enhancement of the algorithm, such as the search of their most optimal hyperparameters.
- Recognition of those dataset features that are most relevant for the estimation of aircraft price and identification of data limitations on the implementation of the model.
- Selection of the model providing the best results, and identification of possible areas of improvement.

1.4. Structure of this Thesis

The structure of this dissertation has been established to provide a clear description of the study while covering all of the objectives outlined in the previous section, identifying the content blocks represented in Fig. 1.2.

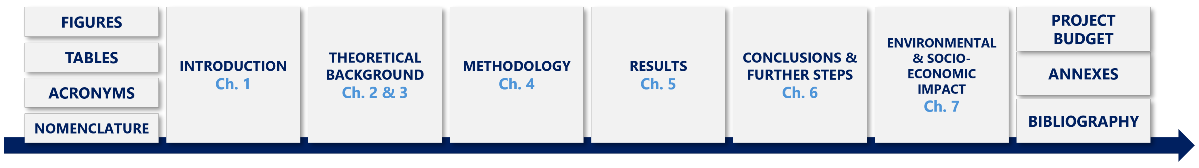


Figure 1.2: General overview of the main content blocks included in this dissertation, and their associated chapters.

The report commences with a series of pages listing the different figures and tables included throughout the report, to provide a more illustrative representation of the concepts covered and the results obtained. Additionally, lists of acronyms and nomenclature are included with the intention of guiding readers through this text.

Subsequently, the present section is approached, describing the contextualization of the addressed topic and allowing the statement of the scope of the study, its objectives, and methodology to reach its accomplishment.

After the introduction of the topic, the two most relevant and extensive blocks of this thesis are addressed: the theoretical background behind the implementation of machine learning algorithms, and its subsequent application to the study under concern.

Theoretical background

The theoretical segment is initiated with an introduction to the pillars defining artificial intelligence and allocating the concept of machine learning within its huge ecosystem. This section explores the classification of the different approaches within AI, highlighting the main methodologies employed for machine learning applications. Additionally, it presents the primary stages followed during the deployment of ML algorithms, from the definition of the problem to the implementation of the model. Similarly, the main tools that sustain the implementation framework are presented.

With a clear definition for AI and ML established, the text deep-dives into the crucial stages that define the development of predictive models, especially exploring the pre-processing of the training data and the implementation of ML algorithms. This section identifies the critical steps that need to be considered in each stage, providing a detailed vision of the most common techniques that are applied for the optimization of data and algorithms. Going into further details, the pre-processing theoretical revision delves into the explanation of data-cleaning and structuring techniques, methodologies for the treatment of outliers, handling of missing values, and exploration of data. Additionally, it addresses the features engineering transformations required for the adjustment of data to the most optimal understanding of the model. This last segment serves as an introduction to the model implementation stage, where the foundations of the different algorithms are presented, together with a discussion of the evaluation metrics that allow the assessment of the models. Finally, this section concludes with the presentation of a series of techniques for the optimization and adjustment of the models.

Implementation Methodology

The second substantial block is centered around the practical implementation, in which the theoretical pre-processing approaches previously described are applied to the specific data frame, adjusting and justifying each of the techniques used with the particular characteristics of the data.

Once the pre-processing of the data is complete, the implementation of an accurate and reliable predictive model is undertaken. In this phase, an iterative process is deployed, in which different feature engineering approaches are evaluated for the distinct models. Each iteration is assessed by comparing a series of evaluation metrics to observe the impact of these approaches on the performance of the models. Throughout this process, the most optimized model will be aimed at, based on the results from the feature engineering impact analysis, the selection of the most optimal hyperparameters, and the implementation of some refinement and optimization techniques.

After the completion of the development of the distinct algorithms, the most optimized results obtained for each of the approaches are presented, allowing the identification of the most appropriate model and the discussion and derivation of insights.

Afterwards, the main findings and takeaways of the study are highlighted in a dedicated section, together with the identification of further studies that could be addressed as next steps for the enhancement of the results.

The study culminates with a comprehensive environmental and socio-economic analysis, assessing the impact of this project on these fields by delving into both the positive contributions and challenges posed by the research.

Finally, some pages are dedicated to the evaluation of the budget consumed during the development of the project.

Additionally, it could be of great interest to remark on the presence of three annexed sections at the end of the report. One of them is dedicated to providing the location where the scripts developed during the project are stored and where they can be accessed. The remaining two annexes address the presentation of intermediate iterations and learning capabilities concerning the implementation of the models. Furthermore, the references to all of the consulted sources of information, mainly consisting of online sources, are cited in the bibliography chapter.

2

Introduction to Artificial Intelligence and Machine Learning: Framework and Tools for the Implementation of ML Algorithms

Recent years have been deeply marked by an exponential development of technological innovation, with *Artificial Intelligence (AI)* as one of the well-known forces leading the transformation of our era. Impacting a wide variety of sectors, AI has positioned itself as a disruptive tool capable of automating processes and optimizing decisions based on the analysis of large amounts of information. Within this field, *Machine Learning (ML)* can be highlighted as one of the most relevant branches of AI, providing algorithms proficient in identifying complex patterns and accomplishing high-precision predictions.

This chapter will commence with a general introduction to the concepts of AI and ML, highlighting their capabilities and understanding the possibilities of their different categorizations. Subsequently, the text will focus on the key stages defining the modeling of ML implementations, ranging from the initial definition of the project to the final model deployment. Additionally, this chapter will introduce the fundamental computational tools that enable the development of ML algorithms.

2.1. A Glance into Artificial Intelligence

The concept of *Artificial Intelligence* dates back thousands of years when the ancient philosophers started to convey the idea of machines performing a specific task on their own, a concept lately termed as *automatons* by the Greek period. However, it was in 1950 when the British logician and computer scientist, Alan Turing, published the *Computer Machinery and Intelligence* text, propelling the interest in AI systems. Subsequent decades were marked by both rapid growth and limitations for its research, facing some slowdown periods due to the low interest of private investors and governments. It was in 1997 when the *IBM* company developed a program capable of beating the best chess player in the world. Since this milestone, the global leading tech companies started to introduce AI into their commercial products, making it available in the everyday lives of millions of consumers [51].

In current times, AI refers to those computer systems capable of performing tasks that typically require human-like intelligence and problem-solving skills, using algorithms and computational models to understand datasets, detect patterns, and be able to make predictions based on data [15]. All of this is powered by its capacity to continuously learn over time from new information, significantly improving the system's performance.

This AI existing today is also known as *Weak AI* (or *Narrow AI*) when classified based on its capabilities. This term refers to those algorithms that are designed and trained to perform a specific task, without being able to operate beyond their designated purpose [58]. However, two additional categorizations of AI can be distinguished, although they remain purely theoretical concepts under investigation:

- The *Strong AI*, also termed as *Artificial General Intelligence (AGI)*, is intended to be auto-trained from previous learnings and perform different tasks from the ones it was initially designed to. This learning capability could provide computers with an intelligence comparable to that of humans.
- On the other hand, *Super AI*, if ever achieved, will learn, process information, build beliefs, understand sentiments, and feel emotions. The cognitive abilities of this type of IA are expected to surpass those of human beings.

Along with AI classification on capabilities, this new technology can also be grouped based on its functionalities. According to this classification, four general groups can be distinguished [28]:

- *Reactive Machine AI*, which corresponds to the more elementary type of AI. In this case, the systems do not have storage memory, working with the data available as of that time. Its results derive from intensive analysis of data employing statistical exploration.
- Different from the previous category, the *Limited Memory AI* can retain information from past interactions, complementing present data and providing enhanced outcomes.
- Diving into more complex models, the *Theory of Mind AI* would process thoughts and emotions from other interaction entities, being able to emulate human relationships.
- The last typology of AI that could be developed is the *Self-Aware AI*, which, going a step further, would be able to understand its own condition and interact with humans on an emotional level.

These different categorizations of AI will require more complex algorithms as their autonomy and cognitive capacities become more similar to those of humans. As may be derived from the previous description of AI capabilities, both the Reactive Machine AI and the Limited Memory AI can be identified as Narrow AI, defining the models behind a wide variety of applications today. On the other hand, the Theory of Mind AI and the Self-Aware AI correspond to Strong AI and Super AI, respectively. Although these last AI functionalities have not been achieved yet, the big technological players in the market are focused on their intensive research, as they could completely change the way we perceive the world today.

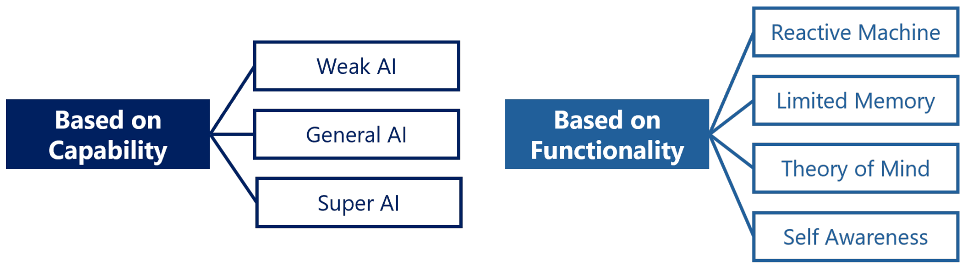


Figure 2.1: Representation of AI classification schemes, based on capabilities and functionalities.

Diving into the mechanisms that allow the implementation of the AI currently available, soon the notion of *Machine Learning* emerges as the most powerful and extensively used framework. In the next section, the fundamentals, classification, and implementation procedures for this new concept will be explored.

2.2. Machine Learning Fundamentals

Over the past few years, ML has been the most common, successful, and practical way to implement AI, creating some confusion between these two terms. Countless times, ML and AI are mentioned together, often resulting in the misclassification of these concepts as similar ones. As previously mentioned, AI is a broad term used to describe all the different approaches and methods that allow computer systems to do something smart. Alternatively, ML is a set of specific techniques to do that, involving the learning from data samples instead of the programming of explicit rules. In this way, ML becomes a subset of AI [43]. This relationship can be visualized in Fig. 2.2, where the concept of *Deep Learning (DL)* is introduced as a subset of ML, which will be more deeply explored in the next chapters.

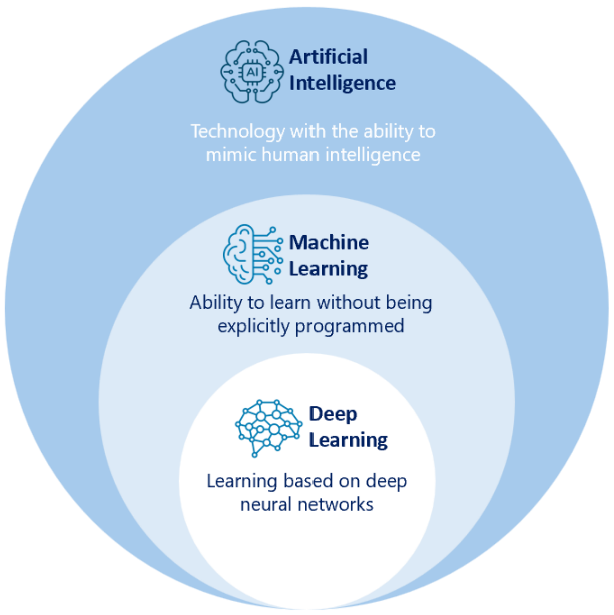

Artificial Intelligence
Technology with the ability to mimic human intelligence
Machine Learning
Ability to learn without being explicitly programmed
Deep Learning
Learning based on deep neural networks

Figure 2.2: Hierarchical relationships between Artificial Intelligence, Machine Learning, and Deep Learning.

A more extensive description of machine learning capabilities can be obtained by taking a look at the different learning techniques that can be implemented. In general terms, three main machine learning methodologies can be distinguished: *Supervised*, *Unsupervised*, and *Reinforcement Learning* [60], [103].

2.2.1. Supervised Learning

Supervised Learning can be selected as the most common learning method used by machine learning algorithms. In this approach, the model is trained with a set of labeled data. In simpler terms, each sample (input) in the dataset is defined by a series of features and is associated with a known label (output). In this way, the model learns the correspondence between the feature and its output, being able to predict results from unseen or future data.

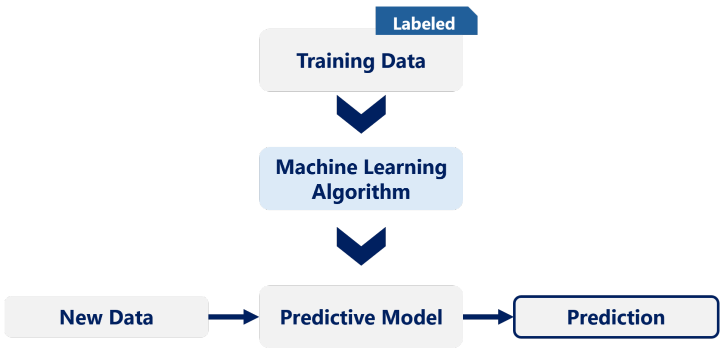


Figure 2.3: Graphical representation of supervised learning processes.

In supervised learning algorithms, the procedure for output prediction will vary depending on the type of label to be predicted. The estimation of a discrete class label will be based on *classification* tasks, while continuous outcomes will require a *regression* approach. These two distinct scenarios result in two different subcategories within supervised learning algorithms:

1. Classification for the prediction of class labels outputs:
   In this type of supervised learning, the model is trained with discrete outcomes, which can be understood as the group membership of a given instance. This is the case of the spam-detection algorithms used by email software, which learn a set of rules to distinguish between spam and non-spam emails. This example corresponds to a *binary classification* as only two output scenarios are possible; however, a classification model can assign any class labels presented in the training dataset, making it possible to perform *multi-class classification* tasks. Illustrative examples of multi-class classification are those algorithms capable of identifying a newly introduced character when trained with multiple handwritten letters and digits.

2. Regression for the prediction of continuous outputs:
   For this supervised learning subcategory, the model is trained with a certain number of attributes, named as *predictor* or *exploratory* variables, and their corresponding continuous target variable. The algorithm works on finding the influence of these predictor variables on the attribute that is to be predicted. In this case, when a new instance is introduced, a numeric continuous outcome is derived from the learned relationships, although that value was not previously included in the training dataset. An example of this type of supervised machine learning model is the prediction of annual incomes based on education level, work experience, industry sector, and age.

2.2.2. Unsupervised Learning

Contrary to what has been explored for supervised learning, unsupervised methods train the model with unlabeled data or datasets with an unknown structure. In this type of automatic learning technique, the algorithm finds data patterns and relationships by itself, allowing the discovery of hidden similarities or groups within the input data. Because of its exploratory character, this type of learning is commonly used for *clustering*, *association*, and *dimensionality reduction* tasks.

1. Clustering:
   This process involves organizing a pile of unstructured information into meaningful groups (clusters) according to their similarity. By this technique, data is explored, resulting in the identification of the different clusters and the classification of data points into the groups with more similar objects [57]. These aggregation processes are widely used for market segmentation activities, commonly implemented for customer classification and targeted offerings within companies.

2. Association:
   This unsupervised learning algorithm counts the frequency of co-occurring or related elements present on an extensive set of data, normally defined by thousands of variables. This model allows the identification of relationships and associations within information difficult to process and evaluate [91]. An application for this association rule learning can be found in supermarkets, where sales logs are explored to identify relationships between products, such as customers who buy milk and bread often also purchase butter. Leveraging these insights, sales can be improved by placing certain items together or targeting advertisements based on buying patterns.

3. Dimensionality reduction:
   It is not uncommon to frequently face situations where a dataset with a very high dimensionality needs to be processed. Normally, the large size of the dataset is caused by an extensive list of features measured for each of the observations recorded, although not all of the attributes are equally relevant to a given problem statement. In this scenario, dimensionality reduction can provide a reduction in noise by just maintaining the most meaningful information within the dataset. This process also decreases the storage space needed and improves the computational performance of the model [87]. Dimensionality reduction is commonly applied for the visualization of complex data, as high-dimensional structures are quite complex to interpret, but simpler 2-D or 3-D representations are rapidly processed.

2.2.3. Reinforcement Learning

The last type of machine learning is Reinforcement Learning, which considerably differs from the previous algorithms studied. In this case, the model is composed of an *autonomous agent* which learns how to perform a task based on its interaction with the *environment*. The learning is performed by evaluating the task performed by a so-called *reward signal*, in this way, the system follows a trial and error approach, learning the actions that maximize the reward signal. Thus, autonomous decisions are made in response to a given environment without needing human guidance or instructions [62]. Reinforcement learning is the machine learning technique behind the development of robotics devices, with self-driving cars being one of the latest innovations in the field.

Once the ML concept has been explored together with its main classifications and capabilities, the next step is to get an overview of the procedure to be followed for the implementation of these algorithms. Although different machine learning tasks have been presented, the fundamental stages constituting their development can be generalized for all of the cases. The next section will be focused on the explanation of the global implementation framework, which will establish the foundational pillars for the scope of this thesis.

2.3. Roadmap for the Implementation of a Machine Learning Model

The development of the majority of machine learning practices can be fitted into a global schema of sequential steps that ensures its traceability and optimization throughout the complete implementation process. In this way, any machine learning algorithm can be implemented following six general steps [101] [33]: *problem definition, data acquisition, data-preprocessing, model training, model evaluation,* and *model deployment*.

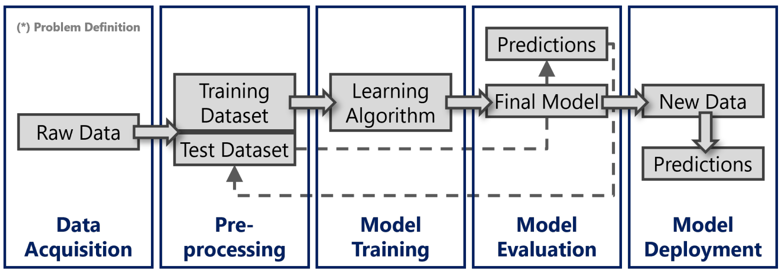


Figure 2.4: Flow diagram of the stages of the machine learning implementation, from the problem definition to the deployment of the model. Notice that although the problem definition corresponds to a strategic and theoretical step, it also needs to be considered in the implementation roadmap.

Initially, a preliminary study needs to be conducted before the implementation to precisely define the problem, the required data, and the main objectives associated with them. This will enable the specification and alignment of subsequent stages, tailored to the specific type of machine learning application.

Once the objectives and scope of the model are defined, the next step involves the collection of the data that will be used for the training of the algorithm in subsequent stages, ensuring that it contains the proper information for the problem to be addressed.

After the acquisition of the data, the next step consists on its pre-processing and preparation. This critical step will allow the understanding of the raw dataset and the refinement of the information to be more easily interpreted and analyzed by the machine learning algorithm. It encompasses a large variety of activities from the structuring, formatting, and cleaning of data, to the treatment of outliers and missing values, or the implementation of advanced feature engineering. During this stage, a preliminary data exploration and analysis are also performed to identify key patterns and relationships. The precise pre-processing of data may significantly influence the model's precision and performance.

The next step involves the implementation of the learning model, which will need a prior study of the different algorithms that may be suitable for the problem. Once the model to be built is selected, the adjustment of its hyperparameters is of huge importance for the precise control of the behavior of the algorithm.

After the implementation of the model, the next step is to evaluate its performance when conducting its designated predictive task. This procedure is performed by the computation of a series of evaluation metrics that will determine the deviation of the predictions from the real values. If not enough accuracy is achieved, an iterative process is put in place, repeating previous steps until a better performance is reached.

Once the model provides good predictions, the last stage in the construction process of a machine learning algorithm is its integration into a production environment for the development of its target function.

2.4. Tools and Technologies for the Construction of Machine Learning Algorithms

The computational tools employed for the development of ML projects directly impact the quality of the analysis of data, the performance of the algorithm, and the simplicity of its implementation and maintenance. Thus, they correspond to fundamental pieces ensuring the effectiveness, precision, and scalability of the model. Since programming languages, development platforms, or implementation environments, there exists a wide variety of tools, each of them performing a specific role within the machine learning framework.

The most basic aspect that will guide the selection of the tools is the programming language to be used. In general terms, there exists a group of languages designed for the development of high-performance and production code, such as C, C++, Fortran, or Java, while another collection is intended for prototyping purposes, such as Ruby or Python [72]. However, among all of these options, there is a language that distinguishes itself above the rest in the field of machine learning applications: the *Python* programming language.

This preference for Python is due to its great legibility, simplicity, and flexibility, making it accessible for not-so-experienced programmers. Additionally, although presenting lower computation-intensive capabilities compared to other options, Python enjoys a large number of libraries and frameworks built upon Fortran and C implementations. These libraries facilitate the programming process by the predefinition of mathematical operations, the treatment of data, or the implementation of machine learning algorithms, all of this being supported by an extensive and collaborative community [53].

In the case of this specific study, the *Anaconda* platform distribution will be used. It corresponds to an open-source Python distribution environment that clusters all fundamental Python packages for data science, math, and scientific computing in a user-friendly ecosystem.

Within the machine learning context, there exists an extensive selection of Python libraries that provide optimized tools for the realization of key activities such as the pre-processing of data, the implementation of algorithms, and their evaluation, accelerating the development process. Subsequently, the most relevant libraries for machine learning applications will be exposed, classified by their function within the implementation procedure [42], [34].

Numeric and scientific computation

- NumPy: This is the essential library for the realization of scientific calculations in Python. It supports the handling of multi-dimensional arrays and provides mathematical functions for the analysis of large amounts of data. NumPy is equally effective for linear algebra, Fourier transformation, and random number capabilities.

- SciPy: Based on NumPy, this library expands its capabilities with the addition of modules for optimization, statistics, signal processing, eigenvalue decomposition, etc.

- Matplotlib: Corresponding to a plotting library, Matplotlib can create static or interactive graphics and diagrams, constituting the main toolbox for data visualization.

Data Analysis

- Pandas: Although not directly related to machine learning, this library is the preferred tool for the preparation and analysis of data. It allows the importing and exporting of information in different formats while offering some complex functionalities for the structuring and manipulation of tables: addition/removal of information, merging or aggregation of datasets, etc. In this way, Pandas provides a set of tools capable of removing inconsistencies and achieving data alignment, having a critical role during the pre-processing step.

Machine Learning focused

- Scikit-learn: Being one of the most popular libraries for the implementation of traditional machine learning algorithms, Scikit-learn includes classification, regression, clustering, and dimensionality-reduction methods for both supervised and unsupervised algorithms.
- TensorFlow, Keras, and PyTorch: In this case, these three libraries are employed for the development of deep learning tasks by the use of neural networks. TensorFlow was the first open-source platform developed for deep learning techniques, being based on the resolution of high-performance computations involving tensors. On their side, Keras and PyTorch provide other alternatives for deep learning techniques, being simpler to use and more flexible.

With the description of the main tools used in machine learning modelization, the contextualization of the framework surrounding the work under scope can be considered closed. The next chapter will explore each of the details to be considered during the different phases defined in Section 2.3, which will present the theoretical basis for the computational segment of this project.

3
From Data to Prediction: Stages of Machine Learning Model Implementation

After the brief introduction of the ML implementation stages in the previous chapter, this section will focus on the exploration of those concepts in greater depth. In this way, the text will deep-dive into the main aspects to take into consideration throughout the different stages, emphasizing their scope and main techniques to achieve it.

The objective of this chapter is to provide readers with a detailed description of the procedure employed for the development of this work, ensuring reproducibility and justifying decisions taken throughout the algorithm implementation process. Additionally, the pillars for the comprehension of the obtained results are established, allowing the subsequent derivation of conclusions.

3.1. Stage 1: Problem Definition and Scope of the ML Model

The adequate definition of the problem to be addressed and the identification of its scope correspond to the first determinant step in the machine learning implementation process. Although it may seem a plain and straightforward phase, it is a critical stage that is often overlooked due to the eagerness to move to more advanced phases of the development of the algorithm [17].

The key purpose of this stage is to get a clear understanding of the question that needs to be solved and establish a precise implementation frame for the modeling [98].

In alignment with this, the phase is initiated with the identification of the key pillars of the problem, together with its main objectives. This facilitates the selection of the most optimal learning (supervised, unsupervised, by reinforcement, etc.), and the recognition of the kind of data that is needed for the training of the model.

This first stage will also guide the complete preparation and exploration of the data, defining the most appropriate pre-processing techniques and avoiding deviations from the central focus. Additionally, the selection of the most suitable algorithm approach may also be forecasted with a good understanding of the problem.

Finally, the assessment of the performance of the algorithm will be enhanced with a clear definition of the objectives of the project (precision, generalization capability), allowing the selection of the evaluation metrics that could successfully assess the algorithm's performance and facilitate iterations for optimization.

Therefore, as can be derived from previous paragraphs, the problem definition stage impacts the complete modeling process, ensuring the proper formulation from the early phases and providing a solid definition for the development of an effective model that fulfills objectives and contributes to the context in which it will be deployed.

3.2. Stage 2: Data Acquisition

As outlined in the previous chapter, machine learning models are no more than data-driven algorithms constructed based on mathematical equations and trained with a given set of data. Following the correct definition of the problem performed in the previous stage, the necessary knowledge is now available for the identification and selection of the proper data that will secure the training of the model, entering into the second stage of ML models implementation: *data acquisition*.

In a world shaped by data, there exists a wide range of possibilities for obtaining the dataset that will later feed the model. The choice of data largely depends on the nature of the problem; however, several data sources are commonly used in ML applications [77], [30]:

Open-Souce datasets

Open-source datasets correspond to data collections that are publicly accessible, allowing the use, modification, and distribution of their information. They are generally developed and maintained by developer communities, companies, or institutions that offer them without the payment of any fee. However, their lack of particularization to the problem under concern can limit their applicability, and in numerous cases, they require the implementation of a meticulous pre-processing of the information. Some well-known repositories of public datasets include [90]:

- Kaggle: An online Google platform that provides resources and a collaborative environment for data scientists and the ML community.
- UCI Machine Learning Repository: Constructed and maintained by the University of California, Irvine (UCI), this data collection is composed of more than 400 datasets for the development, test, and assessment of ML algorithms.
- Google's Datasets Search Engine: Different from previous examples, this platform does not store data, but it allows the search of datasets from different sources (academic, corporative, governmental) in a centralized way.

Data available on online platforms

Apart from the extraction of databases from dedicated repositories, information contained on platforms such as online websites or social media can also be publicly accessible. This generally corresponds to information about users' opinions and behavior, product characteristics, or any other data that is available online.

However, the way in which information is obtained from online platforms is not as straightforward as in the previous case, requiring the implementation of automatic extraction mechanisms. Many platforms offer Application Programming Interfaces (APIs) that enable the retrieval of structured data in real-time. However, when information is not available through APIs, but still accessible on the online website, a process known as *web-scraping* is used, involving the use of scripts to extract data by parsing the website HTML.

Private datasets

It is important to notice that apart from the information that everyone can access through the Internet, there exists a large volume of data that remains private for a certain company or institution. As an example, private companies count on vast amounts of information regarding their transactional logs, customer records, or any other operational data included in their Enterprise Resource Planning (ERP) tool. In this way, by the exploration of their internal data, IT-driven companies can implement ML models and other advanced technologies to continuously monitor and enhance their capabilities.

Self-extracted data

Previous data sources represent scenarios in which existing information can be adjusted to fit into the scope of the study. However, it may not always be the case, requiring the construction of the dataset that will be fed into the ML algorithms for their training. This is commonly the case for the analysis of physical studies or the monitoring of processes, for which measurement tools, sensors, or IoT devices may be required for the recording of inputs or signals that will subsequently constitute the training dataset. As well, although much more labor-intensive, data may be manually recorded; as in the case of the collection of user opinion through surveys or interviews.

3.3. Stage 3: Data Pre-processing

Although there is a wide range of platforms or mediums from which data can be obtained, the reality is that structured, clean, and consistent databases are not common today. Instead, data is generally full of misalignment, noise, or missing values, which can reduce the effectiveness of predictive algorithms and lead to inaccurate systems. Consequently, data scientists take as mandatory the realization a data sanity check before starting the training of the algorithms. This initial step normally consumes around 80% of the time of the total model implementation [94].

Data pre-processing refers to the series of transformations applied to the dataset before providing it to the learning algorithm, intending to convert raw data into a processed and structured format for its analysis [45]. The pre-processing stage highly depends on the initial quality of the data to be treated, requiring a previous observation of the information to subsequently apply the mechanisms needed.

This section will cover the key aspects to consider for the proper pre-processing of the data, presenting a series of techniques that will allow the treatment of inconsistencies, the exploration of the data, and its transformation to ensure the most accurate learning of the algorithm.

3.3.1. Preliminary Exploration and Inspection

Before starting the realization of any cleaning or transformation steps, it is essential to perform a general exploration of the data under scope. This first observation will provide an overview of the information contained within the dataset, allowing the performance of a first data quality assessment and the identification of opportunities and constraints for the development of the work. Three main activities can be highlighted for a proper initial inspection of data:

Understanding of dataset size, rows, and columns

The size of a dataset will be determined by the number of rows and columns. While rows indicate the volume of available data (records), columns represent the dimensionality of the information (features). Both a small and a large number of rows and columns can directly impact model performance. Small datasets may not be easy to generalize or extrapolate to the prediction of new input data, and on the other hand, large datasets may degrade the model performance due to an increased computational cost that does not necessarily improve performance.

During this step, it is also crucial to understand what is represented by each of the rows (instances recorded) and which information is associated with them in their corresponding columns.

Understanding of data types

In programming languages, the type of data is an attribute that is associated with any variable, indicating to the system how to interpret this piece of information. When a variable is associated with a given data type, the valid operations allowed are limited to the ones associated with that specific data type. In those cases, a run error will occur if a value of the wrong category is introduced or if a not-allowed operation is performed [64].

The criticality of data types is even higher when analyzing information and implementing machine learning algorithms. It is quite important to ensure that the system properly identifies the correct type of data stored in each of the variables, as different pre-processing techniques will be required depending on the data category. Additionally, as will be explored in the next sections, some machine learning algorithms can not handle all types of data, for which additional transformations may be required. Figure 3.1 provides an overview of the typical data types that can be encountered in ML problems.

Preliminary identification of potential data issues

Once the size, features, and data types are identified and understood, the final phase of the initial exploration primarily involves a comprehensive examination of the raw data. This process can be conducted by focusing on a selection of representative samples in the case of large datasets. Its purpose is to gain initial insights into the unprocessed data structure, its formats, potential inconsistencies, empty cells, or any other anomaly that could impact the proper training of the algorithm.

Any inconsistencies identified during this stage will be systematically addressed in the subsequent steps, which focus on a thorough and meticulous data-cleaning process to ensure the reliability and integrity of the dataset.

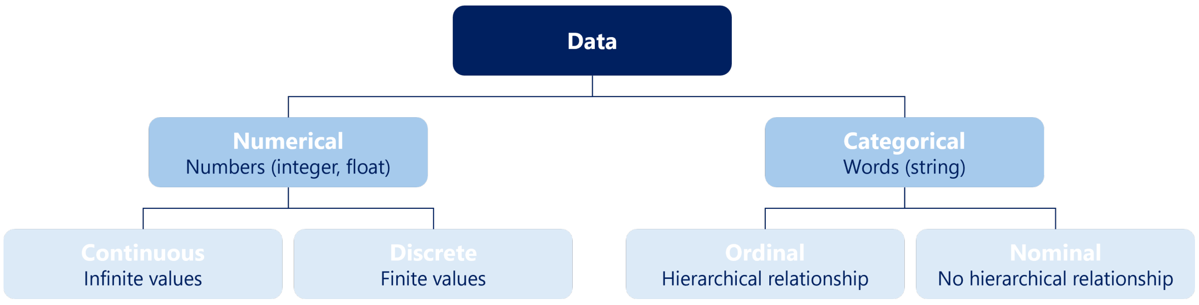


Figure 3.1: Overview of common statistical data types used in machine learning applications.

3.3.2. General Data-Cleaning and Structuring

Data cleaning, also known as *scrubbing*, aims to identify and correct any errors found in the data, as well as optimize the dataset by the removal of irrelevant data or duplicates. This process is intended to provide a first sanitation of the information by the realization of a series of minor tasks, which will facilitate the implementation of major feature pre-processing techniques and will enhance data analysis [70].

The most relevant elementary data-cleaning tasks can be found enumerated below [102], [112]:

1. Removal of irrelevant data:
   Normally, the initial source of information that will be used to train a machine learning algorithm is not designed for the specific objective of the work. Indeed, it is quite common to encounter extensive datasets with a high number of records or attributes, being only a part of them meaningful for the development of the study. Including non-relevant information can not only slow down processing time but also induce misleading tendencies during learning. As an illustration, a study about technology consumption after the big boom in the 2000s does not need to include information from previous years.

   Additionally, note that some features can be removed when they do not provide specific information about a record; this is the case of having a feature with the same value for all instances. Similarly, when the value of a given attribute is completely different for each of the records, no relationships can be acknowledged by the algorithm, making it possible to delete these columns for improved efficiency.

2. Removal of duplicates:
   Duplicates, also referred to as dupes, occur in datasets when an identical record with the same set of features appears more than once. Duplicated data is normally produced due to some kind of overlapping during the collection of information, for example, when two different sources are blended, containing some identical records, or by the submission of redundant forms.

   As in the previous case, the presence of duplicates implies a larger consumption of processing resources, resulting in a reduction of performance.

3. Standardization of values:
   This step is based on the identification of errors or inconsistencies in the information. It is not uncommon to encounter some misleading categories, formats, or scales in datasets, especially when data is manually inputted or different sources are combined. The most common inconsistencies to be considered are itemized below:

   - Typographic errors, capitalization, extra-white spaces, or abbreviations related to the spelling of a categorical value.
   - Usage of different measurement scales, for example, introducing a distance measurement both in meters and kilometers.
   - Formatting inconsistencies, which may affect dates, currencies, digits, and any other structure within attributes.

- Not known data not identified as missing values and including *"0"*, *"N/A"*, *"null"*, *"none"* values or any other similar indicator. In this situation, the best solution is to remove this incomplete data and try to estimate null values during subsequent steps.

As may be already deducted, these format misalignments can easily induce the double-counting of the same categories or the incorrect comparison of the data, hindering proper decision-making and distorting analysis.

4. Data organization:
Finally, the last main data-cleaning step is related to the proper structuring of the information. Some features may present multiple pieces of data representing different concepts. For example, the grades of several exams separated by commas within the same cell (i.e., [8, 7, 10]). In these cases, it needs to be evaluated if this organization could affect the correct training of the model and implement the proper amendments if needed. In the case of the example, the initial column could be segregated into three different columns, each of which representing a single exam grade.

3.3.3. Exploratory Data Analysis

After a first general cleaning and the standardization of attributes and their values, the next step is to work on the complete understanding of the dataset. This detailed analysis of the data is performed through the so-called *Exploratory Data Analysis (EDA)*.

The EDA allows the full understanding of the content of the data, identifies relationships among variables, discovers patterns, and provides the proper insights on features to guide the subsequent steps in the implementation of the algorithm.

With these objectives in mind, three different scales of data analysis can be distinguished [29]:

- Univariate analysis: Involving the exploration of the attributes in the dataset in their own. In this way, the analysis focuses solely on a single feature, without analyzing relationships with other variables. Univariate analysis is, therefore, the simplest method for data observation.
- Bivariate analysis: In this case, the exploration of data involves two different attributes, having as its primary objective the understanding of associations between variables.
- Multivariate analysis: When relationships need to be found within three or more variables, then a multivariate analysis needs to be followed. In this case, more intricate associations appear within data, normally corresponding to patterns not easily visible when variables are individually analyzed. The methods to perform this analysis are more complex than in previous cases and highly depend on the specific objective of the study.

As will be explored in the next paragraphs, there exists a large number of methods, techniques, and tools to perform these analyses, all of them based on statistical or visualization principles.

Statistical Description of Data

When datasets contain numeric variables, the most straightforward method to inspect their attributes is statistical analysis. With this purpose, the most common concepts to explore during the statistical evaluation are the measurements of the *central tendency* and *variability*, [71] [10].

The measurements of central tendency correspond to those parameters that provide information about the central or most typical values within the data, allowing the identification of the range where most of the observations are located. There exist three different metrics that define central tendency:

- Mean: Corresponding to the arithmetic average of all the observations.
- Median: Referring to the middle value when data is arranged.
- Mode: Being simply defined as the most common value in the distribution.

On the other hand, variability measurements indicate the extent to which data points deviate from the central tendency, providing details about the spread and dispersion of the values. In this case, the relevant metrics are:

- Range: Defined as the difference between the largest and smallest values.
- Variance and Standard Deviation: These parameters are highly correlated, while the variance corresponds to the average of the squared differences from the mean, the standard deviation is the square root of the variance. These metrics quantify the amount of value dispersion in a dataset, understood as the deviation from the mean [36].
- Interquartile Range (IQR): For the definition of this statistical parameter, it is important to previously define the concept of *percentile* [8]. In general terms, percentiles are the values that divide an ordered dataset into 100 equal portions. Thus, a percentile represents the value below which a specified percentage of the data set lies. As an example, the value of the percentile 45 is larger than the 45% of the observations but smaller than the rest. Having understood the definition of percentiles, the IQR is defined as the difference between the third (Q_3) and the first (Q_1) quartiles, corresponding to the 75% and 25% percentiles, respectively [18].

In the presence of extreme values in the data, some of these parameters can be highly affected. This is the case of the mean and the range, which present a high sensitivity towards outliers. In contrast, the median, mode, and the IQR are more robust, and the influence of extreme values is not that relevant. Additionally, as will be deeply analyzed in later sections, the standard deviation and the interquartile range can be used for the identification of these extreme values.

The exploration of all of these statistical parameters results in the identification of the type of distribution followed by the data, a fundamental aspect in the implementation of machine learning algorithms. As will be explained in later sections, some algorithms work under the assumption of data following a specific distribution. Additionally, some preprocessing techniques, such as the treatment of outliers or the handling of unbalanced data, are quite dependent on how the data is arranged.

There exists a large variety of possible data distributions, such as the binomial or hypergeometric for discrete variables, or the uniform or exponential for continuous ones. However, it is the *continuous normal distribution* the one which represents one of the most important concepts in statistics and the backbone for machine learning algorithms [9].

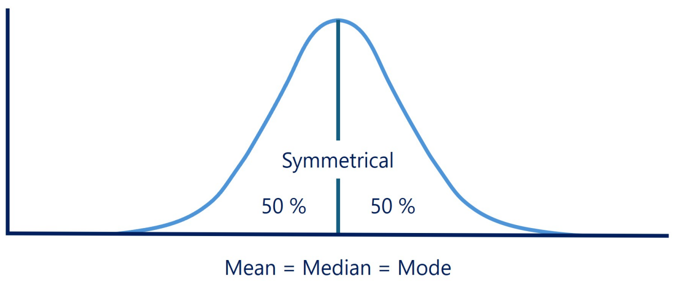


Figure 3.2: Graphical representation of normally distributed data, showing symmetry around its mean, which has equal value to its median and mode.

An attribute is normally distributed when values are perfectly symmetric around its mean, and therefore, have the exact same values for its mean, median, and mode. When visually represented, normal distributions show a characteristic bell-shaped curve, with most of the values clustered close to the mean and decreasing probability as moving away from the center. Because of its symmetric behavior, this type of distribution can be described only by two of the previously described metrics: the mean and the standard deviation [99].

The relevance of the normal distribution, also named as Gaussian, is justified by the *Central Limit Theorem (CLT)*, which states that independently of the original population, the sample means will converge to a normal distribution as the sample size gets larger, explaining its extensive presence in nature and relevant scientific fields.

These distributions are commonly used as a default assumption in several modeling applications because of their simplicity, understanding, and broad applicability, particularly useful for implementing predictive algorithms or computing the likelihood of certain events. For this reason, in some cases, by the application of some transformation methods, data is treated to fall into a normal distribution.

Because of the relevance of Gaussian distributions, there exist two additional statistical metrics to specifically quantify the degree of deviation of the data from a normal distribution [118]:

- Skewness: This parameter evaluates the asymmetry of a distribution, identifying the grade at which data is accumulated at a given interval of its range. If the value of this metric is positive, it indicates that most of the values are shifted to the left, having a distribution with a long tail at the right. On the other hand, a negative value will be obtained for left-skewed distributions (i.e., most of the values are accumulated at the right).

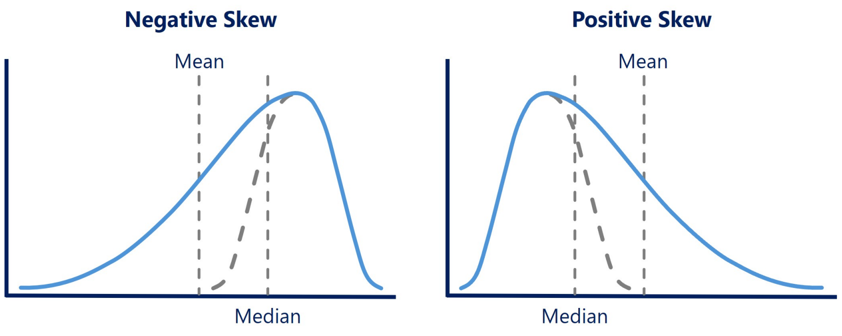


Figure 3.3: Deviation from normal distribution based on the sign of the skewness metric: Left-skewed distributions for negative values and right-skewed distributions for positive values.

- Kurtoise: In this case, the *tailedness* and *peakedness* of the distribution are compared to that of a normal one, being a consistent indicator of the presence of outliers. The value of this variable is equal to 3 when the data is normally distributed (mesokurtic distribution). If a different value is obtained, two different scenarios are possible: Values greater than 3 indicate that the peak of the curve will be accentuated with heavier tails, meaning that extreme values are frequent (leptokurtic distribution). On the contrary, if a value smaller than 3 is obtained, flatter peaks and lighter tails will characterize the distribution, meaning that fewer extreme values than in a normal distribution are present (platykurtic distribution).

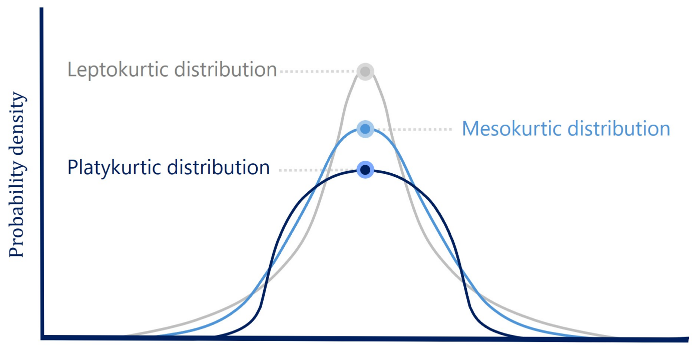


Figure 3.4: Distribution variation based on Kurtosis value: Leptokurtic, Mesokurtic, and Platykurtic distributions.

Although normality is an assumption for certain machine learning algorithms, it is not a requirement for data to be normally distributed. The normalization of the information can facilitate the work for some algorithms, such as Linear Regression, Logistic Regression, or Gaussian Naive Bayes; however, they can also be effective if only model errors are Gaussian [9]. A more detailed exploration of the effect of data distribution on the performance of the model will be provided in Sec. 3.4.1.

Data Visualization

As previously covered, the statistical study of the numeric variables in a dataset is a fundamental step in the exploration of the data. However, the study of the central and variability metrics presents some limitations as it provides some figures that can be used to get a general overview of the data behavior, but a more detailed understanding of the data distribution may be required. Additionally, statistical analysis is only applicable to numeric attributes, not being able to get insights from categorical variables. For this reason, the visual exploration of data is a complementary step during the EDA.

Data visualization consists of the conversion of information into visual representations, including charts, graphs, and maps. Its main objective is to represent in a visual format the distribution of values within a given attribute, for both numeric and categorical data types. Different visualization techniques can be used depending on the different types of data and the analysis objectives [79], [16], [92].

- For univariate analysis, numeric variables can be studied mainly by two graphics: histograms and line charts. Histograms are one of the most relevant representations as they allow the identification of the type of distribution and its comparison to the Gaussian one. On the other hand, line charts are useful for the display of time trends. When exploring categorical variables, bar and pie charts are the most common visualization techniques used to represent categories or groups and their proportions.

- The study of the relationship between two variables, the bivariate analysis, can be divided into three big groups:

    1. Numeric vs. Numeric: Normally addressed by the use of scatter plots, where attributes are represented in the x and y-axis of the graph and samples are represented as points.
    2. Numeric vs. Categorical: In this case, it is possible to use both violin and swarm plots for the visualization of the density of the data. Although less representative than previously explained methods, these charts could also be used for the categorical univariate analysis.
    3. Categorical vs. Categorical: In this instance, the most common practices make use of stacked bar charts or clustered bar plots for the study of their relative distribution.

    Additionally, one of the most relevant representations to study the relationship between a couple of variables is the correlation matrix. Correlation matrices, also known as heatmaps, can quantify the relationship between each pair of variables in a dataset [82].

    The construction of these matrices is quite straightforward when the dataset is composed of numeric features, using the *Pearson correlation coefficient*, which measures the linear relationship between the variables. This parameter ranges from 1, when there exists a strong positive linear relationship, to -1, when the relationship is strongly negative. A value of 0 indicates no association.

    In the case that categorical variables are included, a more complex approach is established. For measuring the relationship between two categorical variables, the *Cramer's V method* can be used, providing association values in the range [0,1], 0 indicating no association and 1 representing a perfect association. This method is based on the *Chi-Square Test* [1] and is symmetrical to the Pearson coefficient.

    However, the correlation between a numeric and categorical variable is not that simple; a specific technique exists for binary categorical variables called the *Point-Biserial method*, but if that is not the case, the encoding of the categorical variable may be required for the computation of the Pearson coefficient.

[1]The Chi-Square test is a statistical method used to figure out if categorical data differs from what should be expected, two deviations of this method are possible: the chi-square test of goodness of fit, which analysis the distribution of a single variable, and the chi-square test of independence, focused on the association between two variables [106].

Linear Pearson Coefficient	**Cramer's V Coefficient**	**Point-Biserial Coefficient**
$R = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2 \sum_{i=1}^{n}(y_i-\bar{y})^2}}$	$V = \sqrt{\frac{\chi^2/n}{\min(k-1,r-1)}}$	$PB = \frac{\bar{x}-\bar{y}}{\sigma} \times \sqrt{\frac{n_x n_y}{n^2}}$
x_i : Value of predictor variable y_i : Value of target variable $\bar{x}$: Mean of predictor variable $\bar{y}$: Mean of target variable n : No. of samples	χ : Chi-squared coefficient n : No. of samples k : No. of columns r : No. of rows	$\bar{x}$: Mean of predictor variable $\bar{y}$: Mean of target variable σ : Standard deviation n_x : No. samples of predictor variable n_y : No. samples of target variable n : Total no. of samples

Table 3.1: Mathematical formulation for the different possibilities of correlation coefficients depending on the nature of the data: Linear Pearson coefficient for numeric variables, Cramer's V Coefficient for categorical variables, and Point-Biserial Coefficient for binomial categorical and numeric variables.

- Finally, the analysis of more than two variables increases in complexity because of representation limitations. Although there exist different complex techniques, exploratory methods can be implemented by the derivation of previously mentioned visualizations by comparing groups or clustering information, as seen in ridge plots.

3.3.4. Outliers Treatment

The handling of outliers corresponds to another fundamental practice for any data science or machine learning project. In general terms, outliers can be understood as those samples contained in a dataset that are particularly detached from the rest of the data, comprehending this separation as the presence of extreme values on one or more of their attributes. Outliers can be easily identified by some standardized techniques. Some of the most common methods for outlier identification are described below [116], [44] :

1. Visualization method:
   This is the simplest technique, although not always the most effective one. It simply involves the generation of graphics for the attributes in the dataset and the identification of outliers by means of a simple visual inspection, taking advantage of the charts explored in the previous section. However, visualization methods do not always provide good results, as some data distributions can not be easily identified, for which additional statistical tools may be required.

2. Z-score or Standard Deviation method:
   Whenever data is found to follow a normal (Gaussian) distribution, the standard deviation method could be suitable for finding its outliers. As previously explored, normal distributions are characterized by the mean (μ) and the standard deviation (σ). By knowing the values of these metrics, and making use of the *empirical rule*, or the *68-95-99.7 rule*, a series of thresholds can be established defining where most of the values lie:

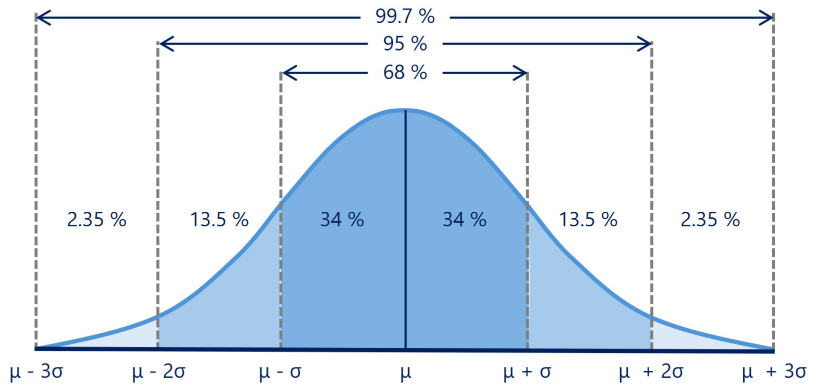


Figure 3.5: Percentages of samples distributed within one, two, and three standard deviations from the mean for normally distributed data, following the empirical rule.

- The 68% of the values are within one standard deviation with respect to the mean.
- The 95% of the values are within two standard deviations with respect to the mean.
- The 99.7% of the values are within three standard deviations with respect the mean.

According to this segmentation, the most common criterion is to use the $\mu \pm 3\sigma$ range as the interval where the majority of the values are included, identifying as outliers those participants outside this limit.

3. Interquartile Range method: Unlike the previous method, the interquartile range approach is applicable when data is not normally distributed. Recalling the concept of percentile explained in previous paragraphs, the IQR is defined as the difference between the third and first quartile, which defines the limits where the 50% of the data is included. Similarly to previously done with the standard deviation, the IQR is now used to define the upper and lower limits outside which outliers are located:

   - Lower limit: $Q_1 - m\ x\ IQR$
   - Upper limit: $Q_2\ +\ m\ x\ IQR$

   Where m normally takes a value equal to 1.5. This method can be easily visualized by the use of box plots. As can be observed in Fig. 3.6, boxplots consist of a box whose limits correspond to the IQR of the variable, and the line inside represents the median. Additionally, the whiskers extend a distance of $m\ x\ IQR$ from the first and third quartile, leaving outside these limits the samples that are identified as outliers [105]:

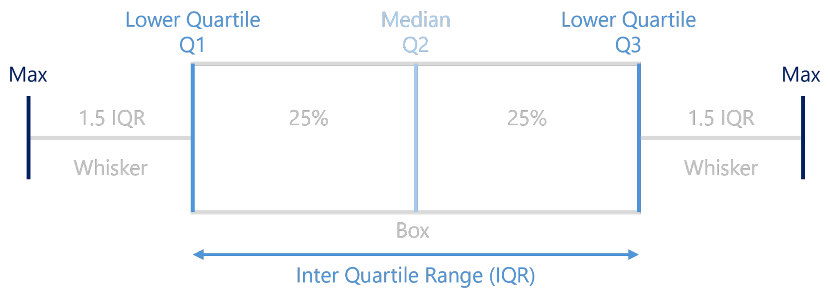


Figure 3.6: Representation of box plot structure with no skewness. If some skewness is detected within the represented variable, the median will be moved from the center of the box.

Apart from these three main methods, there exist other advanced techniques for the detection of outliers [88]:

- Mahalanobis Distance method: This technique also corresponds to a statistical approach, however, this case is not based on the definition of data limits. Instead, the Mahalanobis distance measures the distance of a data point from the mean of the distribution, taking into account variable correlations by the use of the covariance matrix. Points separated by a distance larger than a threshold considered for this distance are identified as outliers.

$$D_i = \sqrt{(x_i - \mu)^T S^{-1} (x_i - \mu)} \tag{3.1}$$

  Where x_i is the attribute value, μ is the mean of the variable, and S^{-1} is the inverse of the covariance matrix.

- Local Outlier Factor (LOF): This algorithm aims to examine the local neighborhood of each data point within the dataset to explore its similarity with other points. The LOF computes and compares the density of an observation to that of its neighbors, identifying as outliers those presenting a density lower than expected.

- Isolation Forest: Corresponding to an unsupervised machine learning algorithm, isolation forest involves the partitioning of data points until the observation is isolated. This method records the number of splits required to isolate an observation, known as the *path length.* As outliers are isolated more rapidly than normal points, the smaller the path length, the higher the probability of being an outlier.

- One-Class Support Vector Machine (OC-SVM): Similarly to the previous case, OC-SVM also corresponds to an unsupervised machine learning method that works by finding a hyperplane or a hypersphere that best separates data points. Its objective is to maximize the margin between the observations and the hypersurface, identifying the boundary that encompasses the large majority of data points, being those that fall outside of the boundary considered as outliers.

Normally, the presence of outliers within datasets is due to two fundamental causes: errors produced during the capture of the data or samples with some of their attributes presenting values that are different from the rest. In the first case, the record should be removed to avoid the inclusion of inconsistent data into the learning algorithm. However, when an extreme value is present and it is correctly describing samples, two scenarios are possible:

- Anomalies can be kept if they correspond to feasible values that, although not common, could be interesting for the model to learn to process samples out of the normal range. In fact, the learning from outliers is the key concept behind diagnostic systems used in the health field.
- Outliers should be treated if, although they are a correct representation of the data, they do not correspond to representative instances for the objectives of the study or the scope of its application, which could potentially degrade the performance of the algorithms.

This treatment of outliers can be performed through different approaches. The simplest one is the direct deletion/trimming of those outlier observations; however, if they correspond to some meaningful data points, information can be lost. Instead, outliers can be handled by the use of capping methods, which consist of modifying outliers with the upper or lower values of the attribute. Additionally, extreme values can be substituted by the mean value, with a so-called winsorizing method. Note that in those cases, although data is not deleted, it is modified, which can distort subsequent analysis [117].

3.3.5. Handling of Missing Values

Missing values found in datasets are one of the most common occurrences during the data pre-processing process, normally interpreted as *NaN* by the programming language. These null values can be associated with errors produced during data collection, including malfunctions of recording sensors or information loss due to the incorrect management of the data.

The handling of these empty values is a critical step for the proper pre-processing and consolidation of the data that will be feeding AI models; in fact, the majority of machine learning algorithms can not work with the presence of missing values.

Although there exists a large number of methods to manage missing values, no common solution can be applied for all scenarios, but the type of the problem needs to be studied to determine the most suitable approach, as selecting the incorrect method could provide biased results, highly damaging the performance of the algorithm.

When addressing missing values, the first step is to understand the different types of cells that are missing in the concerned database, facilitating the identification of the appropriate strategies required to address them. An overview of the common classification for missing values is provided below [100]:

- Missing Completely At Random (MCAR): Empty cells that can not be identified with any pattern within a certain variable; they occur randomly and are not related to any other variables in the dataset.
- Missing At Random (MAR): In this case, missing values are not randomly distributed, but a pattern can be observed and related to other variables in the dataset.
- Missing Not At Random (MNAR): This occurs when empty cells are not random, but differently from MAR, missing values can not be related to any other observed attribute.

While the first two types of missing data, MCAR and MAR, both effects can be neglected as empty cells do not differ from what is already inferred with another cell, MNAR is referred to as non-negligible as these missing values follow a pattern that can not be easily identified.

Once missing data types are identified, the next step is to identify the most suitable technique to deal with them. In general terms, two main strategies can be identified for the treatment of missing values: On the one hand, directly remove those observations or attributes that present some null values, and on the other hand, complete missing inputs with a value based on a reasonable estimate. A more detailed description of these approaches and the methods found in each of these categories can be found below [100], [114], [35], [110]:

Deletion

1. Listwise deletion: This supposes the removal of any observation (participants) who present missing values in any of their attributes within the dataset, only retaining observations with complete information. Although this is one of the simplest techniques for the treatment of null data, its implementation could lead to a considerable reduction of the dataset when the number of missing values is high. Moreover, it could result in a biased sample of the information.

2. Pairwise deletion: For this case of deletion method, only features missing from a given observation are removed, conserving all known information for the different records. This technique results in an uneven size of the dataset for each of the attributes, which may have some limitations during subsequent analysis. For example, statistical studies of variables will be based on different subsets of the data, and some algorithms may not allow different sizes. Additionally, multivariable analysis, such as correlation, will only take those observations where data is complete for the variables studied, reducing the number of participants.

3. Observations/Attributes deletion: This technique is based on the deletion of an observation or column when they present a large number of null values (sparse datasets), and therefore, they do not provide enough relevant information for the study. In this case, only participants and attributes with a low amount of information are removed, keeping others with a manageable number of missing values, which will be tried to be estimated employing other techniques, such as imputation described below.

Imputation

1. Fixed-value: This is the simplest imputation method that can be used to remove the presence of null values. It involves the provision of a given value to those features where information is missing, such as *No information* or *NA*, so although these values do not provide any information about observations, analysis can be performed having labels that identify those features for which information was not available.

2. Imputation with mean/median/mode: Numeric continuous columns presenting missing values can be completed with the mean or median of the non-null values, being a statistical approach for the treatment of missing values which can result in good estimates when values are normally or skewed distributed. On the other hand, null-values in categorical columns (both string and numeric) can be completed by the use of the most frequent category, the mode. This method can be efficient when the number of missing values is a small proportion; however, if a large number of empty values is present, the mean, median, or mode can cause a loss of variation in the data.

3. Forward fill and backward fill: In the case of the forward fill, missing values are completed by taking the value of the previous observation where the missing attribute was known. On the other hand, the backward fill utilizes the subsequent observation with the documented value. These methods present particularly useful applications to the treatment of missing values in time-series data.

4. Interpolation: Similarly to previous methods, interpolation takes advantage of adjacent data points to obtain an estimation for the missing value. It offers good results for time-series data or when values are expected to smoothly vary.

5. Hot-deck imputation: This imputation technique is based on the identification of one observation or a group of them that exhibit similarities to the record for which values are missing. In other words, missing values for a given participant are replaced with the known values from a so-called *donor*, which shares analogous attributes with the sample experiencing missing data.

6. Cold-deck imputation: This method follows a similar approach to the one described for the hot-deck imputation. However, differently from the previous method, cold-deck imputation replaces null values with similar cases from other sources of information corresponding to unrelated samples.
7. Model imputations: The methods presented until this point do not perform any detailed analysis of the correlation of the variables where values are missing with other attributes in the dataset. However, there exists a series of predictive models such as *Simple Regression*, *K-Nearest Neighbors*, or *Multivariate Imputation by Chained Equations (MICE)* -which are explained in later sections- that can detect data relationships, providing more accurate estimates for the missing values. These models are trained with only fully completed observations.

As may already be derived from the explanation of the different missing values treatment techniques, imputation methods provide the best results as they prevent the loss of data in comparison with the deletion methods. However, special care needs to be taken during their application to avoid the distortion of the data.

3.3.6. Feature Selection

The thorough exploration of the data performed during the previous pre-processing steps allows the identification of inconsistencies for which a series of procedures were presented to mitigate them. At this point in the pre-processing stage, the features within the dataset should be refined and prepared to advance into the implementation of the algorithm, making possible the properly selection of features for the problem under study.

In this way, feature selection corresponds to one of the last steps during the pre-processing of the data, which aims to reduce the number of features for the training of the algorithms. Its purpose is to identify those features that are the most relevant for the study, deleting insignificant, redundant, or corrupted ones [78].

The criticality of this step is directly associated with the performance of the model, as the proper selection of features can be translated into a better learning of the algorithm, increasing its accuracy and interpretability. In fact, machine learning models tend to struggle due to the presence of a high number of non-meaningful features, due to the computational cost associated with high-dimensional datasets, and the possible *overfitting* of the model [2].

There exist different techniques to address the selection of variables, which can be classified into three main categories [119], [41]:

1. Filter methods: In this case, the relationship of the feature with the target attribute is independently analyzed, normally using bivariate analysis through statistical tests. This allows the rapid identification of the most significant features, without high computational demands, although the interaction among features is difficult to acknowledge. The most commonly used techniques in this category encompass methods such as Chi-square tests or Pearson's correlation coefficients. In addition to these, some other methodologies can be found, such as the *Fisher's Score*, *Variance Threshold*, or *Mean Absolute Difference (MAD)*, which only retain features with a certain significant variance. Furthermore, there exists a powerful feature extraction technique known as *Principal Components Analysis (PCA)*, which creates new non-correlated attributes from the original ones, maximizing the capture of variance.

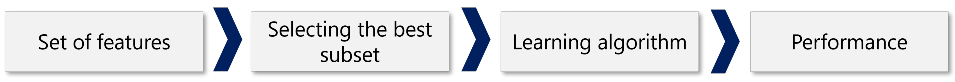


Figure 3.7: Pipeline scheme for the implementation of filter feature selection methods.

2. Wrapper methods: These methods implement the feature selection process based on the particular algorithm to be developed, receiving the name of *greedy algorithms*. They work by training the machine learning model on all possible combinations of features and evaluating its performance after each iteration. After evaluating all the possible combinations of variables, the best subset of data is identified

[2]Overfitting corresponds to an undesirable behavior of algorithms which occurs when they provide precise predictions for the training data but not for the newly introduced observations [7]. It is caused by the tight adjustment of the model to the training data, losing interpretability capabilities.

according to a specified criterion. These approaches provide better results than filter ones, being able to capture more complex dependencies and interactions, however, their computational expense is much more significant.

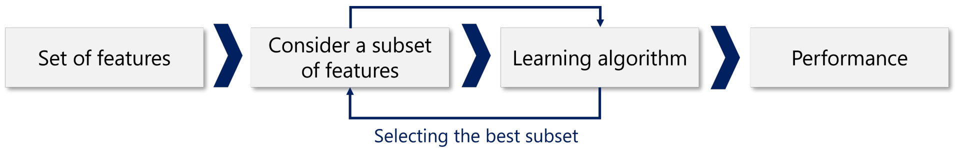


Figure 3.8: Pipeline scheme for the implementation of wrapper feature selection methods.

Among these wrapper methods, the most popular techniques are: the *Forward Selection*, which starts the iterative process with an empty dataset and keeps adding features if relevant for the performance; the *Backward Elimination*, which follows the contrary approach than in previous case, starting with the complete dataset and removing poorly significant features in each iteration; and finally the *Recursive Features Elimination (RFE)*, based on the recursive deletion of features by looking for the smaller set of features possible.

3. Embedded methods: In this case, the feature selection procedure is performed during the own training of the model, taking advantage of the internal properties of the models and providing stronger and more scalable results than in wrapper methods. However, the feature selection is again specified within a given algorithm, and results may not work when applied to other models. A detailed description of specific techniques for these methods will be provided during the exploration of the different machine learning algorithms in Sec. 3.4.

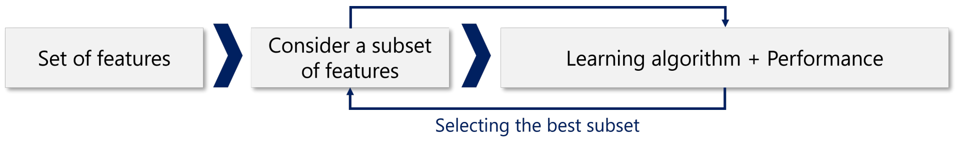

Set of features
Consider a subset of features
Learning algorithm + Performance
Selecting the best subset

Figure 3.9: Pipeline scheme for the implementation of embedded feature selection methods.

It is important to notice that some feature selection may already happen during previous pre-processing steps, when the quality of the data of certain variables is found to be very low. For example, the deletion of variables presenting a high number of missing values is a common technique.

3.3.7. Feature Engineering: Further Feature Transformations for Model Adjustment

Previous phases were oriented on the preparation of the data, ensuring its consistency, understanding, and explorability. Once the relevant features for the study are identified, the objective of this subsequent *Feature Engineering* step is more focused on the manipulation and transformation of variables to ensure that data is provided in a good format for the model, allowing its complete exploitation. This phase is normally understood as a linking stage between data pre-processing and the model's implementation. Indeed, there exist some feature engineering techniques that are quite model-dependent, requiring their study together with the implementation of the algorithms [67].

The next paragraphs will deep-dive into the different techniques that can be encompassed in the *Feature Engineering* step, such as the encoding of categorical variables, the scaling of data, or its transformation.

Encoding of Categorical Variables

The encoding of categorical variables involves their conversion into a numerical format. This step is critical when dealing with datasets presenting categorical features because most of the machine learning algorithms only support numeric inputs. Additionally, encoding has been found to significantly impact the model performance, requiring the implementation of proper techniques for the obtention of accurate predictions [83].

Before exploring the different methods available for the encoding of categorical variables, it is important to understand the two main groups in which they are classified, which were already introduced in Sec. 3.1:

- Nominal variables, including categories which do not present an inherent order. An example of this type could be the cities in a state.
- Ordinal attributes, representing categories which can be logically ordered or ranked. This could be the case of satisfaction evaluations (e.g., low, intermediate, high).

Depending on the type of categorical variables constituting the dataset under concern, different encoding techniques will be implemented to achieve the largest model's learning capacity. Below, the most prevalent encoding methods are examined, together with their main advantages and restrictions [83], [89], [49]:

1. Ordinal / Label Encoding:
   This is one of the most elementary methods, consisting of the assignment of a unique integer to each of the categories constituting the variable. This technique is suitable when categories follow a natural ordinal tendency, but if that is not the case, algorithms can interpret the attribute as having a hierarchy pattern, affecting the prediction.
2. One-Hot Encoding:
   Corresponding to the most popular technique for the encoding of categorical variables, this method generates a binary matrix where each column corresponds to a category of the feature, being composed by values of *1* indicating the position of the presence of a certain category, or *0* for its absence. In comparison with other procedures, the one-hot encoding presents the advantage of preventing the ordinal problem associated with some ordered patterns. However, as a column is added for each of the attribute's categories, its associated dimensionality increase can pose a considerable drawback for this method [50].
3. Binary Encoding:
   It involves the association of each category with a certain binary number. This method does not increase dimensionality as the one-hot encoding, however, it may give rise to challenges when distinctions between categories lack significance.
4. Count Encoding:
   This technique works by replacing the categorical value with its frequency in the attribute and is normally used when a large number of categories exist. Nevertheless, it may introduce bias into the model, as categories with high frequency can dominate the model.
5. Target Encoding:
   In this case, the categorical values are replaced by the mean/mode of the target variable for that particular category. This method corresponds to an advanced technique normally used when working with attributes with a high number of different categories, aiming to reduce dimensionality. However, it can cause overfitting if attributes do not present a relevant correlation with the target variable.

Data Scaling: Normalization and Standardization

Data scaling is the process of adapting the scales of the different numeric attributes to ensure that all values lie within a certain standardized range [68]. This process is of high relevance for datasets presenting features with different magnitudes or units, guaranteeing that a common scale is maintained while conserving the intrinsic relationships due to the variety of value ranges. In this way, scaling ensures a homogenized contribution for all the variables, avoiding the dominance of features of greater magnitude.

Additionally, it facilitates a more efficient convergence for gradient descent-based algorithms such as linear regression or artificial neural networks, where the magnitude of the values affects the step size of the descent, so scaled features will ensure consistent time steps. Similarly, the k-nearest neighbors and the support vector regression, which correspond to distance-based algorithms, are highly influenced by data scales as they use distances to analyze point similarity. On the contrary, tree-based models are quite insensitive to the scale of features, not requiring the scaling of their variables [115].

This adjustment of values within a specific scale is performed through two principal techniques: *Normalization* and *Standardization*. The main principles and differences of these methods can be observed below:

Normalization	**Standardization**
Scaled values fit into a specific range	Scaled values follow a certain distribution but are not limited into a specific range
Appropriate for non-Gaussian distributions	Appropriate for Gaussian distributions
High sensitivity to outliers	Low sensitivity to outliers
Maintains original data distribution	Changes original data distribution
Loses patterns between observations	Preserves patterns between observations

Table 3.2: Fundamental principles behind both normalization and standardization techniques.

Getting deeper on the exploration of scaling methods, different techniques can be implemented for the normalization and standardization of the data [81], [38]:

Normalization

- Min-Max Scaling: Scaled values are made to fit into a range from 0 to 1. This method corresponds to the most widespread normalization mechanism, especially providing good results for neural networks.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{3.2}$$

- Max-Abs Scaling: Similar to the previous technique, but in this case, only the absolute maximum is used during the scaling, making values range between -1 and 1. This method is useful when a feature contains both negative and positive values, and their sign wants to be retained.

$$x' = \frac{x}{max(abs(x))} \tag{3.3}$$

- Unit Vector Transformation: This method converts each of the observations in a dataset into a unitary vector (i.e., with length equal to 1) without losing the direction of its values. It is normally useful when working with distance-based models.

$$x' = \frac{x}{||x||} \qquad with \qquad ||x|| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2} \tag{3.4}$$

Standardization

- Z-Score Standardization: Being the most common standardization method due to its wide applicability, it works by individually centering data around the mean ($\mu = 0$) and establishing a standard deviation of 1 for the resulting scaled variable.

$$x' = \frac{x - \mu}{\sigma} \tag{3.5}$$

- Robust Scaling: Similar to the previous case, but in this case, the resulting values are centered around the median, and the interquartile range is used instead of the standard deviation. It provides good results for outlier-intensive datasets.

$$x' = \frac{x - median(x)}{IQR(x)} \tag{3.6}$$

- Quantile Transformation: This method transforms data to fit into a specific distribution, normally the Gaussian one. It may be difficult to implement and interpret, but it might provide good results when models based on normality assumptions are deployed, such as linear regression.

Feature Transformations

As will be deeply explored in the next sections, some regression algorithms may be affected by the lack of linearity between the independent variables and the target feature, or the absence of normality in the features' distributions [24]. Thus, feature transformation can be applied to raw data to obtain more linear relationships, stabilize variance, and minimize the effect of outliers. All of this contributes to the normalization of the variables and improves the performance of machine learning algorithms. Among the different transformation techniques, two main groups can be identified: *Function* and *Power* transformations. The next paragraphs are focused on the understanding of these techniques and the identification of their frame of applicability [80], [52].

Function Transformations

Function transformations are based on the modification of a feature's values by simply applying a mathematical transformation. Table 3.3 includes the most commonly used techniques for this procedure:

Function Transformation Technique	**Formula**	**Applicability**
Logarithmic Transformation	$x' = ln(x)$	Corresponding to one of the simplest transformations, it is used when a variable contains large and right-skewed data, reducing extreme values and obtaining a distribution closer to the Gaussian one. As a drawback, it can only be applied to positive values.
Reciprocal Transformation	$x' = 1/x$	Similarly to the previous case, this transformation scales values down but making originally small values more relevant. It is also used for right-skewed distributions and is not able to manage null values.
Square Transformation	$x' = x^2$	By squaring the values of the features, this transformation increases the magnitude of the data and provides only positive values, which results in a more symmetrical distribution. It is optimal for left-skewed data that does not linearly relate to the target variable, but instead, describes a curve.
Square Root Transformation	$x' = \sqrt{x}$	Although this transformation is weaker than the logarithmic one, it works similarly by reducing large values and resulting in more symmetric distributions for right-skewed data with the presence of outliers.

Table 3.3: Formula and applicability framework for common function transformation techniques.

However, other customized transformations could be applied for data presenting higher complexity on their distribution. In this way, based on the domain of data values, different functions can be applied to make the initial distribution more similar to a normal one, such as the *sin*, *cos*, *cube*, etc.

Power Transformations

Power Transformations correspond to more advanced techniques involving simply raising a feature's values to a certain power. The implementation of an iterative method automatically finds the value of the power (λ) until the most suitable transformation is found for the normalization of the data. With this purpose, two main methods can be identified: the *Box-Cox* and *Yeo-Johnson* transformations.

$$\text{Box-Cox:} \quad x' = \frac{x^{\lambda} - 1}{\lambda}, \quad \lambda \neq 0 \tag{3.7}$$

$$\text{Yeo-Johnson:} \quad x' = \begin{cases} \frac{(x+1)^{\lambda}-1}{\lambda}, & x \geq 0, \lambda \neq 0 \\ \frac{-(|x|+1)^{2-\lambda}+1}{2-\lambda}, & x < 0, \lambda \neq 2 \end{cases} \tag{3.8}$$

Both techniques allow the stabilization of the variance, making values closer to a normal distribution and being especially useful when other transformations do not perform efficiently. The main difference between these two techniques is that while Box-Cox requires values to be positive, the Yeo-Johnson transformation can work with both positive and negative figures.

3.4. Stage 4 & 5: Delving into Regression Algorithms (Training and Evaluation)

Marking the end of the pre-processing stage, this section will explore the remaining phases in the roadmap for the implementation of a machine learning algorithm. As stated in Sec. 2.3, after the preparation of the data that will be fed into the algorithm, the subsequent stages involve the training of the model, its evaluation, and optimization until reaching its desired performance. Because of the iterative character and dependency between these stages, it is convenient to study these phases in a parallel manner.

Previous phases have been written in a global way to provide a guide for the implementation of any standard type of machine learning algorithm. However, due to the wide area of knowledge that covers this section, the next paragraphs will be dedicated to *supervised regression* models, allowing the focus and limitation of the information to be covered and being aligned with the scope of this dissertation.

In this way, the main concepts defining regression models will be discussed, focusing on the following topics:

- Understanding of the different approaches available for the implementation of regression algorithms, paying special attention to the hyperparameters [3] that define them.
- Algorithm's evaluation metrics for the assessment of the model's performance.
- Optimization and adjustment techniques to enhance the algorithm's performance.

Additionally, during the development of these themes, some main challenges and limitations associated with the implementation of the models will be identified, highlighting specific considerations or procedures to be taken into account for the optimal performance of the models.

3.4.1. Regression Algorithms Approaches

There is no short list of algorithms used for regression analysis, emphasizing linear regression as the most extended model due to its simplicity and adaptability to different problems. However, other regression mechanisms exist, including complex ensemble methods or artificial neural networks, generating the need to understand the different regression approaches and their suitability for the data to be treated. For this reason, the main regression machine learning algorithms will be investigated in the next paragraphs.

Linear Regression Models

Corresponding to the most fundamental statistical method to find the dependency between two features, the *Simple Linear Regression* model assumes a linear relationship between a single independent variable (explanatory attribute, x), and the dependent variable (target response, y). As a result, a linear function for the two-dimensional sample points describes the mathematical equation behind this algorithm [75].

[3] The hyperparameters of a machine learning model correspond to a set of values which are fixed by data scientists before the training of the algorithms, and determine the behavior of the model during the learning process.

$$y = w_0 + w_1 x \tag{3.9}$$

Where w_0 represents the intercept with the y axis, and w_1 is the coefficient of the explanatory variable, which provides a clear explanation of the influence between the variables. Coefficients can be both positive and negative and illustrate the degree of change of the target response for every unitary modification of the explanatory attribute. In this way, the main objective of the linear regression method is to identify the *best-fitting line* through the training data, understood as the equation that reduces *offsets* or *residuals* to a minimum.

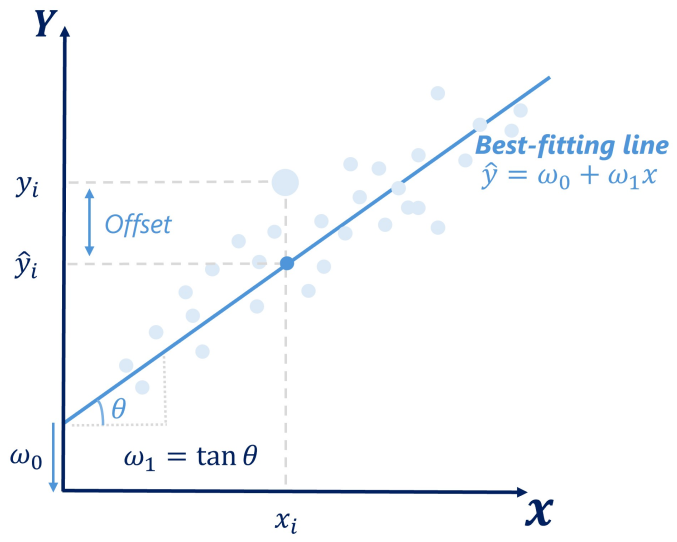


Figure 3.10: Graphical representation of the simple linear regression model, highlighting the best-fitting line, training data points, and offsets which indicate the deviation between actual values and those predicted by the model.

Its simplicity and large interpretability allow this approach to serve as the fundamental building block for more advanced models. Indeed, the simple linear regression can be easily generalized to include more than one explanatory variable, in the so-called *Multiple Linear Regression*:

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + ... + w_n x_n = \sum_{i=1}^{n} w_i x_i \tag{3.10}$$

In this case, a unique target response is intended to be predicted based on more than one explanatory attributes, taking into account that the features are assumed to not present any multicollinearity, meaning that the degree of correlation among the independent variables is very reduced or even null, allowing the study of the individual effect of each attribute [47].

For the implementation of a multiple linear regression, the features selection step plays a fundamental role, making essential the careful identification of the independent variables that will be included for the building of the model, especially for high-dimensional dataframes, as the inclusion of non-relevant or redundant features can result in the overfitting and degradation of the performance of the algorithm.

Under this context, there exist several regularization techniques that contribute to the optimization of linear regression models by handling feature selection, overfitting, and multicollinearity. These approaches work by adding a penalization term to regulate the magnitude of the linear coefficients. Depending on how the penalty term is introduced, three different regularization techniques can be identified [48]:

Lasso (Least Absolute Shrinkage and Selection Operator) Regularization

This model adds the so-called *L1* penalization (the sum of the absolute values of coefficients) to the cost function of the regression model, which is minimized by reducing some of the linear coefficients to zero. Thus, the model completely removes features that are not relevant for the prediction, only retaining the variables that can improve the model's performance. This method can present some limitations if the data includes a large number of intercorrelated variables, as the algorithm will only select one of those features, making possible the loss of relevant information.

$$\text{Lasso Cost Function} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha_L \sum_{j=1}^{m} |w_j| = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + L1 \tag{3.11}$$

Ridge Regularization

In this case, the penalization corresponds to a *L2* term (sum of the squared values of the coefficients), which, instead of vanishing the linear coefficients to zero, just reduces their magnitude. In this way, this regularization keeps all variables but reduces the influence of certain of them, increasing the stability of the model when high multicollinearity is present among the independent variables. This technique is especially useful when all features are valuable but their impact on the model needs to be limited.

$$\text{Ridge Cost Function} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha_R \sum_{j=1}^{m} w_j^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + L2 \tag{3.12}$$

Elastic Net

This technique results from a combination of *L1* (Lasso) and *L2* (Ridge) penalizations in a unique cost function. Thus, the elastic net method combines the advantages of previously explored penalization techniques, being able to efficiently select the optimum features for the model, while keeping stability and handling multicollinearity.

$$\text{Elastic Net Cost Function} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha_L \sum_{j=1}^{m} |w_j| + \alpha_R \sum_{j=1}^{m} w_j^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + L1 + L2 \tag{3.13}$$

As observed in previous equations, the cost functions that govern these techniques include a hyperparameter α which controls the quantity of regularization that is applied to the model. In the case that this parameter is equal to zero, no regularization will be applied, resulting in an algorithm that will behave as a standard linear regression. On the other hand, as this factor increases, the higher the effect of penalization in the model.

Moreover, in the case of the Elastic Net, an additional hyperparameter $l_1 = \frac{\alpha_L}{\alpha_R}$ is required to control the proportion of Lasso and Ridge penalizations:

- If $l_1 = 1$, the Elastic Net will behave as a Lasso penalization.
- If $l_1 = 0$, the Elastic Net will behave as a Ridge penalization.
- If l_1 is in the interval $(0, 1)$, a combination of both penalizations will be applied.

Tree-Based Models

Until this point, all algorithms explored were based on the determination of the line that best fits samples into the training data set, allowing the prediction of the value of a target response when new attributes are introduced. Contrary to these models, *Tree-Based* regression follows a non-linear approach capable of detecting intricate patterns in complex datasets.

The most elementary algorithm constituting the tree-based family is the standard *Decision Tree* model. This method is based on the identification of a smaller dataset within the data according to a set of decision rules derived from the input attributes. Normally, the optimum data division is obtained by minimizing the residuals of the predictions or the variance within each subgroup. In this way, a tree-like structure is established, repeating the division of each subset until a specified termination criterion is reached, such as the minimum number of samples in a leaf (*min_samples_leaf*), the maximum depth of the tree (*max_depth*), or the minimum number of samples in a node to allow their subdivision (*min_samples_split*) [39].

This process is illustrated in Fig. 3.11 where each diamond shape represents a condition to be assessed or a decision point, also known as an internal node in the tree-like structure. Depending on the outcome of the decision, data is connected through *branches* to new decision points until fulfilling the termination criteria into the corresponding *termination or leaf node* [76].

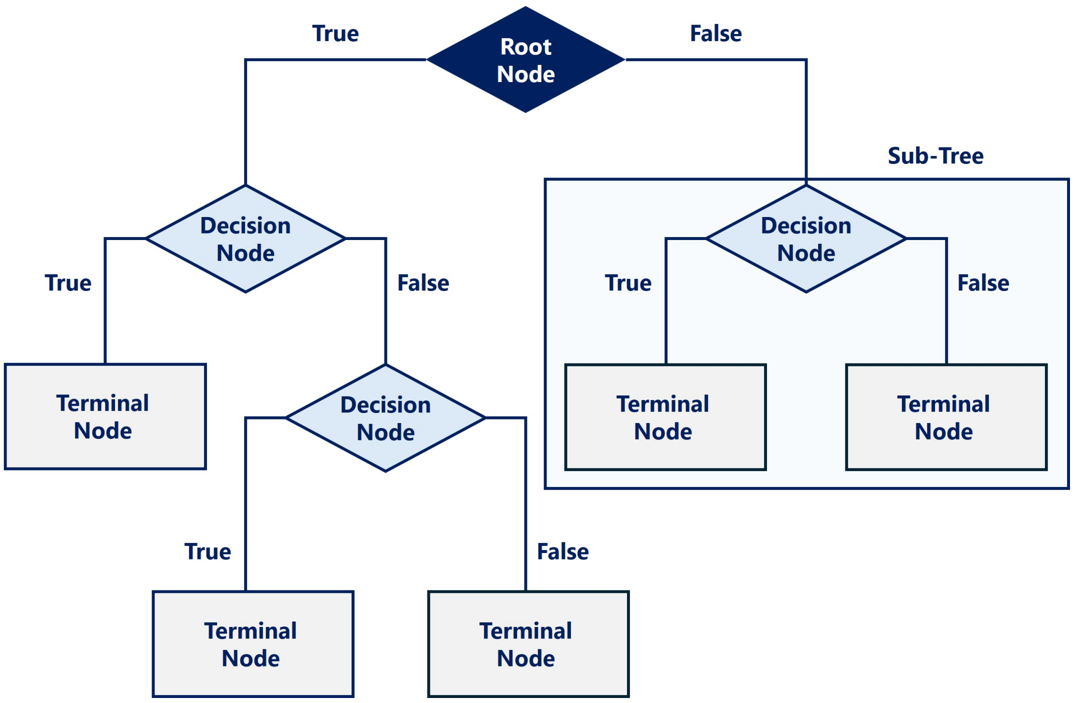


Figure 3.11: Diagram for the representation of a decision tree structure, illustrating data division at decision nodes until reaching predictions at termination nodes.

Once the decision tree is constructed based on the training data, the new inputs follow the decision path defined until arriving at its final subset, where the target variable is computed as the mean value of the samples in that subgroup.

Decision tree algorithms are easily implemented as they do not require complex data formats and transformations like other models. Additionally, they can be enhanced by the implementation of more advanced versions such as *Random Forest* or *Gradient Boosting* approaches.

Both random forest and gradient boosting correspond to the so-called *ensemble learning techniques* [40], understood as the combination of multiple *weak* or *base* models, allowing the construction of more robust and reliable algorithms. The combination of the models can be performed in two different ways:

- By independently training the models with different subsets of the data frame and subsequently averaging the predictions. This combination technique receives the name of *bagging ensemble* and constitutes the pillars for the construction of the random forest algorithm.
- On the other hand, the combination of models can be performed by a sequential training of the algorithms, allowing the amendments of inaccuracies from previous models. This technique receives the name of *boosting ensemble* and acts as the backbone for the gradient boosting method.

It is important to emphasize that these ensemble models can be applied to other machine learning approaches, distinct from the decision tree, following the same foundation. However, its applicability to tree-based models is much more frequent than for other algorithms, making random forest and gradient boosting gain traction as commonly employed models for regression problems. A more detailed description of these methods is provided in the next paragraphs.

Random Forest

Random forest involves the construction of different and independent decision trees, each of them being trained in parallel with a random subset of data, where decision points are defined based on a aleatory selection of variables, aiming to reduce correlation among the trees and increasing the diversity of the overall model [61]. When an observation wants to be predicted, its attributes are processed by the different trees, obtaining the final output as the mean of the results of the distinct trees.

Gradient Boosting (GBoost)

In this case, the sequential implementation of decision trees allows models to be trained, reducing the loss function in each pass by implementing gradient descent methods. With this objective, each model iteration evaluates the gradient of the loss function, understood as the residual between the prediction and the real observation, and a new model is trained to rectify inefficiencies from previous iterations [26]. The final prediction is obtained as the weighted sum of all particular tree predictions, where the weight factor α_{LR} is known as the *learning rate* and controls how each new model contributes to the final prediction [95].

$$\hat{y}_i^{(n+1)} = \hat{y}_i^n + \alpha_{LR} \cdot f_i(x_i) \tag{3.14}$$

In comparison with the conventional decision tree algorithm, both the random forest and gradient boosting present similar advantages. These algorithms increase accuracy and reduce overfitting, being especially relevant when facing high-complex dataframes. However, they present a larger complexity, making these models not so easily interpretable and requiring the adjustment of additional hyperparameters, such as the number of trees to be included (*n_estimators*) or the learning rate (α_{LR}) in the case of GBoost.

K-Nearest Neighbors Regression (KNN)

The *K-Nearest Neighbors Regression* corresponds to a non-parametric algorithm which uses distance proximity to identify similar data points to the one of the query. The model works by computing the distance between the newly introduced sample and the other observations in the training dataset employing the *Minkowski equation* [59]:

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|)^p \right)^{1/p} \tag{3.15}$$

As can be observed in the previous Eq. 3.15, this formula is characterized by a variable p defining the order of the equation and the type of Minkowski distance that the model will use. Depending on the value of this hyperparameter, different distances can be used, as can be observed in Tab. 3.4.

p value	**Distance**	**Relevant Considerations**
1	Manhattan	This distance highlights due to its robustness towards outliers.
2	Euclidean	It presents a high sensitivity to outliers, requiring the scaling of data.
>2	High-order distance	Useful if dimensions with larger differences want to be exaggerated, but may cause overfitting and a lack of generalization.

Table 3.4: Distance metrics based on p hyperparameter and relevant considerations about their impact on the model.

Additionally, a hyperparameter k needs to be defined by the developer to determine the number of neighbors that will be considered for the prediction of the target variable. Lower values of k (e.g., 1 or 3) may result in noisy predictions as the variance increases while the bias decreases, which may cause the overfitting of the model. On the other hand, larger values increase bias and reduce variance, generating more smoothed predictions [74].

Finally, the predicted value is computed as the average of the target responses of the k nearest neighbors selected. In some advanced refinement of the model, closer points can be considered to have a higher influence on the final prediction than those that are located further apart. This is performed by the addition of a supplementary hyperparameter *weights*, which defines the impact of each neighbor depending on its proximity to the query sample.

This method is highlighted due to its simplicity to implement, being particularly efficient for the processing of small datasets and the identification of non-linear features. However, it may offer limited capabilities for analyzing large sets of information, as an expensive computational cost is required to calculate the distance to each datapoint. Moreover, the k-nearest neighbor algorithm presents some sensitivity to the presence of irrelevant features and variations in scale, demanding the implementation of some normalization and dimensionality-reduction processes [22].

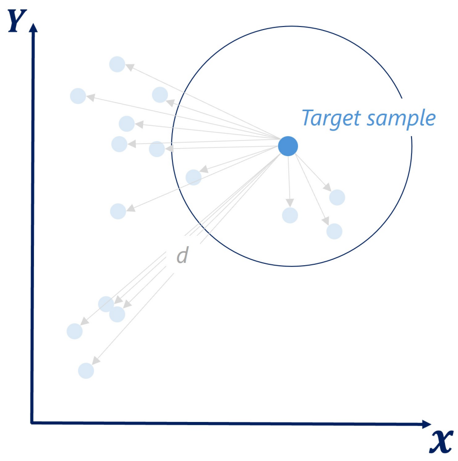


Figure 3.12: Graphical representation of a simple k-nearest neighbor algorithm for a 2-D dataset, representing distances between the target sample and training observations, and indicating the selection of the closest neighbors.

Support Vector Regression (SVR)

The *Support Vector Regression* corresponds to a regression model derived from the *Support Vector Machine (SVM)* algorithm used as a classification method. The SVM works by representing the training data points in a high-dimensional space, allowing the identification of a *hyperplane* that separates classes into regions as wide as possible. This algorithm seeks to maximize the orthogonal distance between the hyperplane and the closest data points of the different classes, minimizing errors during the classification task.

In the case of the *Support Vector Regression*, although following a similar approach to that of the SVM, the objective is to find a hyperplane that, instead of dividing the space per category, provides a function capable of predicting the relationship between the attributes and the target variable [113].

In this way, the SVR works by representing the points in the space and establishing a *decision boundary* as wide as possible, where the deviations between predicted and actual values are under a certain error margin. This is controlled by an epsilon-intensive loss function (ϵ), which allows errors in predictions within the boundary and penalizes those outside this range.

This model requires the definition of additional tuning hyperparameters, such as the regularization parameter (C), which establishes a trade-off between the maximization of the boundary and the minimization of the prediction error. Thus, a smaller value for this parameter provides the model with a wider boundary, permitting more errors, whereas larger values narrow the margin and restrict the errors allowed [86].

More advanced hyperparameters can be used to define different kernel functions for the hyperplane and optimize the model performance. These kernels can be linear, polynomial, or radial basis functions that are used to adapt the multidimensional space, allowing the modeling of non-linear and complex relationships [27]. In the case that the kernel selected is not linear, a variable γ can be used to control the influence of data training points.

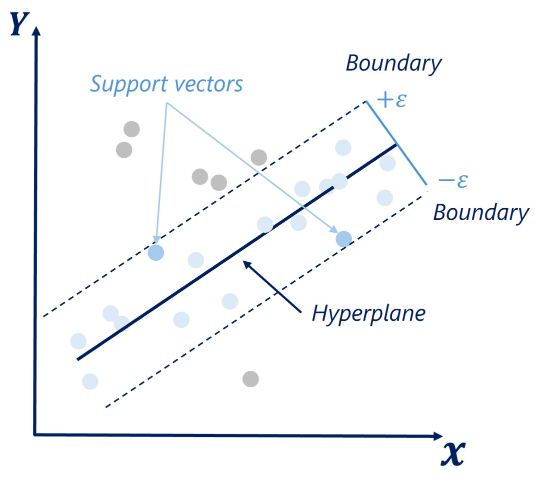


Figure 3.13: Diagram of a SVR algorithm for a 2-D dataset, representing a linear hyperplane and the boundaries defined by the hyperparameter ϵ.

These tuning hyperparameters can be identified as one of the main drawbacks of SVR, as their selection can constitute a complex process and require validation processes, which adds to the high computational costs required, especially when non-linear kernels are employed. Additionally, the SVR is quite dependent on the data scale, for which normalization may be needed. On the other hand, this method presents a high robustness to over-adjustment even for high-dimensional datasets.

Artificial Neuronal Networks (ANNs)

Artificial Neural Networks correspond to one of those machine learning algorithms inspired by the way of working of biological neurons, and which represent a specific subset among the ML ecosystem, formally referred to as *Deep Learning*.

This method defines a neuronal network as the complex interconnection of thousands or even millions of the so-called *neurons* or *nodes*, corresponding to the most fundamental processing units. These neurons are organized within the network into three main types of layers, as can be observed in Fig. 3.14.

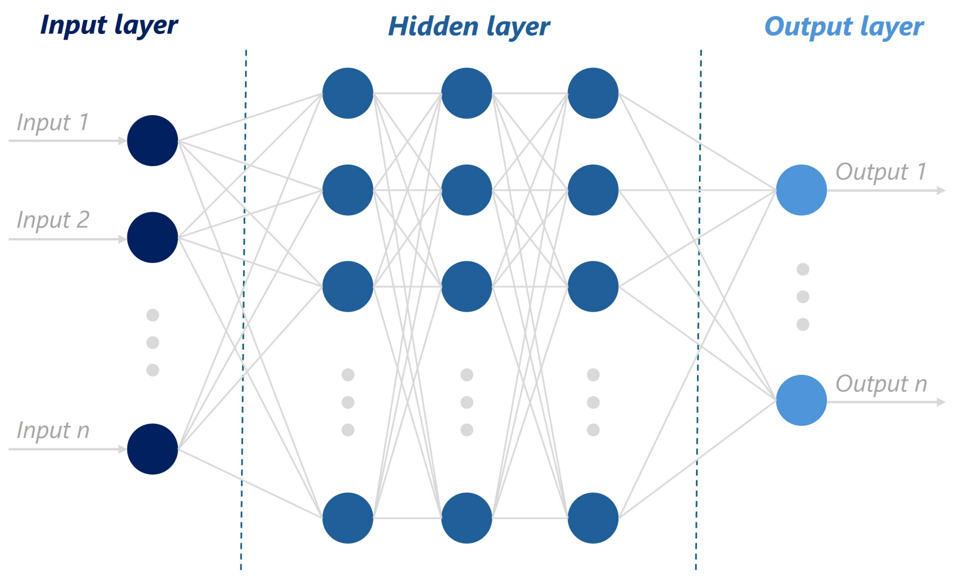


Figure 3.14: Distribution of neurons into the different layers of an artificial neural network structure: input, hidden, and output layers.

- Input layer: Each training attribute has its own neuron on this layer, responsible for receiving input information.
- Hidden layers: Neurons in this layer apply a mathematical function to their inputs and transmit the outcome to the next layer, being capable of extracting patterns within data. The number of hidden layers corresponds to one of those model's hyperparameters that needs to be defined.
- Output layers: This layer results in the final prediction of the target variables, with one neuron dedicated to each response. In this way, if only one value is expected, the output layer will be constituted by a single neuron.

Individual nodes can be connected to multiple nodes in the layer beneath them, from which data is received, as well as several neurons in the layer above them, to which data is transmitted. In this way, when the algorithm is active, the nodes receive different inputs, understood as numeric values, from each of their incoming connections. These entries are weighted and summed by neurons that will only transfer the output to the next layer if the result is above a certain threshold.

$$z = w_1 x_1 + w_2 x_2 + ... + w_n x_n + b = \sum_{k=1}^{n} w_k x_k + b \tag{3.16}$$

Where x_k represents the entry values, w_k are the weights, and b corresponds to a bias.

If non-linear analysis wants to be performed during this stage, a so-called *activation function* is introduced, allowing the inclusion of curvature in the data and the modeling of more complex problems. Although a wide variety of activation functions exists, such as the sigmoid or hyperbolic tangent functions, the *Rectified Linear Unit (ReLU)* is the most optimal for regression purposes. This activation function works by simply providing an output equal to zero when the input z is negative, and an output equal to the input whenever this is positive [11]:

$$ReLU(z) = max(0, z) \tag{3.17}$$

This step allows neural algorithms to preserve the required complexity to analyze complex relationships while preventing the model from encountering some complications, such as the vanishing gradient issue [4][46].

When the algorithm's training begins, all function weights and margins are given random values. The training dataset is fed into the input layer and propagates to the succeeding hidden layers, transmitting all computations performed in the neurons until reaching the output layer, where a prediction is obtained. This early training phase is also called the *forward propagation* stage. At the end of this phase, the predicted value is compared with the real target variable through a loss function, initiating a *back propagation* stage where weights and biases are iterated using the derivative of the loss function, and updating these parameters in the opposite direction.

The implementation of this model requires the definition of a series of hyperparameters, such as the number of hidden layers or neurons, as previously mentioned, the learning rate (α_{LR}), or the selection of the most optimal activation and loss functions. The adjustment of these hyperparameters, together with their high computational cost and the overfitting risk, makes this algorithm one of the more complex to implement. Despite its complexity, this model is capable of easily identifying intricate patterns and relationships within complex data structures, being relevant by its great adaptability and generalization [63].

3.4.2. Model Evaluation Metrics

Different from other machine learning applications, such as classification, where the objective is to predict a discrete value within a finite space, regression machine learning tries to estimate a continuous variable as close as possible to the actual values. In this way, the evaluation of the performance of the model is performed by the quantification of the deviation of the predicted values from the real ones. With this purpose, ML algorithms are normally trained with only the 80% of the training data available, retaining the remaining 20% for the computation of a series of evaluation metrics.

[4]The vanishing gradient method corresponds to a challenge faced during the training of artificial neural network algorithms when the derivatives or slopes of the activation functions become progressively small, endangering the correct training of the model [32]

These metrics guide the iteration process for the refinement and adjustment of the algorithm, being also critical for the comparison of models and the selection of the most suitable one for the problem under consideration. Although a long list of model evaluation metrics exists, the most common assessment parameters for regression problems are presented in Tab. 3.5 below [31], [84], [4]:

Metric	**Formula**	**Description**
Mean Absolute Error (MAE)	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$	It indicates the absolute difference between the predicted and actual values averaged for all the observations in the dataset and it is expressed in the same units as the target variable This is one of the most used loss functions due to its high interpretability.
Mean Squared Error (MSE)	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	Similar to the MAE but measuring the squared difference instead of the absolute one, penalizing larger errors. It is also very common in machine learning applications, although it is more difficult to interpret than the MAE.
Root Mean Squared Error (RMSE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	It can be understood as an intermediate option between MAE and MSE metrics. Similarly to the MSE, this metric penalizes large errors but in the original variable unit, as the MAE does.
Mean Absolute Percentage Error (MAPE)	$\frac{1}{n}\sum_{i=1}^{n}\lvert\frac{y_i - \hat{y}_i}{y_i}\rvert$	It indicates the average percentage of error between predicted and actual values. The closer this percentage is to zero, the more accurate the model is.
Median Absolute Error (MedAE)	$median(\lvert y_i - \hat{y}_i\rvert)$	It can be understood as the mean of the absolute errors of the predictions, being less sensitive to outliers than previous metrics.
R-squared	$R^2 = 1 - \frac{sum_{i=1}^{n}(y_i - \hat{y}_i)^2)}{sum_{i=1}^{n}(y_i - \hat{y})^2)}$	Also known as the *coefficient of determination*, this metric represents the proportion of variance in the target variable that could be predicted by the algorithm. It takes values between 0 and 1, where the higher its value, the more variance the model can explain.

Table 3.5: Common model evaluation metrics employed in regression applications.

3.4.3. Model Optimization and Adjustment

As commented in previous sections, the development of a machine learning algorithm is not a straightforward process, normally requiring the realization of multiple iterations until reaching the desired performance of the algorithm. Additionally, as can be derived from the previous paragraph, the design of the models is not a trivial process, as a deep knowledge of the training data and the algorithm fundamentals needs to be acknowledged.

In this context, there exist two main techniques that can support data scientists and developers during the development of the algorithms: *cross-validation* and *hyperparameter optimization.*

Cross-Validation

Cross-validation corresponds to a more robust technique used for the evaluation of the performance of machine learning algorithms. While the split of data into train and test subsets, as explained before, provides an initial glance at the model's predictive performance, more robust and reliable evaluations may be required to accurately identify misalignments in the model. This context is where cross-validation makes its appearance, corresponding to a statistical method that offers more stable evaluations of the performance of machine learning algorithms.

The way in which cross-validation works is by the division of the training data into a certain number of subsets, k, commonly known as *folds*. Subsequently, an iterative process is established in which the model is trained with a subset consisting of $k-1$ folds, using the remaining fold for the validation of the learning. In this way, the model is trained k times using all the available training data, both for learning and validation purposes [37].

This splitting of data explained is the most common form in which cross-validation is normally applied, receiving the name of *K-Fold Cross-Validation*, however, other variations can be implemented. Some illustrative examples of these possible deviations are listed below:

- Stratified K-Fold Cross-Validation, which works similarly to the previous case, ensures that each fold has the same proportion of categories as the complete dataframe.
- Leave-One-Out Cross-Validation, where only one sample is used in each validation step, training the model with the rest of the data in each iteration. This method significantly increases the computational step, but more accurate evaluation metrics can be obtained.

For each of the training iterations, a series of evaluation metrics is assessed, which are averaged for the obtention of a more robust and representative evaluation of the model. The work in which cross-validation works makes the model less prone to memorize training data, reducing overfitting and ensuring that it provides more accurate predictions for unseen observations. Being especially relevant for cases in which databases are small, as all samples are used both for training and validation, making use of all available information. However, this method has an associated cost to pay when applied, its large computational resource consumption [13], [85].

Hyperparemeter Optimization

Apart from the optimization of the training of the model, the algorithm itself can be enhanced by the selection of the most efficient hyperparameters. Although there exist different procedures to perform this hyperparameter optimization, two methods stand themselves above the rest: *Grid Search* and *Randomized Search* [25].

Both the grid and randomized search methods work by performing the training of the model by evaluating different hyperparameter combinations through a cross-validation approach.

In the case of the grid search, the developer specifies a series of hyperparameters and the algorithm assesses all their possible combinations, selecting the one providing the most accurate results. Although this technique is exhaustive, it may have a high computational cost associated, especially for models requiring the configuration of a large number of hyperparameters.

In contrast, the random randomized search selects a random combination of hyperparameters within a defined range, preventing the model from evaluating all possible configurations. In this case, although less computationally expensive, this technique may not provide the most optimal configuration, just a close one.

4 Methodology: Application of ML Implementation Procedure to Aircraft Price Prediction

This chapter will provide a detailed description of the end-to-end procedure established for the development of the aircraft price prediction tool, covering all the stages of ML implementation, from the definition of the problem until the tangibilization of a precise prediction model. The methodology approach followed in this chapter will be based on the combination of the best practices covered in previous sections, adjusted and particularized to the problem under concern.

Notice that the material developed during the pre-processing of data and the implementation of the models described in next sections can be accessed through the link included into App. A of this text.

4.1. Problem Definition

As already stated in Sec. 1.2, the key purpose of this study is the development of a machine learning algorithm capable of predicting the price of aircraft based on a series of technical and operational features. To provide a wider applicability to the tool, it will be focused on the estimation of the price of general aviation aircraft, offering its predictive capabilities to both private owners and aeronautical companies.

The objective is to train the model with a structured and labeled dataset including the attributes of different aircraft observations, thus being able to acknowledge the relationships between their different features and their price, employing a *supervised learning*. Additionally, due to the numerical and continuous nature of the variable to be predicted, the problem will be addressed by the implementation of a *regression* algorithm.

Finally, the construction of this tool will be guided by techniques and procedures focused on boosting the algorithm's accuracy, providing predictions as close as possible to the real values, and covering as many aircraft configurations as possible.

4.2. Data Acquisition

Aligned with the problem scope defined in previous paragraphs, a dataset containing the required information for the training of the model was then searched for. In the first place, an attempt was made to identify potential open-source datasets, and although different sources were found, these lacked aircraft observations and did not contain the required level of description. Moreover, as no access to any private or corporate information was available, it was decided to extract the data from *AvBuyer* [1], an online platform mainly focused on the purchasing and selling of general aviation aircraft.

This website presents the different aircraft in an advertisement format, indicating their main features (such as their manufacturer, model, year of production, identification numbers, flight hours, etc.) and highlighting the price at which they are purchased or sold. Therefore, because the information was not available to be directly downloaded into a data frame format, data was extracted using an automated *web-scraping* process that leveraged a publicly accessible script.

4.3. Data Pre-Processing

As described in the previous theoretical chapters, once the database that will be used for the training of the regression model is acquired, the next step is to analyse and sanitize the information. The objective of this stage is to allow the implementation of corrective techniques that will prepare the data for the subsequent feeding into the models, and the identification of meaningful data features, relationships, and limitations characterizing the specific dataset.

4.3.1. Preliminary Exploration and Inspection

Firstly, a general observation of the data was performed, identifying 3530 rows and 18 columns, with rows representing aircraft samples and columns being the variables that characterize each entry. The different columns constituting the dataset are listed below, together with a brief explanation for each of them:

- **Condition.** Being a variable used to provide a general description of the state of the aircraft, this column presents three different values: *Used*, *New*, *Project*.
- **Price.** This is the objective variable intended to be predicted based on the other aircraft features, being the estimated value for the purchase and selling of the aircraft.
- **Currency.** This column just indicates the monetary system used to evaluate the *Price* column, presenting five different values: *USD*, *EUR*, *GBP*, *CAD* and *CHF*. These abbreviations correspond to the United States Dollar, Euro, Great Britain Pound, Canadian Dollar, and Swiss Franc, respectively.
- **Category.** This variable offers a general classification of aircraft based on their configuration, with thirteen distinct categories available.
- **Year.** An attribute indicating the year in which the aircraft was manufactured.
- **Make.** A column in which the aircraft manufacturing company is indicated.
- **Model.** This feature includes the manufacturer's design model.
- **Location.** Being a column used to specify the place in which the aircraft was located at the time of their recording.
- **S/N.** This corresponds to the serial number of the aircraft, also known as the Manufacturer Serial Number (MSN). This column includes a unique identifier assigned by aircraft manufacturers during their production for tracking purposes along their manufacturing and maintenance [107]. Serial numbers can be composed of a sequence of both letters and digits.
- **REG.** This parameter is the aircraft's Registration Number, which also corresponds to a unique alphanumeric code, in this case, assigned by the aviation authority of the country where the aircraft is registered. Its main purpose is to identify the vehicle during operations and legal and regulatory processes [73].
- **Total Hours.** This column represents the total hours of operation for the given aircraft.
- **Engine 1 Hours.** This is a parameter indicating both the number of hours that the main engine has been active since its last maintenance and the type of that maintenance.
- **Engine 2 Hours.** Similar to the previous column, this variable indicates both the number of hours since the last maintenance and the type of that maintenance, but in this case, related to the second engine of the aircraft.
- **Prop 1 Hours.** As included for engines, now this column represents the operating hours and maintenance type for the propellers driven by the main engine.

- **Prop 2 Hours.** Similarly, this column indicates operating hours and maintenance type for the propeller associated to the second engine.
- **Total Seats.** Simply representing the sum of available seats in the aircraft.
- **Flight Rules.** Being a general classification for the type of procedures that govern aircraft operations under specific conditions within certain airspaces, depending on their equipment. This parameter allows the classification of aircraft into three different groups: *VFR*, *IFR*, or *BTH*. VFR stands for Visual Flight Rules, for aircraft only capable of operating using external visual references; and IFR refers to Instrument Flight Rules, understood as the capability to fly relying on navigation instruments. The additional category, BTH, is an abbreviation for "both," which is intended to represent aircraft able to operate under either VFR or IFR.
- **National Origin.** This column indicates the country where the aircraft was manufactured, understood as the location where the final assembly was performed.

Once all the variables were correctly understood, the next step was to take a deeper look at the quality of the information gathered in each of the columns. A rapid analysis resulted in the identification of three main issues that were affecting the integrity of the information:

1. Many columns presented non-homogeneous formats for their values, which could compromise the correct understanding and learning of the algorithm.
2. The number of missing values for some columns was quite large, with only 5 out of the 18 columns in the dataset having complete data for all their entries. As an example of this incompleteness, *Flight Rules* only presented 868 non-null values out of the 3530 rows, meaning that 75% of the values in this column were missing.
3. The type of data was not the one that could be expected from an intuitive approach. Some parameters as *Price*, *Year*, or *Total Hours*, could be expected to be numeric; while some others, such as *Condition* or *Flight Rules*, could be expected to be categorical. However, all of them were identified as "Object".

As an illustration for the last two items mentioned, both the non-null values count and the data type for each column can be observed in Table 4.1.

ID	Column	Non-null values	Data type
0	Condition	1761	Object
1	Price	2530	Object
2	Currency	1978	Object
3	Category	2530	Object
4	Year	2530	Object
5	Make	2530	Object
6	Model	2530	Object
7	Location	2518	Object
8	S/N	2527	Object
9	REG	2528	Object
10	Total Hours	2433	Object
11	Engine 1 Hours	1582	Object
12	Engine 2 Hours	381	Object
13	Prop 1 Hours	1165	Object
14	Prop 2 Hours	263	Object
15	Total Seats	1152	Float 64
16	Flight Rules	868	Object
17	National Origin	2522	Object

Table 4.1: Overview of columns in the initial raw database, including their number of non-null values and data type.

These detected data concerns represent a general understanding of the key issues that need to be addressed during subsequent steps of the data pre-processing stage. Improved data quality and consistency will be the main topic of the following sections, centered on the realization of a comprehensive analysis of each column, identifying necessary amendments and the procedure to implement them effectively.

4.3.2. General Data-Cleaning and Structuring

As described in Sec. 3.3.2, general data-cleaning and structuring are the first steps for the homogenization of complex datasets. Their main intention is to provide raw data with enough maturity and consistency to perform subsequent major transformations and analysis of data. At this stage, no complex amendments are made, only minor adjustments to the existing data.

Focusing on the dataset under study, a large number of inconsistencies and possible data inefficiencies were detected, requiring the application of numerous techniques for the correct treatment of the data. The next paragraphs will explore the different procedures applied, classified into the two main pillars of this stage.

General Data-Cleaning

Focusing on the homogenization and optimization of the dataset, five main actions were addressed:

1. Removal of values not providing information. During the general inspection of the data, it was detected that certain cells contained values that did not provide meaningful information. Examples of such values included terms such as *TBD*, *TBC*, *n/a*, *NA*, or *Not Listed*. These entries did not contribute to the characterization of the aircraft and could hinder the correct training of the model, therefore, they were deleted (i.e., converted to a *NaN* value).

2. Deletion of repeated rows. The process for the removal of duplicated aircraft observations was divided into two distinct approaches:

   - Firstly, the database was examined for entirely identical rows, resulting in the identification of three duplicated rows. These lines were directly removed as they did not provide any additional information.
   - In a second place, an analysis revealed the presence of rows that, while not entirely identical, shared common values for *S/N* and *REG* fields. This was not possible as serial and registration numbers are unique identifiers for each aircraft. After a deeper insight into the data, these rows were determined to correspond to the same aircraft but varied in the completeness of their information, with some rows having fewer filled columns. Specifically, 147 instances of repeated *S/N* and 17 instances of repeated *REG* values were identified. This issue was addressed by retaining the row with the largest number of fields completed. Missing values were sequentially searched on the next most complete row and supplemented where available.

3. Deletion of non-relevant columns. The omission of columns that do not provide any insight into the case under study is a common practice when processing large amounts of information. Nevertheless, to put this procedure into practice, it is necessary to have a good understanding of the influence and relationships among the different parameters. As this corresponds to an early stage of the dataset analysis, it was decided to follow a conservative approach and not remove any column at this point. However, as previously mentioned, both columns *S/N* and *REG* present unique values for all the aircraft, meaning that no relationship could be found regarding these variables. Based on this reasoning, *S/N* and *REG* parameters were deleted, reducing the number of columns to 16.

4. Correction of typographical errors. During the visualization of the different values contained in each of the columns, it was detected that some entries presented some typing mistakes, or in other cases, different nomenclatures were used to designate the same features. This issue could reduce the efficiency and accuracy of the prediction; therefore, the identified conflicting values were unified. Examples of this issue and their solution addressed can be visualized in Tab. 4.2.

   It is important to note that this step was only performed in columns presenting a manageable number of values. However, columns in the dataset including a large variety of categories can not be processed so directly, as is the case with *Make* and *Model* columns.

Impacted Column	Conflicting values	Final Homogenized values
Category	Single Engine Piston	Single Engine Piston
	Single Piston	
Category	Turboprops	Turboprop
	Turboprop	
National Origin	EU	EU
	United States	
National Origin	Swtizerland	Switzerland
	Switzerland	
National Origin	Czechoslovakia	Czech Republic
	Czech Republic	
National Origin	Britain	United Kingdom
	United Kingdom	

Table 4.2: Typographical inconsistencies detected within the dataset and final wording selected for their amendment.

5. Unification of values. Some of the categorical columns included values, which apparently represented different concepts within the attribute; however, for some categories, this was not the case. An example of this is the *Flight Rules* column, which included *VFR*, *IFR*, and *BTH* values. Nevertheless, it is known that although aircraft have the instruments to fly under instrumental flight rules, they will always be able to fly using visual references. Because of that, *BTH* value is referring to a vehicle capable of flying under *IFR*, so *BTH* was substituted by *IFR*. Another illustrative example of this type of value-homogenization amendment concerned *Category* column, which included *Twin Piston* and *Multi Engine Piston* values. In general terms, these two attributes do not represent the same concept, differing in the number of engines represented by each of the terms. However, as the dataset under study is based on common general aviation aircraft, it was assumed that any vehicle classified as *Multi Engine Piston* included only two engines, so these two terms were unified.

Data Structuring

In the former step, the dataset resulted in a more homogenized and efficient version of its initial information. Subsequently, data structuring complemented previous amendments with a series of adjustments focused on enhancing the structure of data and column formatting:

- *Price* column in the initial database included both the numeric value of the cost and the abbreviation of its currency, being *150,000 USD* an example of the values formats in this column. The last piece of information representing the monetary unit was redundant, as this information was already included in the *Currency* column. For this reason, only numeric values were retained on the *Price* variable, keeping the currency saved into a temporary column for comparison and completion purposes, as will be explored during the next sections.

- *Category* variable is a feature intended to provide some kind of classification for the different entries in the dataset. Nevertheless, by taking a look at the 13 different values included in this column, it could be rapidly identified that the values were not providing the same level of information as different classification terms were mixed. As a result, the way to mitigate this misalignment of information was to define three classification schemes: *Category*, *Propulsion*, and *No. Engines*, deriving three separate columns from a previously singular attribute. In Tab. 4.3, the mapping of the initial *Category* column with the new classification schemes can be observed.

  It is important to notice that multiple cells for *Propulsion* and *No. Engines* columns remained empty as those values could not be directly derived from the former *Category* value. However, these empty values could be estimated based on general tendencies from their corresponding category, as will be explored in later sections.

Former Category	Mapping new Category	Mapping Propulsion	Mapping No. Engines
Single Engine Piston	Airplane	Piston Engine	1
Multi Engine Piston	Airplane	Piston Engine	2
Private Jets	Airplane	Turbofan	NaN
Turboprop	Airplane	Turboprop	NaN
Turbine Helicopters	Rotorcraft	Turboshaft	NaN
Piston Helicopters	Rotorcraft	Piston Engine	NaN
Gyrocopter	Rotorcraft	Gyrocopter	NaN
Military/Classic/Vintage	Military/Classic/Vintage	NaN	NaN
Ultralight	Ultralight	NaN	NaN
Gliders/Sailplanes	Motor Gliders/Sailplanes	NaN	NaN

Table 4.3: Mapping of initial *Category* column into newly created *Category*, *Propulsion*, and *No. Engines* columns depending on their initial value.

- It was also detected that there exist different formats in which *Location* values were reported, most cells in this column corresponding to one of the following schemes: *city-state-country*, *state-country*, and *continent-country-state*. As can be observed, the details of the information were not the same for the three formats because of that, four additional columns were built to organize information and allow the consolidation of data. The new columns created were *City*, *State*, *Country*, and *Continent*. Examples of the transformation applied to *Location* column depending on the format of the data can be visualized in the Tab. 4.4.

Former Location format	Format example	City	State	Country	Continent
city-state-country	Cameron, TX, USA	Cameron	TX	USA	NaN
state-country	LA, USA	NaN	LA	USA	NaN
continent-country-state	North America, USA, FL	NaN	FL	USA	North America

Table 4.4: Mapping of initial *Location* column into new defined *City*, *State*, *Country*, and *Continent* columns depending on its initial format.

Although the vast majority of entries in the *Location* column conformed to one of these identified formats, approximately 30 values were reported in an entirely different manner. Given that this number of samples represented a manageable quantity, the necessary adjustments were made by manually populating the new columns. On the other hand, as can be observed in Table 4.4, some of the values established as *NaN* can be known when information for a lower hierarchical geographical instance is given (i.e., a continent can be known if the city, state or country are known), this will be approached as well in later steps when managing missing values.

- *Total Hours* column also presented some discrepancies in the formatting of its values. While the vast majority were simple integer values, some cells were found to have a *hh:mm:ss* arrangement. This was simply converted to hours, keeping minutes and seconds as decimals. Additionally, some cells included commas and points, which, in some cases, were misleading by the programming code. The solution to this issue was to remove commas representing thousands and to retain points when a decimal quantity was required.

- Finally, columns *Engine 1 Hours*, *Engine 2 Hours*, *Prop1 Hours*, and *Prop 2 Hours* were commonly treated as they presented the same format, consisting of a numeric value representing hours followed by its last type of maintenance, being *5560 SMOH*, an example of a value on these columns. The process followed was simply the division of the numeric values and the type of maintenance in separate columns, resulting in four additional columns as visualized below in Tab. 4.5 and 4.6.

Former Column	New column including hours	New column including maintenance
5560 SMOH	5560	SMOH

Table 4.5: Example for the division of *Engine 1 Hours*, *Engine 2 Hours*, *Prop1 Hours*, and *Prop 2 Hours* into columns including the number of hours and the type of maintenance.

Former Colum	New column including hours	New column including maintenance
Engine 1 Hours	Eng. 1 Hours	Maint. Eng. 1
Engine 2 Hours	Eng. 2 Hours	Maint. Eng. 2
Prop 1 Hours	Prop. 1 Hours	Maint. Prop. 1
Prop 2 Hours	Prop. 2 Hours	Maint. Prop. 2

Table 4.6: Scheme of columns created from initial *Engine 1 Hours*, *Engine 2 Hours*, *Prop1 Hours*, and *Prop 2 Hours*.

Finally, as mentioned in Section 3.3.1, the Python console did not automatically identify the correct data type for all columns. As a result, it was necessary to manually enforce the appropriate data types for certain columns to ensure their accurate recognition and proper handling in subsequent steps of the data analysis process. Table 4.7 below indicates the former data types in the database and the ones imposed for consistency. Notice that after this preliminary observation and structuring of data, our dataset resulted in six additional columns, while others were removed. Additionally, features were reordered for the simplification of the analysis.

Column ID	Feature	Former Data Type	Processed Data Type
0	Price	Object	Float32
1	Currency	Object	Category
2	Year	Object	Int32
3	Condition	Object	Category
4	Total Hours	Object	Float32
5	Total Seats	Object	Int32
6	Flight Rules	Object	Category
7	Category	Object	Category
8	Propulsion	-	Category
9	No. Engines	-	Int32
10	Make	Object	Category
11	Model	Object	Category
12	City	-	Category
13	State	-	Category
14	Country	-	Category
15	Continent	-	Category
16	National Origin	Object	Category
17	Eng. 1 Hours	Object	Float32
18	Maint. Eng. 1	Object	Category
19	Eng. 2 Hours	Object	Float32
20	Maint. Eng. 2	Object	Category
21	Prop. 1 Hours	Object	Float32
22	Maint. Prop. 1	Object	Category
23	Prop. 2 Hours	Object	Float32
24	Maint. Prop. 2	Object	Category

Table 4.7: Visualization of columns after the data-cleaning and structuring steps, showing the former and processed data type for each column.

4.3.3. Outliers Treatment

After the correct structuring of the dataset, a first analysis of the information was performed to identify outliers that could compromise the integrity of the data. For this purpose, each of the variables was inspected, focusing on the detection of significantly different values due to errors during the recording of the information or the inclusion of aircraft whose features deviated from the overall pattern of the dataset.

With this purpose, the numeric variables characterizing each of the aircraft were visualized using *box plots*, following the IQR approach for outlier treatment. The selection of this method was performed due to its simplicity, opportunity to visually analyze outliers, and its capability to detect extreme values within non-normally distributed attributes. As can be observed in the Fig. 4.1, a considerable number of points are located outside the whiskers of the main boxes for the different variables. As explained in the previous Sec. 3.3.4, these points are beyond 1.5 times the IQR from Q_1 and Q_3, generally considered outliers for the particular attribute.

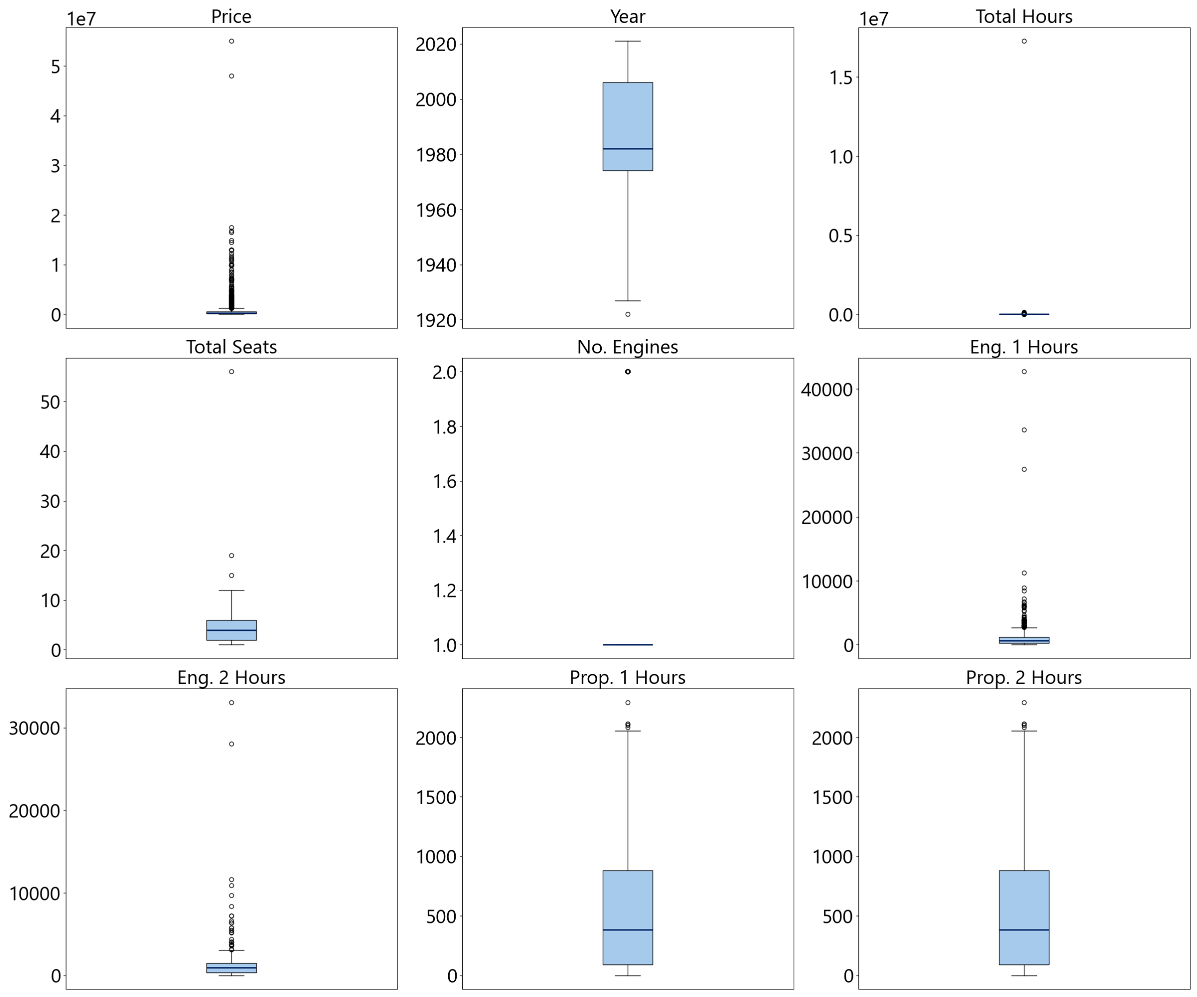


Figure 4.1: Box plots for numeric variables before the treatment of outliers.

After performing a detailed analysis for each of the variables, different rationals were behind the plotted outliers, defining a concrete method for their treatment.

In the case of the *Price* variable, two points were observed to be quite far away from the rest of the values; these entries were found to correspond to commercial aircraft, and therefore, they were removed as they significantly differ from the rest of the aircraft typology in the dataset. The remainder of the points outside of the whiskers were kept as although their price is not within the most common range, it is appropriate for the aircraft considered.

A high number of outliers were also detected for both *Eng. 1 Hours* and *Eng. 2 Hours*, for which a common approach was followed. The first step involved the classification of outliers by the categories in *Propulsion*, as the time that an engine can operate between maintenance highly depends on the type of the engine itself, obtaining Fig. 4.2.

Although the exact time between maintenances is determined by the engine manufacturer, depending on the specific engine model and the type of maintenance, some intervals could be defined based on the aircraft propulsion for the maintenance that is performed with the largest time interval, the overhaul. In this way, the Time Between Overhauls (TBO) was examined for each *Propulsion* category [54]:

- The time that a piston or turboshaft engine can operate between major maintenance is quite similar, ranging from 1,800 to 2,400 hours.

- In the case of turbofan and turboprop engines, the time interval between overhauls ranges anywhere from 3,000 to 6,000 flight hours.

As can be observed in Fig. 4.2, the limits defined by the upper limits of the whiskers are quite aligned with the major value of the theoretical interval defined for each propulsion category. Because of that, every point out of the whiskers was removed, resulting in Fig. 4.3.

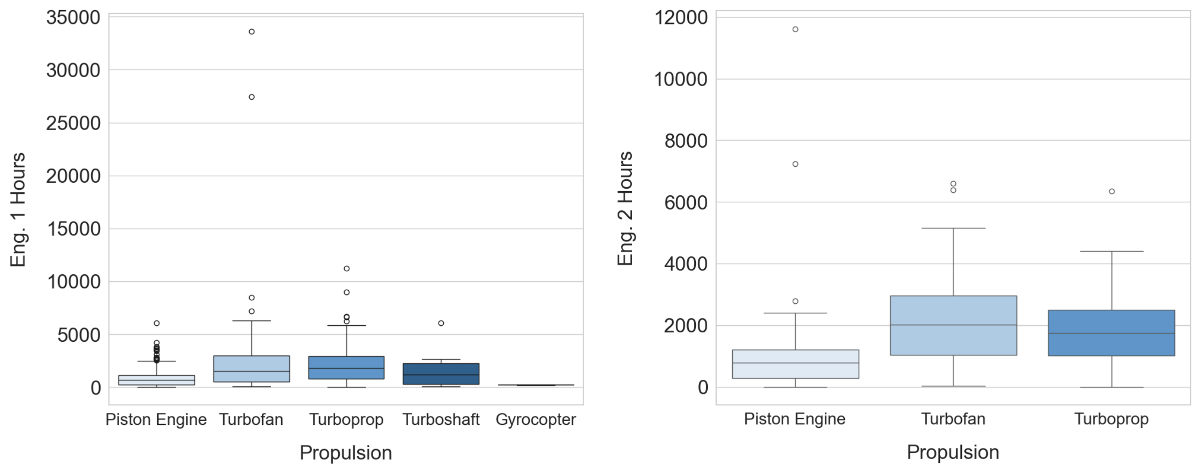


Figure 4.2: Box plots for *Eng. 1 Hours* and *Eng. 2 Hours* variables gouped by *Propulsion* type before the treatment of outliers.

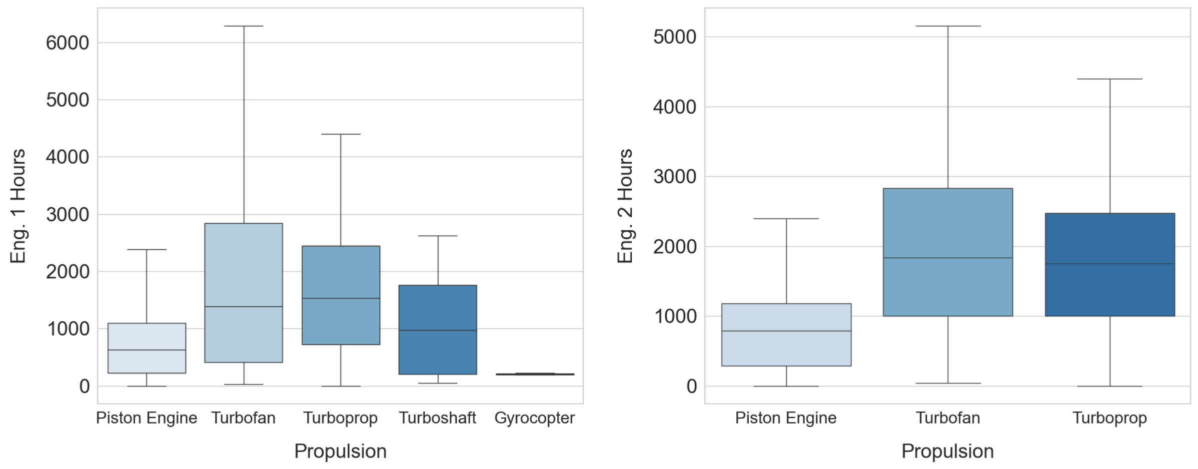


Figure 4.3: Box plots for *Eng. 1 Hours* and *Eng. 2 Hours* variables grouped by *Propulsion* type after the treatment of outliers.

A similar approach was addressed for *Total Hours* variable, but now classiflying outliers by *Category*, as observed in Fig. 4.4.

In this case, some research was also performed to estimate a lifespan interval for each of the different categories of aircraft. This variable can not be directly treated as previously performed with *Eng. 1 Hours* and *Eng. 2 Hours* as there is no clear regulated guide for defining aircraft lifespan. Normally, this lifespan varies depending on the usage patterns, maintenance, and operational conditions. However, after doing some research, a top limit was established for each category to delete improbable values:

- The airframe of a general aviation airplane can last anywhere from 10,000 to 30,000 flight hours [108]. Getting into propulsion specifications, it was found that piston-engine aircraft can reach up to 15,000 hours, while turboprop and turbofan (private jet) models widen the top limit to 30,000 hours [111].

- To estimate the lifespan of rotorcrafts, some common models were reviewed, concluding that turboshaft helicopters normally have a lifespan from 10,000 to 30,000 flight hours, while for piston-engine helicopters, this lifespan is a bit reduced.

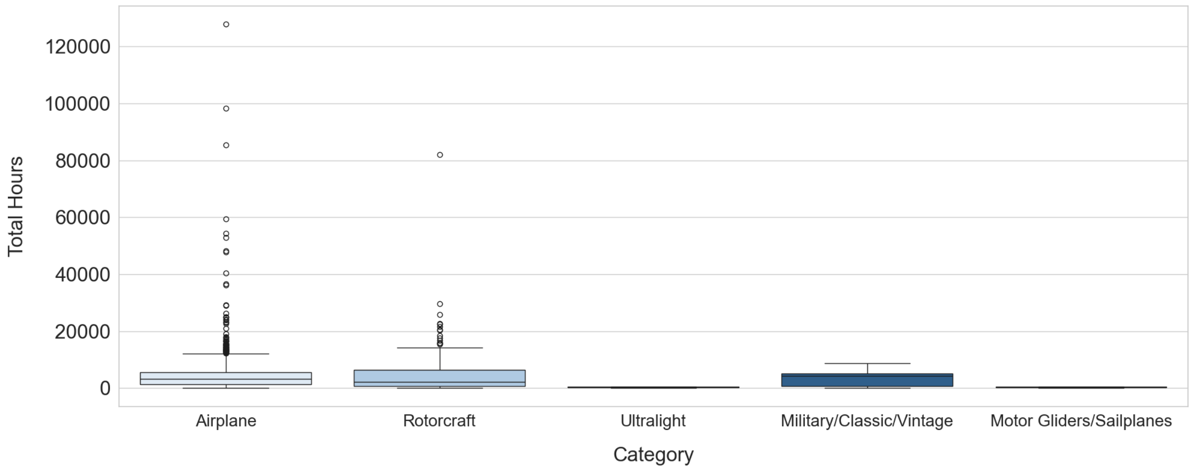


Figure 4.4: Box plots for *Total Hours* grouped by *Category* before the treatment of outliers.

In this way, a threshold was established on a value of *Total Hours* equal to 30,000 flight hours in the case of *airplane* and *rotorcraft* values. Notice that for *ultralight*, *military/classic/vintage*, and *motor gliders/sailplanes* no outliers treatment was required as they do not present any extreme value that could be misleading. After the deletion of outliers, Fig. 4.5 was obtained:

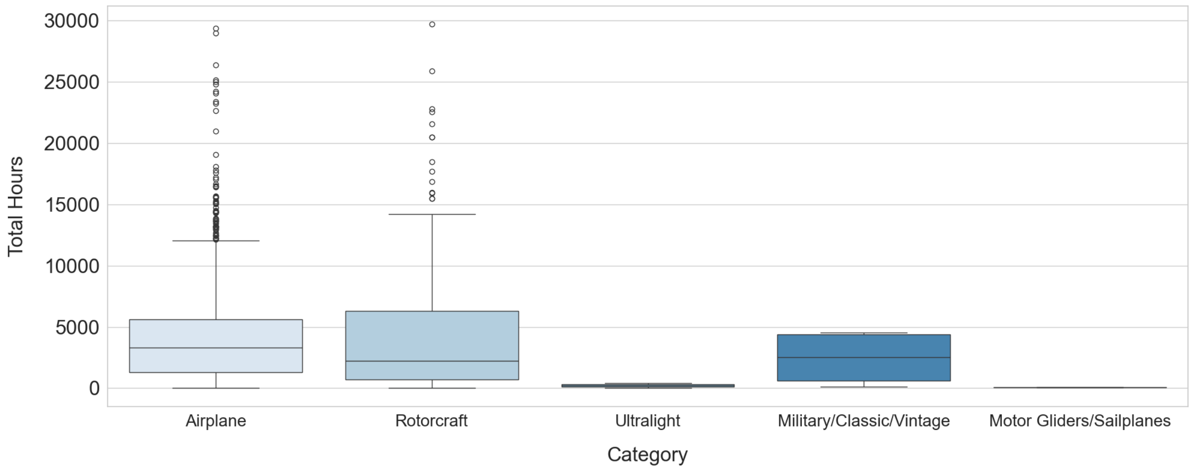


Figure 4.5: Box plots for *Total Hours* grouped by Category after the treatment of outliers.

Now focusing on variables with a smaller number of outliers identified, such as *Year* and *Total Seats*, as points out of the whiskers of their corresponding boxplot represented a quite small proportion of the dataset, these outliers were directly deleted.

This was also the case for *Prop. 1 Hours* and *Prop. 2 Hours*. For these variables, it can be observed that the top limit of the whiskers is slightly above 2,000 hours, being aligned with what was found in the literature, stating that aircraft propellers require major maintenance in a range of 1,000 to 2,000 flight hours [3].

Lastly, note that points in *No. Engines* = *2* are not really anomalies; they are just represented as outliers, as the number of aircraft having one engine is larger than those having two engines. Therefore, no points were deleted concerning this variable.

All of these outlier treatment steps resulted in the deletion of 72 aircraft, reducing the size of the dataset to 2,291 rows and resulting in the new box plots visualized below:

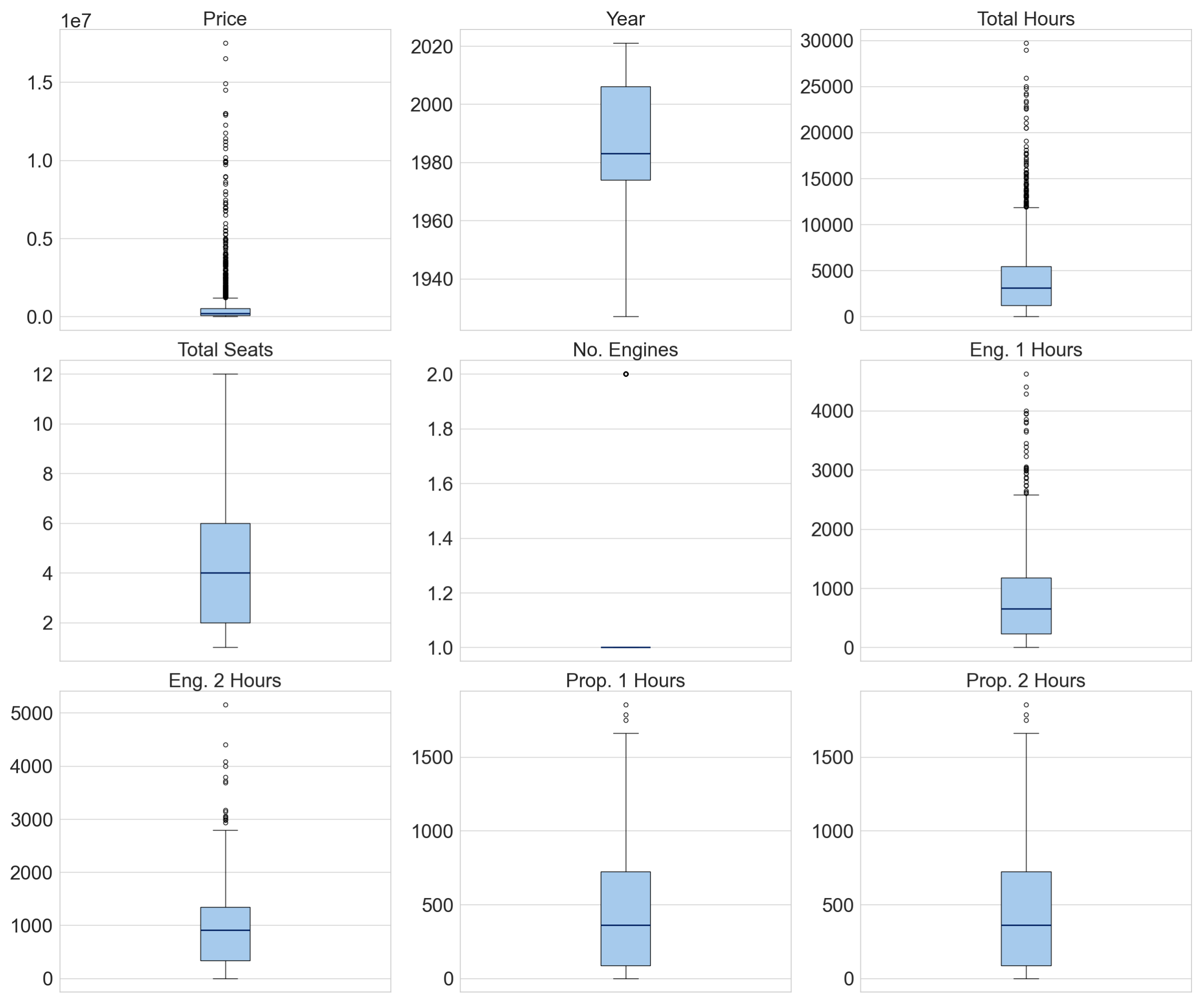


Figure 4.6: Box plots for numeric variables after the treatment of outliers.

As can be noticed from previous paragraphs, the treatment of outliers was focused only on numeric variables, as there is no statistical procedure to identify outliers within categorical variables. However, low-frequency or rare categories could be detected by exploring categorical variables employing inspection techniques. In this specific dataset, *Model*, *Make*, and those variables associated with the location of the aircraft presented values that were very limitedly repeated. However, they correspond to perfectly feasible aircraft, so they were retained for further inspections.

4.3.4. Handling of Missing Values

The general inspection of the dataset performed in the early stages of the analysis rapidly identified many empty cells that needed to be treated. The deletion of extreme/misleading values performed in the previous step resulted in a clean and reliable set of data that could be used for the estimation/derivation of the missing values, providing a complete dataset for the subsequent analysis of the information and implementation of machine learning algorithms.

The first step in the treatment of missing data is to visualize the most critical variables and their proportion of empty cells. This was achieved by the representation of missing data in a matrix format as shown in Fig. 4.7.

From the previous figure, it can be observed that although some variables are more affected by null values, almost all of the features have some empty cells, except for *Category*, *Make*, and *Model* attributes. For this

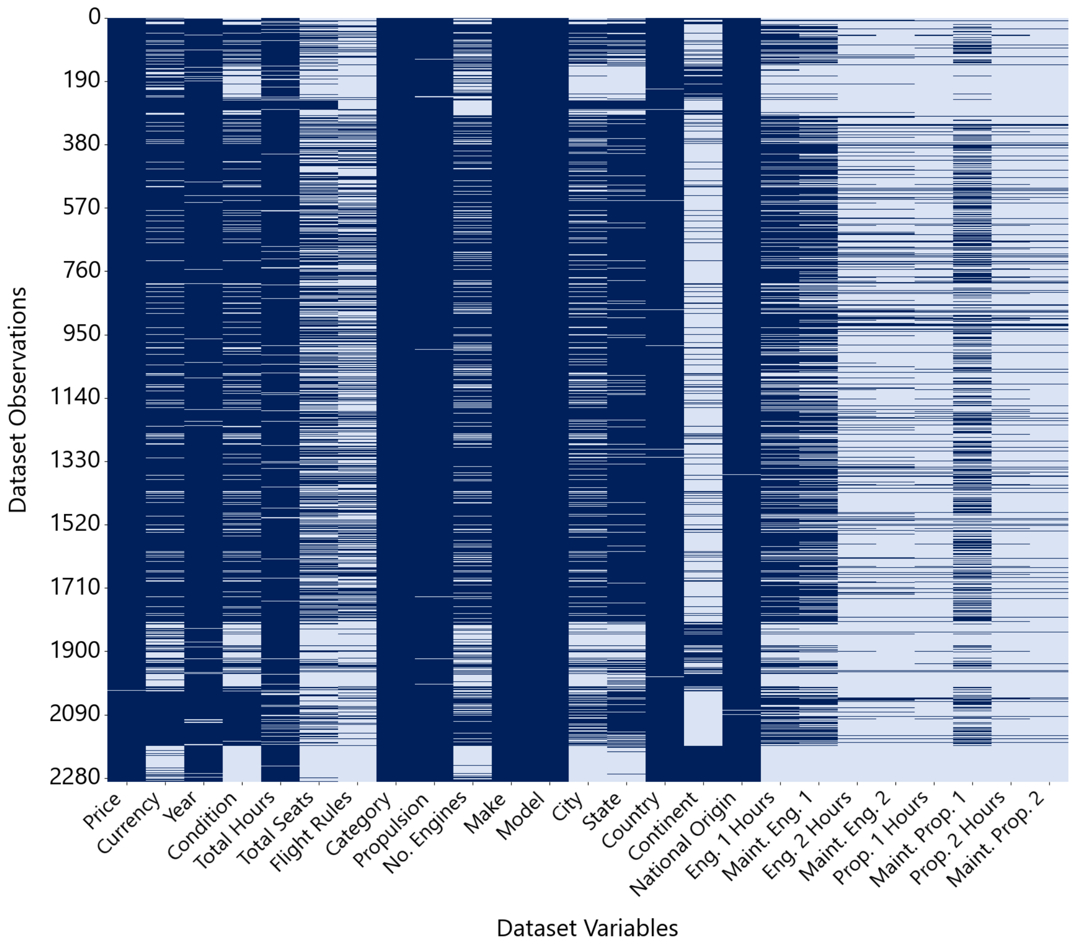


Figure 4.7: Matrix representation of the dataset highlighting missing values in light blue color.

reason, a personalized approach was followed for each of the columns, using the most suitable strategy from those identified in Sec. 3.3.5:

- Price: This column just included one empty cell. After further analysis of the row containing the missing value, it was detected that many other variables were also empty; for this reason, this row was directly removed.

- Currency: As explained in previous sections, raw data in *Price* column included both the numeric value representing the cost of the asset and the symbol representing the currency in which the price was measured. After the cleaning of this column, symbols were removed and saved into a temporary variable. This step allowed to posteriorly complete empty cells into *Currency* column, reducing the initial 417 *NaN* values to seven. The remaining empty cells were completed assuming the currency used in the country where the aircraft was located.

- Year: Missing values in the *Year* column were addressed by imputing the mode value of the variable. Using this approach, the 62 initially missing cells were assigned the value 1987, which corresponds to the year recorded in 15 different rows of the original dataset.

- Condition: This variable presented 611 empty cells, as this number was very high and this column corresponds to a categorical variable, a new category *No information* was created for replacing those empty cells.

- Total Hours: The 116 empty cells in this column were completed using the mean of the existing values in the attribute, in this case having a value of 3,939.38.

- Total Seats: Missing values in this column were treated following a two-step approach. Firstly, a mapping was performed between values in *Model* and *Total Seats* attributes; in this way, empty cells in *Total Seats* were imputed if there existed another row with the same *Model* and known value for *Total Seats*. While it is acknowledged that some aircraft models may offer customizable seating configurations based on their intended purpose (e.g., executive transport, training, or medical use), this approach was considered reasonable given the typically consistent seating capacities within smaller, non-commercial aircraft. This first method reduced the initial number of 1,188 missing values to 655. Secondly, the remaining empty cells were filled with the mode of the variable, which was equal to 4, increasing the initial number of aircraft with four seats from 597 to 1252.

- Flight Rules: This column presented a large number of missing values, as 1,465 of its cells were initially empty. Following a similar approach as in the previous column, a mapping was performed between values in *Model* and *Flight Rules* attributes, reducing empty cells to 793. As the number of remaining missing values was still too high, a new category named *No information* was created and assigned to those cells.

- Propulsion: This column had a reduced number of empty cells, as only 13 values were missing. After taking a deeper insight into their corresponding rows, it was identified that this variable was missing for aircraft that were classified as *Military/Classic/Vintage*, *Ultralight*, and *Motor Gliders/Sailplanes*. Additionally, all of these rows had a value in *Model* column, which allowed the research for the type of propulsion of each of these aircraft and the completion of the column accordingly.

- No. Engines: This column could not be completed similarly to the approach used for the *Propulsion* attribute, as it contained a total of 693 missing values. This significant number of missing entries made it impractical to individually search for and assign the corresponding value to each aircraft in the dataset. For this reason, a two-step approach was again followed. Firstly, for those rows with existing values of *Eng. 1 Hours* but null value of *Eng. 2 Hours*, the value of engines was assumed to be 1; on the contrary, if both *Eng. 1 Hours* and *Eng. 2 Hours* attributes had a value assigned, then the number of engines was assumed to be 2. This method reduced the number of missing values to 552, for which a second strategy was implemented, based on the assignment of the mode depending on the *Propulsion* type. In this way, using the rows where *No. Engines* was not missing, the most repeated value of engines was computed for the different *Propulsion* categories and their corresponding proportion, obtaining Tab. 4.8 below:

Propulsion category	**Mode**	**Proportion**
Turbofan	2	77%
Turboprop	2	52%
Turboshaft	1	91%

Table 4.8: *No. Engines* mode and its proportion for *Propulsion* categories.

  As can be observed, this procedure may have induced some error as *No. Engines* is being filled based on the most frequent value of the dataset, although that frequency is not significant, as in the case of *Turboprop* aircraft. Also, notice that Tab. 4.8 only includes the type of propulsion for which *No. Engines* values were missing.

- City: This variable is the highest-granularity value derived from the initial *Location* column, and its level of information may not be required for the project under scope. This, together with the 762 missing values, resulted in the decision to delete this attribute for the training of the model. However, this column was temporarily conserved to support the treatment of missing data concerning *State* and *Country* variables.

- State: Missing values found in the *State* column are likely due to the nature of the definition of this variable, corresponding to MNAR empty cells. The division of countries into federal states is not a global common practice. In fact, in the dataset, it is only provided when aircraft have *United States* in *Country* column. Consequently, the missing values for countries other than the United States were replaced with *N/A*, reducing the number of empty cells from 512 to 58. Following this initial step, a thorough process was applied to address the remaining missing values. This involved identifying matching *City* rows with known *States* and substituting the missing values accordingly, resulting in the reduction to 12 empty cells. The remaining missing values were manually imputed as they had values for *City*, allowing the determination of the state in which the aircraft were located.

- Country and Continent: Following the same procedure as for *State* column, empty cells in *Country* attribute were completed by looking for rows with matching states and known countries. On the other hand, *Continent* variable was completed based on *Country* column. This method allowed to fully determine the 12 and 1,725 initial missing values for *Country* and *Continent* columns, respectively.
- National Origin: This column presented only six empty cells. By taking a deeper insight, it was detected that those rows in which this attribute was missing corresponded to either aircraft replicas or homebuilt flying vehicles. Because of that, it was considered that a good estimation for their national origin value could be using the country in which they were located.
- Columns related to engine/propeller flight hours and maintenance type: These columns contained a substantial number of missing values, making them challenging to complete for their use in algorithm training. For this reason, these attributes were removed from the dataset.

Finally, the result of the implementation of the different missing data techniques resulted in a complete collection of 16 features, removing 9 columns from the initial data whose missing values could not be treated.

4.3.5. Exploratory Data Analysis

After the pre-processing techniques mainly related to the homogenization, structuring, removal of extreme values, and completeness of the raw data, the EDA represents a fundamental step for the detailed analysis of attributes' characteristics and patterns that may significantly affect advanced modeling techniques, as will be explored within the following steps.

It is important to mention that the EDA is generally performed during the early pre-processing stage, typically before the treatment of outliers and missing values. Nevertheless, for the case under study, due to the relatively high number of these inconsistencies, it was decided to implement it after these pre-processing steps, as the distributions could be considerably altered. For this reason, simpler mechanisms were applied to detect discrepancies in previous phases, maintaining the thorough exploration of data for selecting the features that will train the model and for analyzing the most suitable feature transformation techniques.

The analysis was organized into two main partitions: One of them dedicated to the visualization and exploration of the different variables in a univariate way, allowing the identification of values distributions and patterns within the own attribute. In contrast, the second partition was focused on the bivariate analysis of the different predictor variables with respect *Price* column, detecting correlations and tendencies among them.

Univariate Exploration of Data

In this phase of univariate exploration of data, different analysis methods were employed depending on the type of variable to be analyzed:

- Continuous numeric attributes were explored by the computation of their descriptive statistics for the achievement of a global vision of their distribution, which was subsequently complemented by their graphical representation employing histogram plots. These plots incorporate a tendency line to facilitate the interpretation of both the shape and dispersion of the data.
- On the other hand, categorical and discrete numeric variables were visualized mainly through the use of pie charts, as these correspond to the most effective graphics to illustrate the relative proportions of the categories within a variable. As an exception, density maps were used for the representation of the geographical attributes, as these plots allow the geospatial distribution of the data, facilitating sample concentration.

In this way, the statistical analysis and the charts generated during the univariate analysis can be found below:

Continuous Numeric Variables

As can be concluded from the statistical values in Tab. 4.9 and the corresponding histograms in Fig. 4.8, both *Price* and *Total Hours* variables present a high positive skewness and excessive kurtosis (leptokurtic distributions) compared with those of a normal arrangement, especially in the case of *Price* attribute. This means that their values are highly right-skewed and with accentuated peaks, which can be translated into the vast majority of prices being concentrated at low and medium values, being less frequent the presence of high aircraft prices. This behavior causes the mean and median to be significantly away from each other and to have relatively high standard deviations.

As seen in the previous Sec. 4.3.3, these extreme values are perfectly within the scope of the problem, being representative samples of aircraft for the application under study, therefore, they are retained for the training of the model. However, some advanced techniques were implemented for the normalization of this variable during advanced stages, especially when the algorithm employed had a high reliance on the normality of data.

Variable	**Mean**	**Median**	**Stand. Dev.**	**Skewness**	**Kurtosis**
Price	670,640.30	198,000.00	1,502,653.00	5.51	38.93
Total Hours	3,939.38	3,642.70	3363.50	2.05	6.99

Table 4.9: Statistical analysis of *Price*, and *Total Hours* numeric continuous variables, including: mean, standard deviation, skewness, and kurtosis.

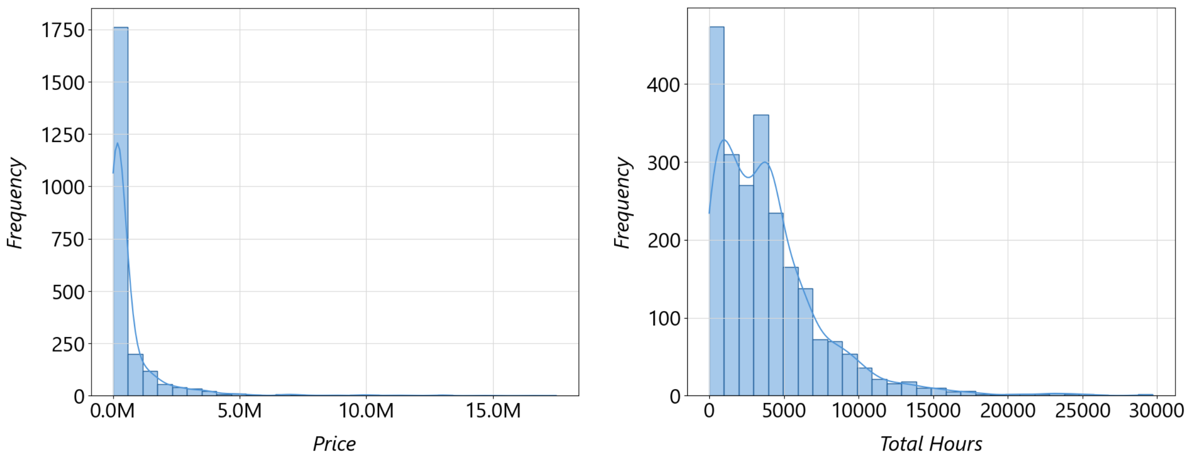
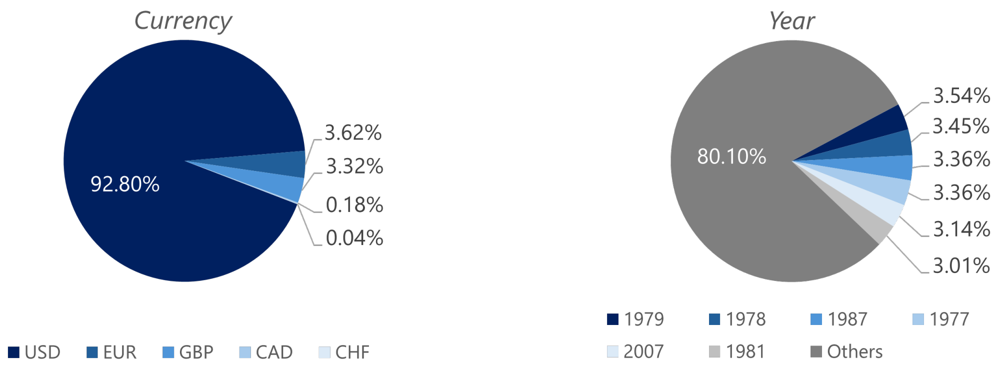


Figure 4.8: Histogram charts obtained for the univariate analysis of *Price*, and *Total Hours* continuous numeric variables.

Categorical and Discrete Numeric Variables

In the case of categorical and discrete numeric variables, the pie charts obtained for their representation can be consulted in Fig. 4.13, and in Fig. 4.14 for the case of the location attributes illustrated by density maps.



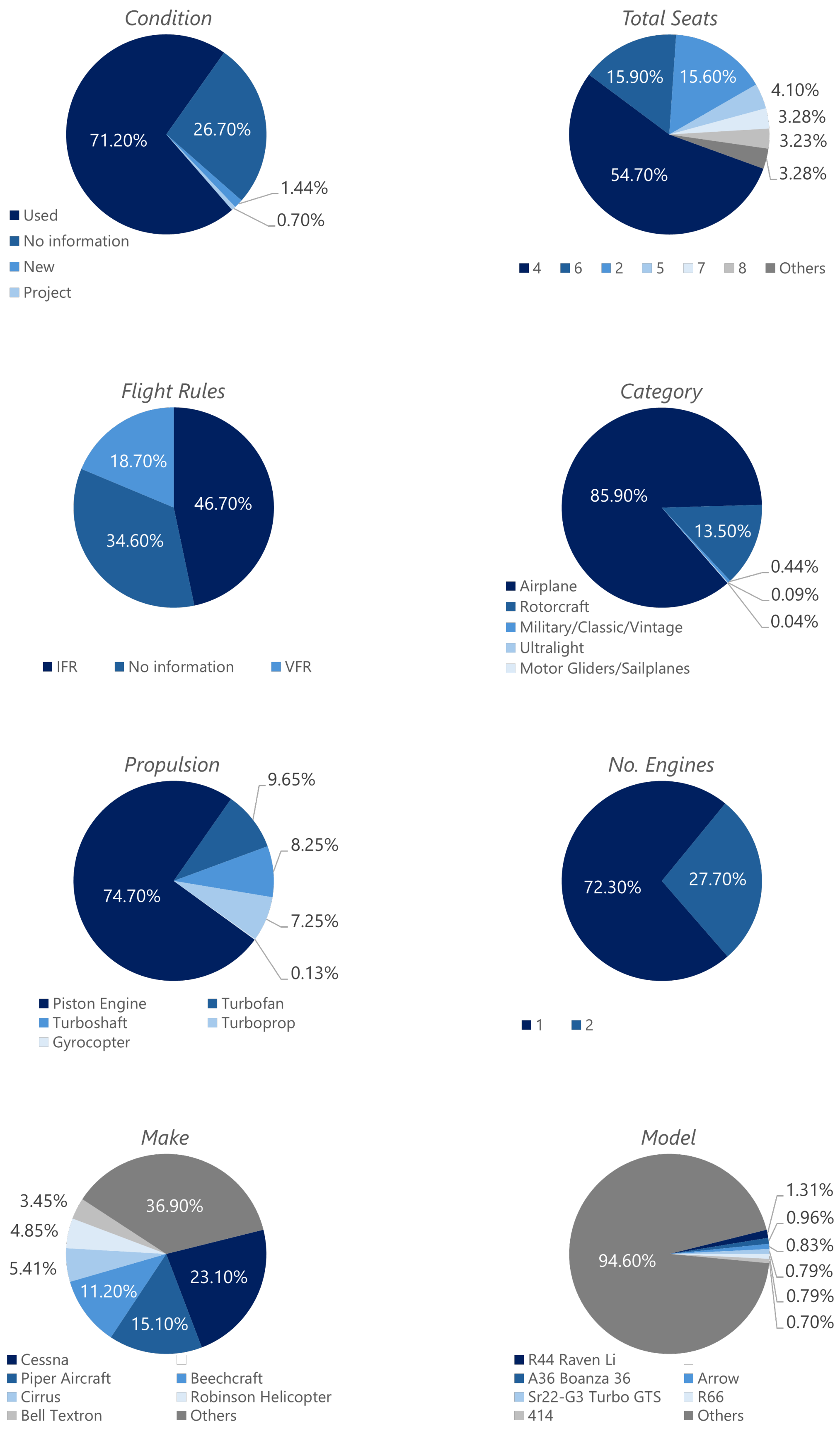


Figure 4.13: Pie charts for the exploration of categorical and discrete numeric variables, including *Currency*, *Year*, *Condition*, *Total Seats*, *Flight Rules*, *Category*, *Propulsion*, *No. Engines*, *Make*, and *Model*.

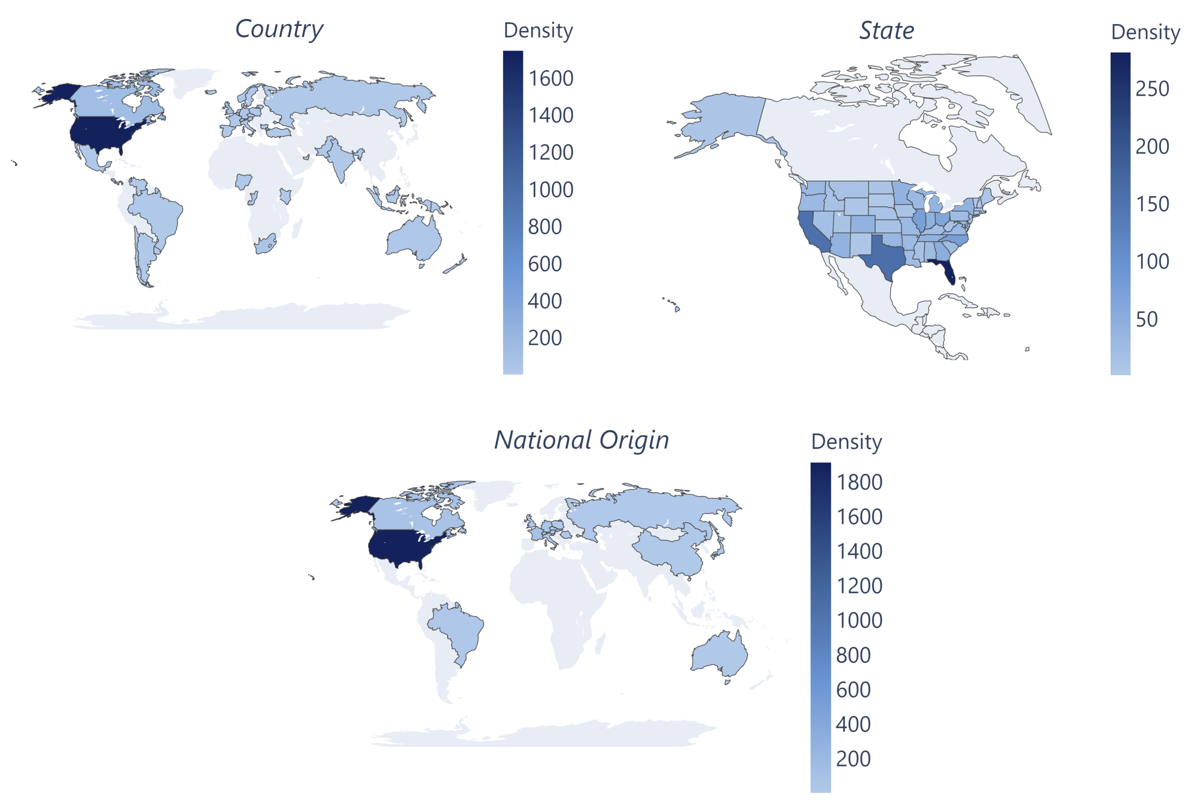


Figure 4.14: Density maps obtained for the representation of categorical geographical variables, including *Country*, *State*, and *National Origin* attributes.

As can be seen in previous charts, there exists a predominance of certain features within the aircraft used for the study. Most of the samples correspond to used piston engine airplanes with a single engine, produced and located in the United States, especially in states such as Florida, Texas, and California. This high presence in the US means that the vast majority of their prices are evaluated in dollars.

More distributed values are observed in other features, having aircraft with different seat configurations, although the most predominant ones are those with 4 seats. A certain uniform distribution is also observed on the type of flight rules available in the recorded samples. Additionally, a higher variety in categories is present for the manufacturers and models of the aircraft, with Cessna, Piper Aircraft, Beechcraft, Cirrus, and Robinson Helicopter being the most frequent manufacturers, with the models included in Tab. 4.10.

Most frequent manufacturer	**Most frequent model for the corresponding manufacturers**
Cessna	414
Piper Aircraft	Arrow
Beechcraft	A36 Bonanza 36
Cirrus	SR22-G3 Turbo GTS
Robinson Helicopter	R44 Raven II and R66

Table 4.10: Most common aircraft models from the most repeated manufacturers within the dataset.

Bivariate Exploration of Data

For the study of the influence of the different attributes on the value of the price of the aircraft, a similar approach to that of the univariate analysis was followed, using a specific visualization technique according to the nature of the attribute to be analyzed. In this case:

- Scatter plots were chosen for the visualization of continuous numeric variables with the *Price* attribute, allowing the representation of every single data point. On this occasion, although the *Year* variable has a discrete numeric character, it was also represented using scatter plots, allowing the inclusion of all its values for the analysis.

- On the other hand, categorical variables and the remaining discrete numeric attributes were analyzed by the use of violin plots. These figures represent the relative density of samples for the different price values within each of the categories of the attributes, allowing the identification of those categories having a greater impact on the aircraft price. Geographical attributes will be explored again following the construction of density maps, in this case, representing the median of *Price* for each of the geographies studied.

Additionally, the linear relationship of all the predictor variables on the target attribute was explored by the construction of a correlation matrix. The obtention of this figure required the implementation of a complex procedure due to the presence of both numeric and categorical variables, demanding the combination of several techniques:

- The computation of the correlation coefficient among numeric variables was addressed by the direct implementation of the linear Pearson's coefficient formula.
- In the case where two categorical variables were involved, a Chi-square test was conducted for the use of Cramer's V coefficient.
- Finally, the correlation between categorical and numeric variables, a case of special interest within the dataset under study, because of the presence of a high number of categorical variables and the numeric character of the target attribute, was tackled by the codification of categorical variables and the implementation of the linear Pearson's coefficient.

In this way, the charts generated during the bivariate analysis can be found below:

Continuous Numeric Variables

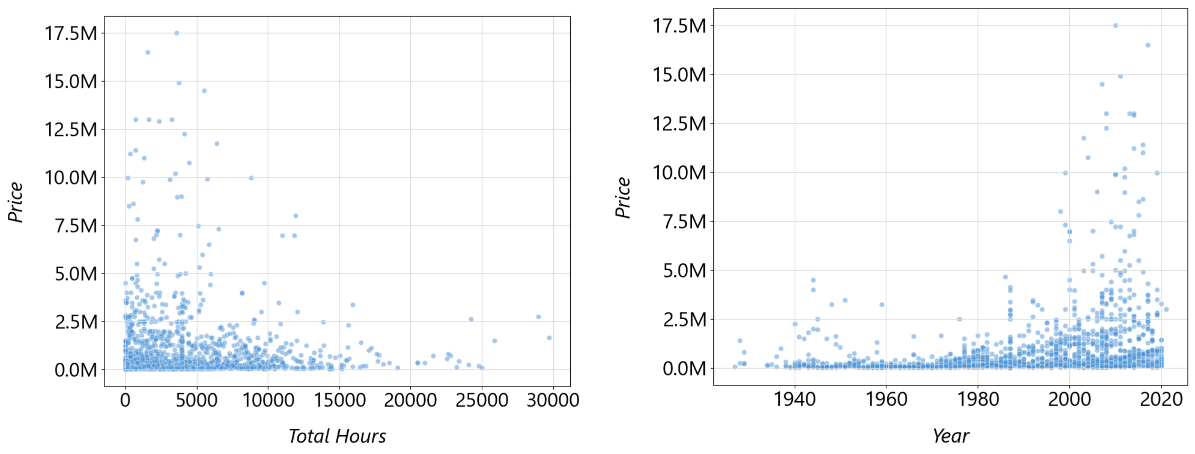


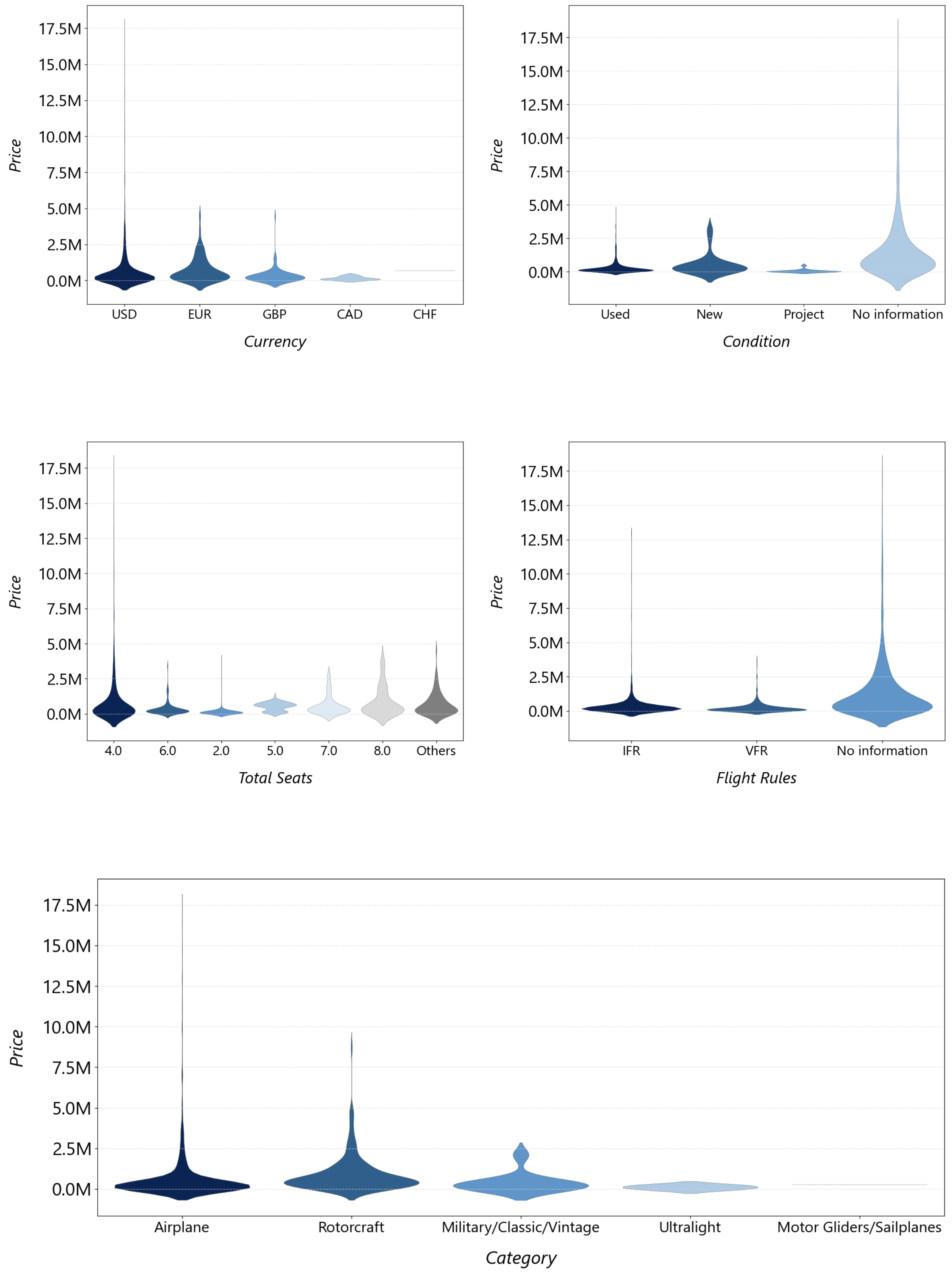
Figure 4.15: Scatter plots for the bivariate analysis of *Total Hours* and *Year* variables vs. *Price* target attribute.

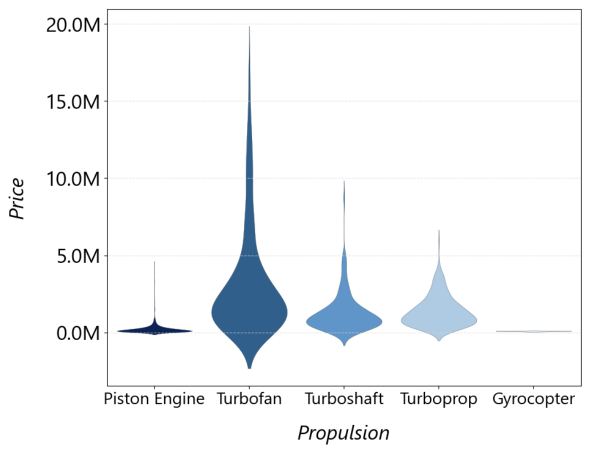
As can be observed in Fig. 4.15, the majority of the points are concentrated at the low price values, being aligned with what was concluded during the univariate analysis. Additionally, it can be observed an increase in the aircraft price as the number of total flying hours is reduced and as the year of manufacturing is increased, meaning that the age and usage of aircraft considerably affect their price. Furthermore, a certain axial symmetry can be observed between both *Total Hours* and *Year* variables, indicating that those aircraft which have been more recently manufactured have the least amount of flying hours accumulated.

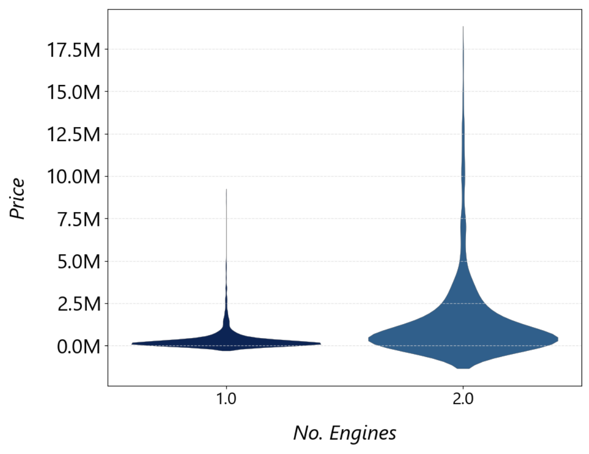
Finally, no linear relationship can be detected between these predictor variables and the *Price* attribute, which will be more deeply analyzed with the construction of the correlation matrix in later steps.

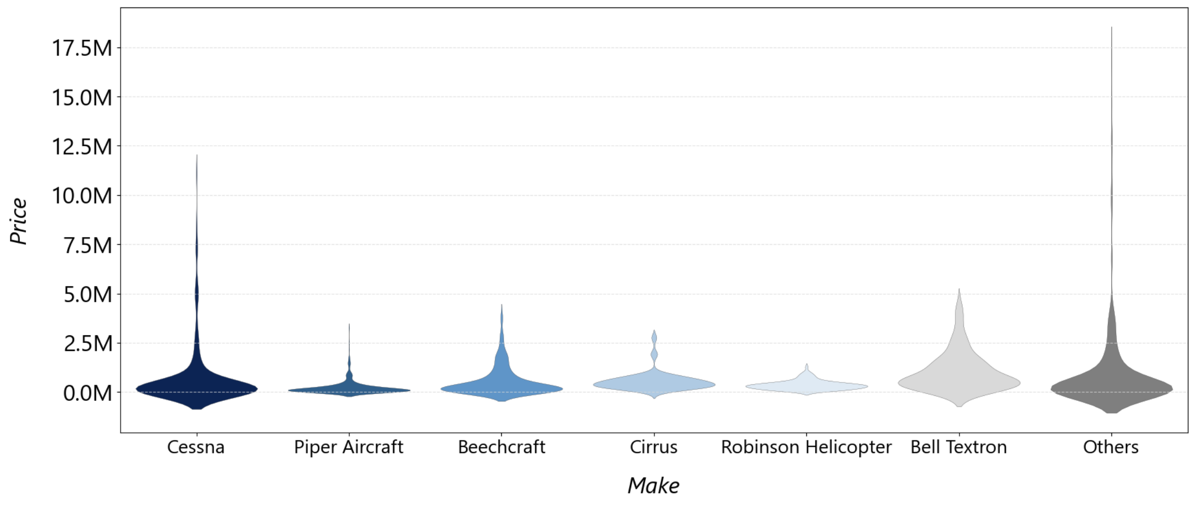
Categorical and Discrete Numeric Variables

In the case of categorical and discrete numeric variables, the violin plots obtained for the exploration of the relationship between predictor features and the target attribute can be consulted in Fig. 4.21, and in Fig. 4.22 for the case of location attributes, using density maps.









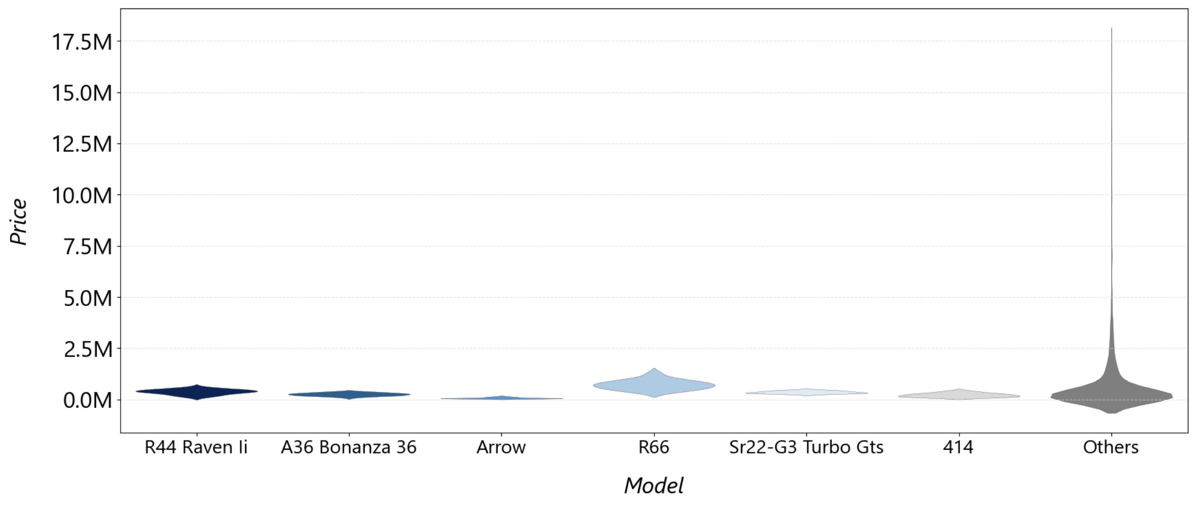


Figure 4.21: Violin plots for the exploration of categorical and discrete numeric variables, including *Currency*, *Condition*, *Total Seats*, *Flight Rules*, *Category*, *Propulsion*, *No. Engines*, *Make*, and *Model* vs. *Price* target attribute.

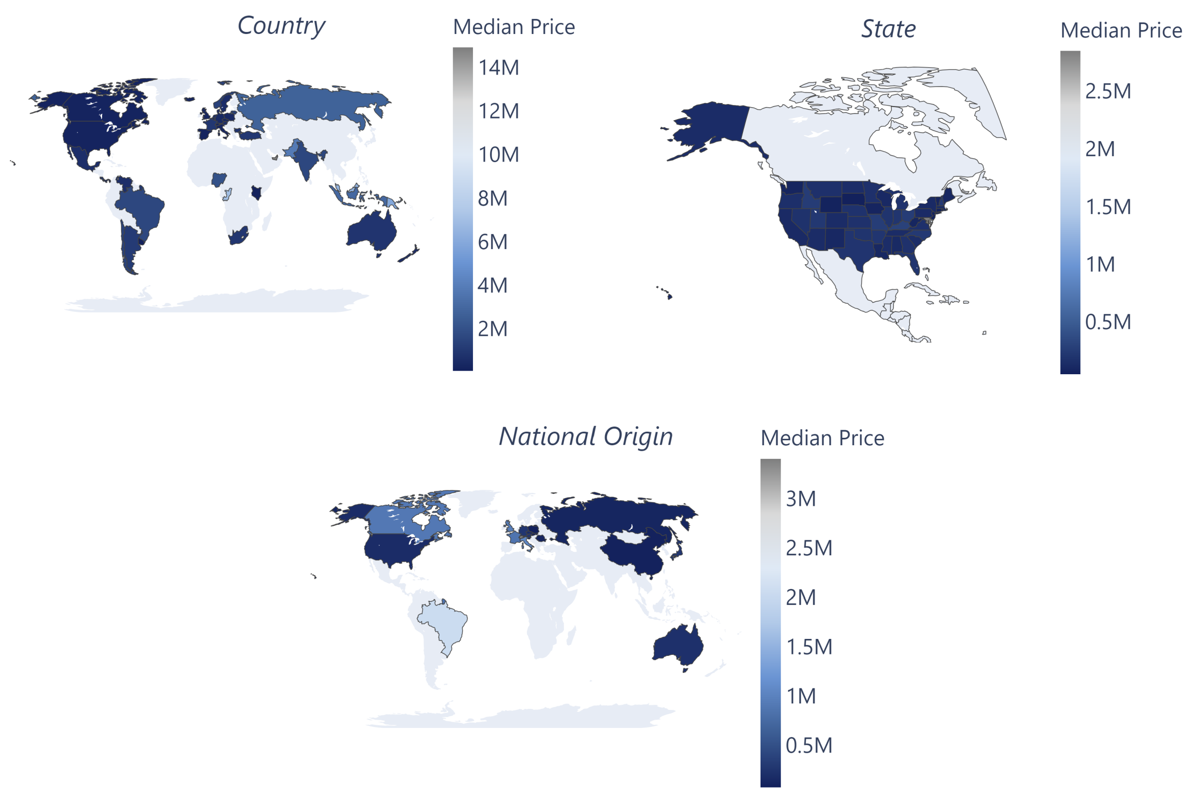


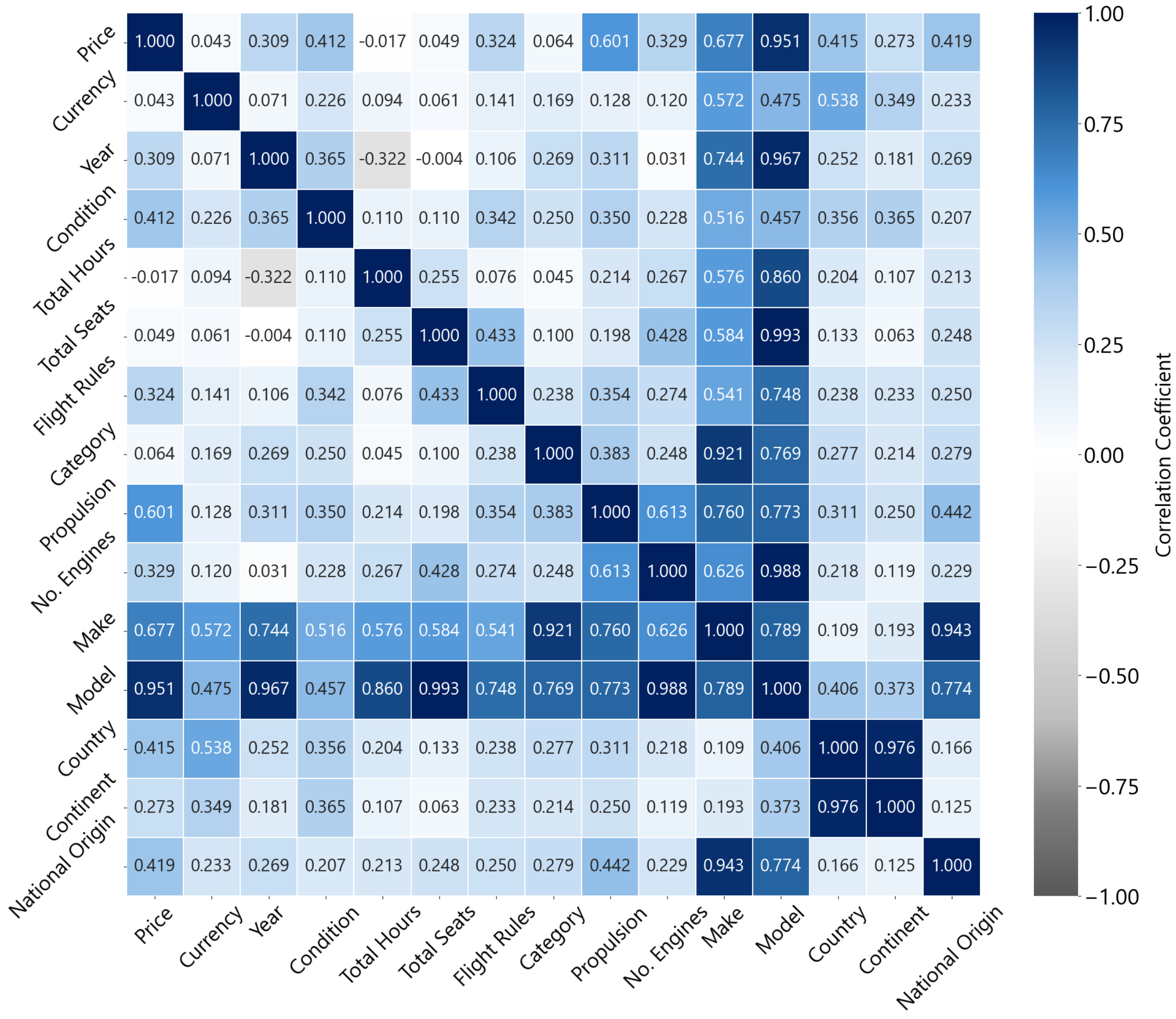
Figure 4.22: Density maps obtained for the bivariate analysis of categorical geographical variables, including *Country*, *State*, and *National Origin* vs. *Price* target attribute. The color code represents the median price of aircraft by geography.

Taking a deeper view into the different charts obtained, the following insights were found:

- The currency used for the valuation of aircraft does not significantly impact the value of their price. In general terms, all currencies show a similar density for the different prices, with a slight increase for those aircraft evaluated in euros. Additionally, the most expensive samples are measured in US Dollars. However, this variable is not determinant as the price associated with a currency is highly dependent on the value of that currency at a point in time, experiencing significant variations over the years.
- For the *Condition* attribute, it was found that a slight price increase is associated with those aircraft labeled as *New*, aligned with what could be expected. Focusing on *No information* category, it can be observed how this label expands for all price ranges, indicating that there are no details about the condition of those aircraft with higher prices.
- A general price increase can be observed as the number of seats rises. In this case, it is relevant to mention that the highest prices are associated with 4 seats, probably caused by the incompleteness of this attribute in the raw data and the application of imputation techniques for its treatment.
- Higher prices are associated with aircraft able to fly under IFR than those that only allow VFR. Additionally, samples with no *Flight Rules* category identified expand again for all price ranges.
- The majority of rotorcraft present higher prices than any other aircraft category, however, airplanes cover a wider range, being associated with those samples with the largest price values.
- Most of the samples corresponding to piston-engined aircraft are concentrated within a lower price range than for the rest of the propulsion categories. Recalling back the univariate analysis where it was found that *Piston Engine* category is associated with almost the 75 % of the observations, this insight is of relevant importance as it reveals that most of the aircraft are concentrated at lower variables because of their propulsion classification dominance. On the contrary, turbofan, turboshafts, and turboprop aircraft are evaluated at higher prices, being the turbofan samples the ones corresponding to higher prices.

- Similarly to the *Propulsion* attribute, although the most frequent value of *No. Engines* was discovered to be 1, those aircraft having 2 engines are found to have larger associated prices.
- In the case of *Make* and *Model* attributes, it is very difficult to assess the influence of their categories on the price of the aircraft by the use of violin plots, as an extensive number of categories are included for each of these attributes. For this reason, further steps will involve the construction of a correlation matrix to try to expand this analysis.
- Finally, by taking a look at the geographical attributes, it can be observed that aircraft located in the US present the lowest prices, especially compared with other countries such as Russia, India, Malaysia, Cambodia, New Guinea, or the Republic of Congo. On the other hand, aircraft produced in Brazil are the most expensive, followed by those manufactured in France, Italy, the United Kingdom, and Canada. This change in price depending on the geographical location is probably caused by the typology of aircraft produced and used in each of these places.

Finally, the correlation matrix for the different variables within the dataset was obtained to get an idea of the linear degree of relationship between the variables, resulting in Fig. 4.23:



	Price	Currency	Year	Condition	Total Hours	Total Seats	Flight Rules	Category	Propulsion	No. Engines	Make	Model	Country	Continent	National Origin
Price	1.000	0.043	0.309	0.412	-0.017	0.049	0.324	0.064	0.601	0.329	0.677	0.951	0.415	0.273	0.419
Currency	0.043	1.000	0.071	0.226	0.094	0.061	0.141	0.169	0.128	0.120	0.572	0.475	0.538	0.349	0.233
Year	0.309	0.071	1.000	0.365	-0.322	-0.004	0.106	0.269	0.311	0.031	0.744	0.967	0.252	0.181	0.269
Condition	0.412	0.226	0.365	1.000	0.110	0.110	0.342	0.250	0.350	0.228	0.516	0.457	0.356	0.365	0.207
Total Hours	-0.017	0.094	-0.322	0.110	1.000	0.255	0.076	0.045	0.214	0.267	0.576	0.860	0.204	0.107	0.213
Total Seats	0.049	0.061	-0.004	0.110	0.255	1.000	0.433	0.100	0.198	0.428	0.584	0.993	0.133	0.063	0.248
Flight Rules	0.324	0.141	0.106	0.342	0.076	0.433	1.000	0.238	0.354	0.274	0.541	0.748	0.238	0.233	0.250
Category	0.064	0.169	0.269	0.250	0.045	0.100	0.238	1.000	0.383	0.248	0.921	0.769	0.277	0.214	0.279
Propulsion	0.601	0.128	0.311	0.350	0.214	0.198	0.354	0.383	1.000	0.613	0.760	0.773	0.311	0.250	0.442
No. Engines	0.329	0.120	0.031	0.228	0.267	0.428	0.274	0.248	0.613	1.000	0.626	0.988	0.218	0.119	0.229
Make	0.677	0.572	0.744	0.516	0.576	0.584	0.541	0.921	0.760	0.626	1.000	0.789	0.109	0.193	0.943
Model	0.951	0.475	0.967	0.457	0.860	0.993	0.748	0.769	0.773	0.988	0.789	1.000	0.406	0.373	0.774
Country	0.415	0.538	0.252	0.356	0.204	0.133	0.238	0.277	0.311	0.218	0.109	0.406	1.000	0.976	0.166
Continent	0.273	0.349	0.181	0.365	0.107	0.063	0.233	0.214	0.250	0.119	0.193	0.373	0.976	1.000	0.125
National Origin	0.419	0.233	0.269	0.207	0.213	0.248	0.250	0.279	0.442	0.229	0.943	0.774	0.166	0.125	1.000

Figure 4.23: Correlation Matrix representing the magnitude and direction of the linear relationships of the different attributes included into the processed dataset.

Focusing on the correlation coefficients obtained for the *Price* target variable, it can be observed that the attribute showing the strongest relationship is *Model*, followed by *Make* and *Propulsion* features. This indicates that one of the most significant drivers of the cost is given by the propulsion system of the aircraft and technical specifications related to a given model. Other variables such as *Year*, *Condition*, *Flight Rules*, *No. Engines*, *Country*, and *National Origin* also indicate certain correlation to the target variable, making possible the extraction of patterns and tendencies for the refinement of the algorithms. On the contrary, *Currency*, *Total Hours*, *Total Seats*, and *Category*, although representing the direction of the relation, have a very low correlation magnitude, meaning that no significant relationships exist between these variables and *Price* attribute.

As will be seen in the next section, the insights derived from this bivariate analysis are of high importance for the selection of the features that will be used for the training of the algorithm, as the correct selection of variables can considerably impact the performance of the models.

4.3.6. Features Selection

At this point, after the implementation of several pre-processing techniques, some variables were already removed from the dataset, mainly due to their lack of quality and the large presence of inconsistencies within their values. This is the case of *Eng. 1 Hours*, *Eng. 2 Hours*, *Eng. 1 Maint.*, *Eng. 2 Maint.*, *Prop. 1 Hours*, *Prop. 2 Hours*, *Prop. 1 Maint.*, and *Prop. 2 Maint.*, which presented null values in the majority of their cells. In addition, *City* feature was also deleted from the dataset during the handling of missing values; however, this variable indicated a location property of the aircraft, which, althought not at the same level of granularity, is already represented with other geographical variables such as *State*, or *Country*.

Similarly, *S/N* and *REG* columns were deleted in the early stages of the pre-processing stage, as they did not provide any variability, having unique values for each of their cells, and therefore not being able to get any insight from these columns.

For the analysis and selection of the most relevant attributes within the remaining dataset, a filter approach was followed, taking advantage of the correlation matrix obtained during the EDA step, corresponding to Fig. 4.23. As observed previously, *Currency*, *Category*, *Total Hours*, and *Total Seats*, present a quite low correlation coefficient with *Price* column, meaning that the values in these columns are no meaningful for the prediction of the target variable, which may lead to the potential overfitting of the model.

Taking a deeper look into the correlation matrix, it can be observed that the majority of the variables have large correlation coefficients with *Make* and *Model* attributes, meaning that the knowledge of these variables can rapidly result in the prediction of the rest. However, the huge variability and categories associated with these features require the study of the effect of the rest of the variables, in case a newly introduced aircraft presents a different manufacturer or model than the ones included in the training dataset.

Additionally, although *Continent* presents a reasonable correlation with respect *Price* attribute, it shows a relationship close to one with *Country* column, not being necessary its presence as it is already inferred in another variable.

In this way, *Currency*, *Category*, *Total Hours*, *Total Seats*, and *Continent* were removed from the dataset that will train the model in subsequent steps.

Finally, as will be explored in more advanced sections, regularization and embedded methods will also be used in the case of certain algorithms, which allows the assessment of the reliance of the filter method applied, and the improvement of the feature selection process.

4.3.7. Feature Engineering

After the selection of the relevant data for the application under study, the next step addressed was the transformation of these variables into the most adequate format for the algorithm to be implemented. As outlined during the theoretical segment of this dissertation, feature engineering corresponds to the last stage of the pre-processing of data and is highly dependent on the model to be trained. Additionally, the selection of the different feature transformation techniques can significantly impact the performance of the model, having a critical role during the implementation of machine learning applications.

For these reasons, an iterative process was established to assess the impact of the different feature engineering techniques for the various approaches considered as possible candidates for the development of the application: *Linear Models*, *Tree-Based algorithms*, *K-Nearest Neighbors*, *Support Vector Regression*, and *Artificial Neural Networks*, as will be more deeply reviewed in next chapter. In this way, a baseline model was established for each of the models, which was iterated with different approaches of the three principal feature engineering techniques: encoding of categorical variables, scaling of data, and feature transformations.

The evaluation of each of the iterations was performed by the computation of a series of general model evaluation metrics, which were compared with those of the corresponding reference case.

The next paragraphs present the main outcomes achieved from the implementation of this iterative process, allowing the identification of the most suitable feature engineering techniques for the different models considered. Additionally, App. B includes a detailed view of the process followed and the evaluation metrics obtained for each of the iterations.

Selection of Categorical Encoding Techniques

The analysis of encoding techniques was addressed by the iteration of the reference models with a series of codification approaches, including the *One-Hot encoding*, *Ordinal encoding*, *Binary encoding*, *Count encoding*, and *Target encoding* methods, obtaining results presented in App. B.1.

In general terms, the impact of encoding techniques was very similar in the different models implemented, showing a global tendency in their performance. Below, the different approaches assessed are ordered from lowest to highest precision achieved:

$$\text{Count Encoding} < \text{Ordinal Encoding} \approx \text{Binary Encoding} < \text{One-Hot Encoding} < \text{Target Encoding}$$

In the case of count encoding, the magnitude of the errors obtained was approximately double that of those obtained with ordinal and binary encoding. This technique is implemented by assigning each value the frequency it is repeated within the feature, without capturing any relationship of the attribute with the target variable, and possibly introducing a bias into the model. This effect is especially relevant when a higher predominance of certain values is present, making less frequent categories have smaller values than those that are more frequently occurring. After the data exploration performed in previous steps, it was observed that a high predominance of some values within some attributes is common to the dataset under study, discarding this method as a feasible encoding technique.

A similar issue was faced by ordinal and binary encoding, which assigns a different value for each category. This can be interpreted by the model as an ordinal relationship between variables, being particularly significant when an extensive list of categories is in place, and when used in algorithms that present some sensitivity towards data scale. Although binary encoding reduces the arbitrariness of the numeric values assigned during the ordinal encoding, both methods can generate artificial relationships among variables.

On the other hand, the one-hot encoding method considerably improved metrics over previous methods. In this case, as the model does not provide any artificial values to the different categories, no misleading pattern was introduced. The limitation of this technique is normally associated with its high computational cost and the high-dimensional resulting dataframe.

Nevertheless, the method that demonstrates the most favorable evaluation metrics is target encoding, due to its high efficiency for encoding features with certain correlation, as in the case of the dataset under study. This method supports the capture of the relationships between the predictor variables and the target feature, facilitating the subsequent training of the model. However, this approach is prone to overfitting if not correctly applied, for which some regularization or validation techniques are required.

Although the general tendency was observed for the majority of the models, some particularities were detected:

- In the case of the tree-based algorithm, although the encoding techniques followed the global tendency, the differences between one-hot, ordinal, binary, and count encoding were not as relevant as in the previous case. The reason behind this is that models based on decision trees are most robust to codification techniques; indeed, when using some specific libraries, tree-based models may be able to work with data frames containing categorical features.
- The results provided by the SVR model correspond to an exception to the global tendency previously commented on. In this case, any of the encoding approaches evaluated did not result in reliable models,

presenting negative values for the R-squared parameter. This behavior indicates that the model does not adequately capture the relationship between the predictor and target variables, providing larger errors than the ones that would result from just using the mean of the predicted feature or being overfitted to the training data. In the case of the target encoding, the model did not achieve convergence before an established time boundary. This exhibited a larger computational effort for the implementation of the algorithm, not covered by the computer capabilities available.

- Finally, for the ANN model, the one-hot approach lost efficiency and showed a similar performance to that of the binary encoding. In contrast with other methods, in a neural network, each neuron learns patterns from the combination of inputs. When one-hot encoding is used, a large number of these inputs correspond to null values, which may result in an inefficient propagation of the signal in hidden layers, requiring more training stages and still not being able to capture complex relationships.

Selection of Feature Transformation Techniques

Following the same methodology as for encoding techniques, different feature transformations on the *Price* variable were assessed to detect the most appropriate one for the implementation of the different models considered, including: *Logarithmic*, *Reciprocal*, *Square*, *Square Root*, and *Box-Cox* transformations.

In general terms, distinct transformations of the target variable seemed to improve the model accuracy for all of the different approaches considered, such as the logarithmic, square root, and Box-Cox transformations, as they normalize right-skewed data while reducing the impact of outliers. However, in the majority of cases, when the error margin was reduced, the R-squared metrics also experienced a slight reduction, making it necessary to achieve a balanced trade-off between accuracy and the variation that the model can handle.

On the other hand, the squared transformation exhibited the poorest results for the majority of the models, being in accordance with what was predicted, as this transformation is intended for left-skewed distributions, which is not the case for the *Price* variable.

The reciprocal transformation is of special interest as it degrades the performance of linear models, SVR, and neural networks, probably due to the exaggeration of small values induced by this technique, making the model to excessively adjust to certain data while losing generalization capabilities. However, it boosts the accuracy of the KNN and tree-based models, which normally struggle with capturing small values.

After the analysis of these global tendencies applied to the particular models, the following outcomes were derived:

- In the case of the linear regression model, after considering the metrics obtained for the different transformations, the logarithmic one was selected, as it was the one maximizing the R-square parameter, while considerably reducing the error.
- On the contrary, although some enhancement was detected for tree-based, KNN, and ANN approaches, it was decided not to apply any transformation to the target variable of these models, as the magnitude of these improvements was not relevant compared with the added complexity associated with transformation techniques.
- The effect of transformation methods in the SVR model was of special interest, as some of the approaches significantly improved the quite inefficient baseline model. In general terms, the same global tendency that for the rest of the models could be observed, however, in this case, the logarithm transformation was the one having the best trade-off between prediction errors and generalization capabilities.

Selection of Scaling Techniques

Finally, the last feature engineering technique to be explored for the dataset under study was the scaling of its attributes, including both normalization and standardization approaches. With this scope, the different models were iterated with a series of these techniques, including the *Standard scaler*, *Min-Max scaler*, *Max-Abs scaler*, *Robust scaler*, *Normalizer*, and *Quantile transformer*.

The general tendency followed by the scaling mechanisms is more complex to assess than in previous transformations. In global terms, the quantile transformer method exhibited the most significant improvements in error reduction and increased generalization capacity of the algorithms, showing the advantage gained from transforming data into a Gaussian distribution, especially for linear models. Robust scaler also provided good results, making evident the benefit of the models from being trained with data in which the impact of outliers is reduced.

On the other hand, the normalizer technique caused convergence issues, generating extreme inaccuracies for some of the models, probably because it represents an inappropriate kind of scaling for the majority of the models, as it works by scaling data samples within a vector but does not help with feature-wise transformations.

In this case, many particularities were detected depending on the model used:

- For linear models, the min-max scaler, max-abs scaler, and robust scaler did not provide any improvement in comparison with the baseline model, as these methods just escalate de coefficients of linear regression. On the other hand, the standard and normalizer approaches did not converge, with the quantile transformation being the only one that considerably reduced prediction error.
- In the case of tree-based models, really similar evaluation metrics were obtained for the different scaling techniques, as these model approaches do not use Euclidean distance and do not depend on the scale of the data. Just a slight improvement was detected with the quantile transformer, probably because of its capability to mitigate the effect of outlier values and achieve a more uniform distribution, improving the stability of decision divisions.
- In general terms, the KNN model improved its accuracy with all of the different scaling techniques. However, its influence was much lower than foreseen, probably because of other constraints that can restrict the refinement, such as the presence of outliers. Because of the lack of significant impact, no scaling technique was used for the KNN model.
- The SVR experiences certain degradation concerning the reference model with the use of min-max scaler, max-abs scaler, and normalizer, showing that contrary to what was expected, these techniques do not provide a suitable data distribution to the highly-distance-dependent SVR model. However, the standard scaler, robust scaler, and quantile transformation showed a stabilization and improved accuracy for the model, with the quantile transformation being the method providing the best results.
- In the case of the neural network, although the quantile transformer provides the smallest error, being selected for the scaling of data in the ANN model, it is not the absolute winner in terms of generalization. This may be caused by an excessive transformation that can limit the capture of complex relationships, instead, the robust scaler provides better R-squared metrics.

After the iterative evaluation process performed for the different feature engineering techniques and the derivation of insights from the analysis of the evaluation metrics obtained, Tab. 4.11 consolidates the different transformations implemented on the training data for the subsequent construction of the models:

Models	**Encoding techniques**	**Target transformation techniques**	**Scaling techniques**
Linear	Target Encoding	Logarithm Transformation	Quantile Transformer
Tree-Based	Target Encoding	-	Quantile Transformer
KNN	Target Encoding	-	-
SVR	One-Hot Encoder	Logarithm Transformation	Quantile Transformer
ANN	Target Encoding	-	Quantile Transformer

Table 4.11: Most optimal encoding, *Price* transformation, and scaling techniques based on available training data for the performance of the different regression models under scope.

After the analysis of the different feature engineering techniques in the different models, it has become evident the relevance of this step. On the one hand, significant performance improvement was observed with the application of certain approaches with respect to the baseline case. But, on the other hand, it was detected that not all the techniques worked as expected according to theoretical notions, highlighting the relevance of particularising the different methods to the data under study and the analysis of their specific effect, leveraging the subsequent implementation of the algorithms.

5 Results

The present chapter is focused on presenting the results obtained after developing and assessing a series of regression algorithms for the application under scope: the prediction of aircraft prices. For this purpose, the following sections will provide an overview of the procedure followed for the construction and optimization of the different approaches considered, presenting the set of hyperparameters that yields the best performance for each of the different models. Additionally, a series of evaluation metrics will be described for each of these optimized algorithms, allowing their comparison in terms of predictive precision and generalization capacity.

The analysis of the models and their evaluation metrics will enable the selection of the most suitable approach for the study, while allowing the detection of model and data particularities and the identification of possible areas for improvement.

5.1. Models' Building Procedure

As already anticipated in the previous section, the models considered as possible candidates for the implementation of the aircraft price prediction tool are those presented in Sec. 3.4.1 of this dissertation: *Linear Regression, Decision-Tree, K-Nearest Neighbors (KNN), Support Vector Regression (SVR),* and *Artificial Neural Networks (ANN)* algorithms.

Going one step further, regularization techniques were considered on top of the simple linear regression model, for which *Lasso*, *Ridge*, and *Elastic Net* models were implemented, aiming to reduce the impact of a possible colinearity of the attributes conforming the training dataset. Additionally, embedded techniques based on the decision-tree model were addressed, such as the *Random Forest* and *Gradient Boosting* approaches. Their objective was to improve the model's efficiency, precision, and reduce the risk of overfitting concerning a plain decision-tree algorithm.

The majority of these models require the selection of a series of hyperparameters for their construction, having a significant impact on the correct training and performance of the algorithm. For this reason, the election of these hyperparameters was performed during the training of the algorithms by the implementation of *Grid Search* techniques, which evaluate the model on different values of these parameters at the same time that a *K-Fold-Cross-Validation* process is performed, resulting in a robust and reliable optimization of the models.

Finally, the fine-tuned algorithms were evaluated by the computation of a series of evaluation metrics, including the *Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Median Absolute Error (MedAE), Mean Absolute Percentage Error (MAPE)*, and the *R-squared (R^2)* metrics. The choice of these parameters was based on the intention to assess the model performance in terms of prediction accuracy and the capability of generalizing unseen samples, while providing easily-interpretable figures.

In this way, the process of algorithm optimization and evaluation required the division of the dataset into different subsets:

- The 80% of the dataset was used for the training, selection of hyperparameters, and refinement of the algorithms through the implementation of grid-search and k-fold-cross cross-validation techniques. Indeed, this 80% of the data was split into five different subsets, four used for the training and one for the validation of each of the fine-tuning iterations.

- The remaining 20% of the data was kept apart from the training of the algorithm, allowing the subsequent testing and evaluation of the model with unseen observations.

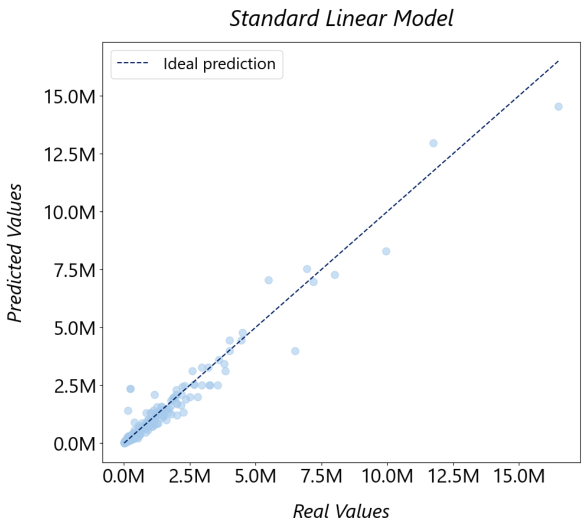
5.2. Models' Assessment

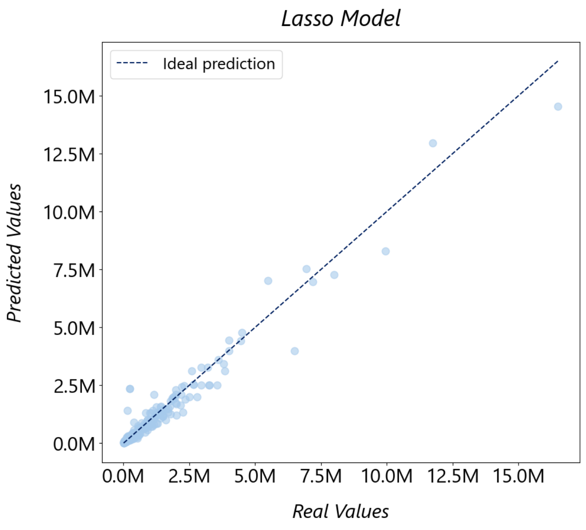
After the implementation of the different regression models described in the previous section, the complete overview of the results obtained can be found in Tab. 5.1 below:

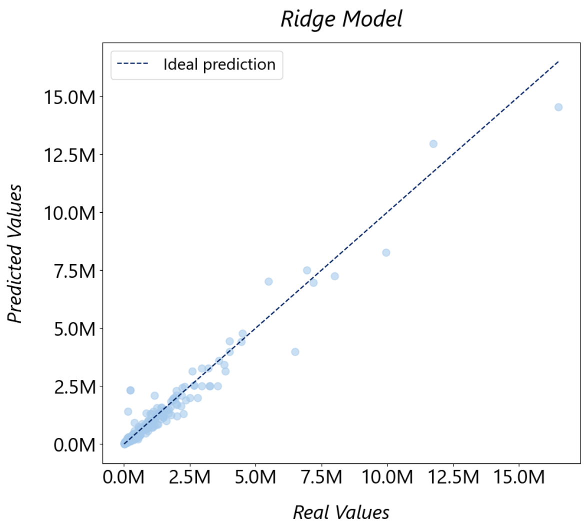
Model	**Hyperparameters**	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
Standard Linear	-	113,067	298,358	27,148	26.56 %	0.9561
Lasso	$\alpha_L = 0.0001$	112,961	298,027	27,293	26.56 %	0.9562
Ridge	$\alpha_R = 1.2068$	113,047	297,874	26,409	26.55 %	0.9562
Elastic Net	$\alpha_{EN} = 0.0010$ $l_1 = 0.1$	112,869	297,567	27,475	26.53 %	0.9563
Decision Tree	*max_depth* = 10 *min_samples_leaf* = 2 *min_samples_split* = 2	122,142	327,207	30,000	24.82 %	0.9490
Random Forest	*max_depth* = 10 *max_features* = "auto" *min_samples_leaf* = 5 *min_samples_split* = 2 *n_estimators* = 200	110,297	289,837	30,059	23.81%	0.9585
Gradient Boosting	$\alpha_{LR} = 0.01$ *max_depth* = 3 *min_samples_leaf* = 2 *min_samples_split* = 10 *n_estimators* = 500	124,016	352,803	27,081	23.24%	0.9386
K-Nearest Neighbors	*algorithm* = "auto" $k = 5$ $p = 1$ *weights* = "distance"	144,237	390,789	41,070	36.18%	0.9272
Support Vector Regression	$C = 10$ $\epsilon = 0.2$ γ = "scale" *kernel* = "rbf"	287,329	791,001	86,653	89.63%	0.6912
Artificial Neural Networks	*activation* = "relu" $\alpha_{ANN} = 0.01$ *hidden_layer_sizes* = (100,) α_{LR} = "constant" *solver* = "adam"	137,423	376,295	30,418	25.45%	0.9301

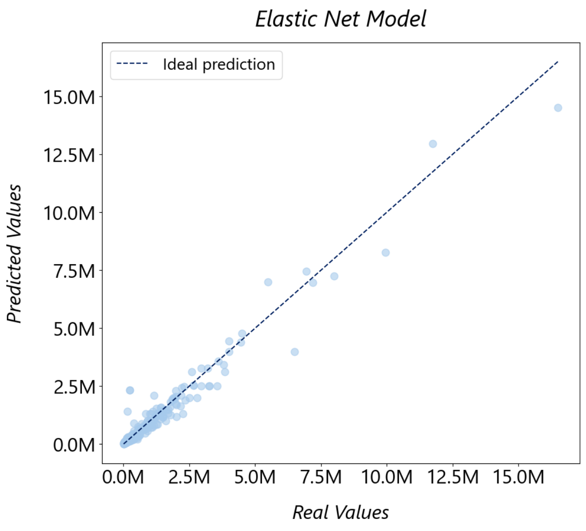
Table 5.1: Optimal hyperparameters and their associated evaluation metrics (MAE, RMSE, medAE, MAPE, and R-squared) for the different regression models evaluated: Standard Linear Regression, Lasso, Ridge, Elastic Net, Decision Tree, Random Forest, Gradient Boosting, K-Nearest Neighbors, Support Vector Regression, and Artificial Neural Network.

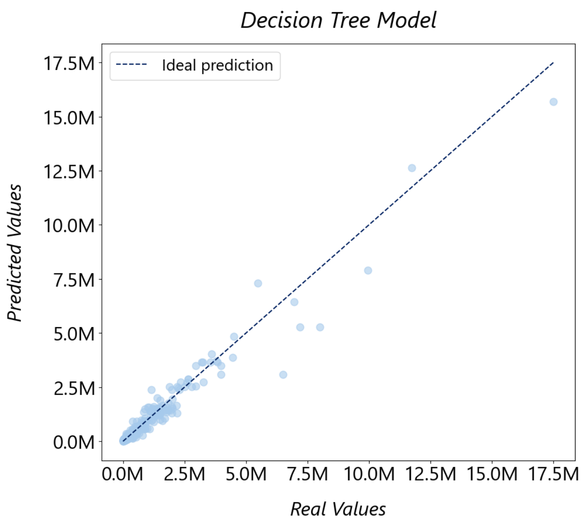
Apart from the calculation of these evaluation metrics, Fig. 5.5 includes a series of scatter plots comparing the values predicted by the algorithms and the actual values for the different models. These plots provide a more representative and intuitive assessment of the performance of approaches, as they include a dashed diagonal line which represents the location where perfect predictions should lie (i.e., when predicted values equal actual values). In this way, if data points are not within this line, the prediction error can be visually derived as the distance of that observation to the line.

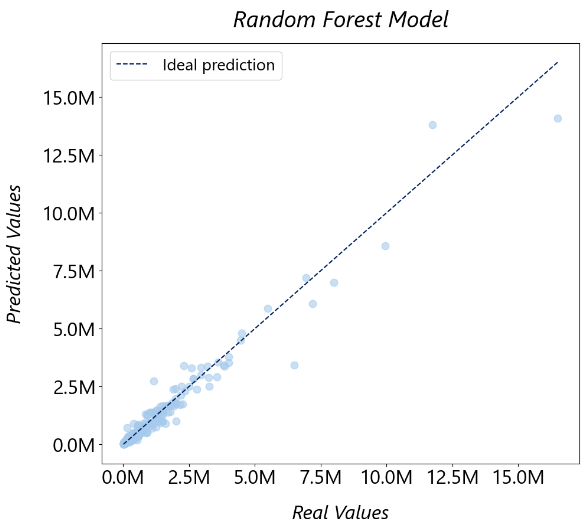

Standard Linear Model
Ideal prediction
Predicted Values
Real Values
15.0M
12.5M
10.0M
7.5M
5.0M
2.5M
0.0M


Lasso Model
Ideal prediction
Predicted Values
Real Values


Ridge Model
Ideal prediction
Predicted Values
Real Values


Elastic Net Model
Ideal prediction
Predicted Values
Real Values


Decision Tree Model
Ideal prediction
Predicted Values
Real Values
17.5M


Random Forest Model
Ideal prediction
Predicted Values
Real Values

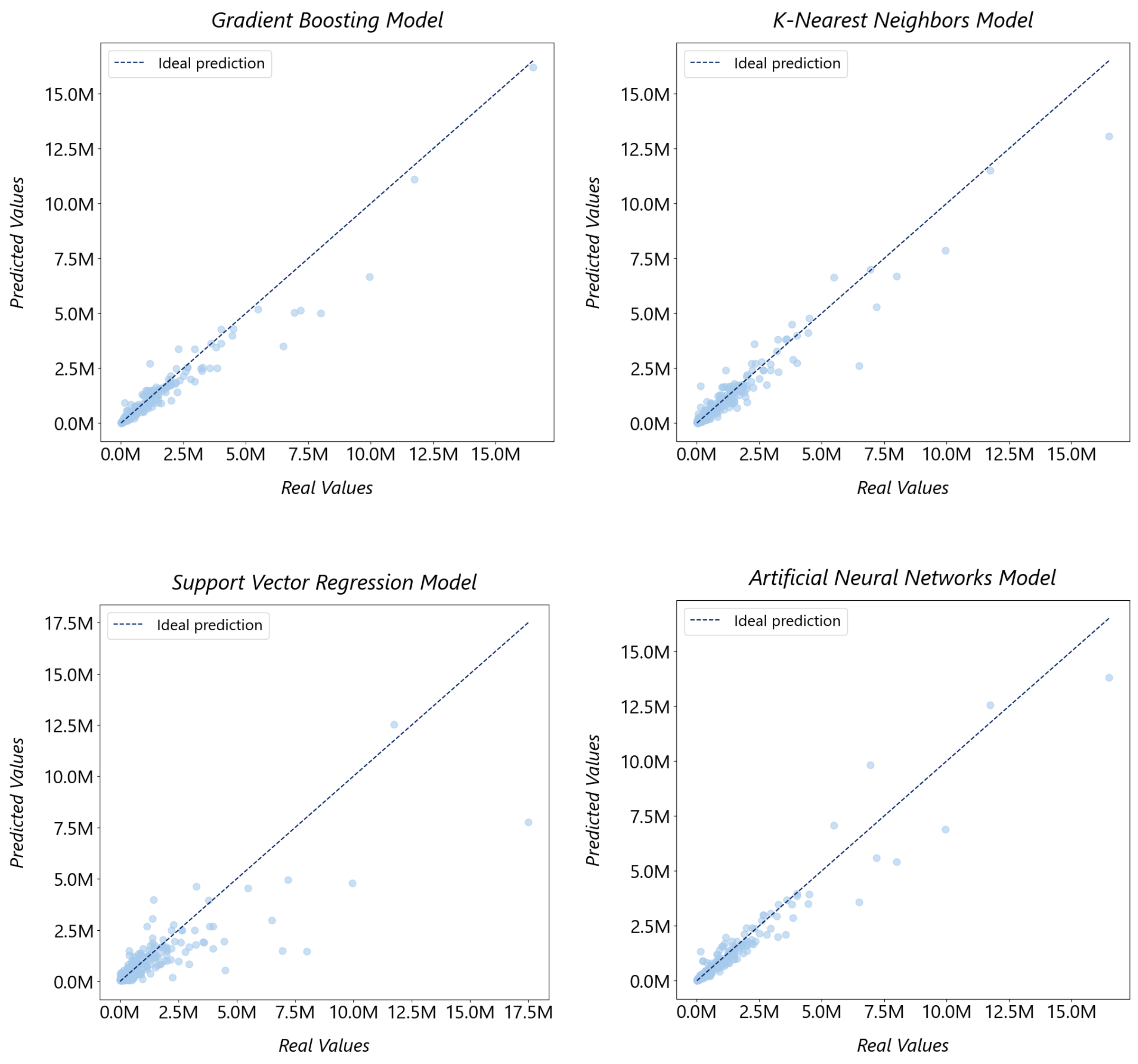


Figure 5.5: Scatter plots comparing actual and predicted values for each of the regression models assessed, showing the line of ideal prediction and the point prices predicted. Any distance deviation from the line of ideal prediction indicates the model's error.

Additionally, the stability and robustness of the models were validated by the plotting of the MAE, MSE, and R-squared learning curves for both train and validation datasets, which can be consulted in the App. C. These metrics provided the expected insights for well-based models, showing a stabilizing tendency towards a reduction of prediction errors and an increase of generalization as the size of the training dataset grows. In this way, the consistency and reliability of the different models were corroborated.

5.3. Discussion of Results

Taking a look at the linear models implemented, it can be observed that although they do not provide the most accurate predictions, they present relatively low errors and a solid capability for the handling of variance, represented with an R-squared parameter close to one.

By focusing on the regularization techniques applied, it is detected that the most optimal value of α in the Lasso approach is close to 0 ($\alpha_L = 0.0001$), indicating that no variables have been removed from the training dataset, meaning that all the attributes have been found relevant by the algorithm and that it behaves similar to a standard linear regression, as can be observed in its evaluation metrics. In the case of the Ridge approach, a moderate regularization is observed with a $\alpha_R = 1.2$, meaning that a certain reduction is applied to the linear coefficient associated with some particular attributes. This can be caused by some colinearity within the data, however, its impact is not relevant, as can be observed in the similarity of the Ridge's evaluation metrics concerning the plain linear regression.

Finally, the results from the Lasso and Ridge regularizations are complemented with those of the Elastic Net approach. In this case, a $\alpha_{EN} = 0.001$ and a $l_1 = 0.1$ are obtained, validating what was extracted from previous methods. The value of the l_1 ratio indicates that only 10% of the regularization is coming from the Lasso approach, while 90% is provided by the Ridge technique. In addition, the level of regularization is quite low. This indicates that the model tends to soften the coefficients instead of eliminating variables, looking for a balance between bias and variance.

Examining now those approaches corresponding to tree-based algorithms, it can be rapidly detected that they provide the most accurate predictions, although in some cases, the generalization capability is degraded.

Concerning the standard Decision-Tree, it can be observed that the most optimum hyperparameters found for its construction define a flexible tree structure capable of learning complex patterns but with a subtle control of overfitting. This can be observed with an intermediate value for its maximum depth ($max\_depth = 10$), which shows that the model can capture relatively complex trends without becoming extremely profound. As well, a value of 2 for both *min_samples_leaf* and *min_samples_split* indicates that a fine adjustment is obtained but avoiding leaves with a single sample, trying to control overfitting.

Focusing now on the Random Forest approach, it can be observed that the average of the results obtained from a series of parallel trees improves those metrics obtained with a standard Decision Tree model, both the accuracy and the model's flexibility. This mentioned improvement is achieved by the construction of 200 trees with a similar configuration to that of the standard model; the combination of this number of trees allows the model to be more resistant to anomalies and contributes to the reduction of overfitting. Moreover, the automatic selection of features allows the reduction of randomness and the robustness of the model.

In the case of the Gradient Boosting method, it provides the smallest MAPE within all the approaches studied; however, a small reduction of the R-squared metric suggests a subtle degradation of its generalization capability. These results are obtained for the subsequent implementation of 500 different trees, which, although less complex than in the Random forest approach, learn at a quite low rate of 0.01. This results in a high computational cost compared to previous models.

Taking a look at the remaining models, the K-Nearest Neighbors, Support Vector Regression, and Artificial Neural Network, it was observed that, although corresponding to more sophisticated and computationally demanding approaches, their performance was not improved.

The most accurate predictions in the case of the KNN model were obtained with the consideration of the Manhattan distance ($p = 1$) for the selection of the 5 nearest neighbors using the best search strategy in each case, additionally, it provided more relevance to those samples closest to the one to be predicted. Despite its optimization, this model deteriorated the performance compared with previous approaches. This behavior can be attributed to its low generalization capabilities when datasets with a considerably high dimensionality are used, as the distance between points can lose its meaning.

In the case of the SVR model, the least favorable outcomes were identified. It can be observed that the model tries to adjust well to the data despite the possibility of losing some generalization capabilities, as can be derived from the considerably high value of $C = 10$. The same tendency can be observed with the choice of a low tolerance margin, ϵ, that only allows predictions within an error margin of 0.2 times the real error. Additionally, the model uses a RBF (Radial Basis Function) kernel, allowing the capture of non-linear relationships between variables. This can indicate that data is affected by some non-linear tendencies, also the poorest results of the SVR model can be caused by the capture of more subtle patterns that may not be necessarily relevant for the prediction.

Finally, taking a look at the Artificial Neural Networks approach, the selection of ReLU (Rectified Linear Unit) as the activation function also suggests that the model needs to capture non-linear relationships, which is aligned with what was derived from the SVR model. Additionally, a $\alpha_{ANN} = 0.01$ is also consistent with what was observed during the analysis of the linear models, in which a very limited level of regularization is used, without significantly penalizing coefficients. On the other hand, the use of 100 neurons together with the use of the Adam solver may indicate that the problem is influenced by some complexity, such as irregular gradient or some noisy data, although a single hidden layer is needed.

6 Conclusion and Further Steps

Throughout this text, the procedure for implementing a machine learning algorithm capable of predicting aircraft prices given a set of features has been explored. The primary motivation behind this study stemmed from the limited aircraft depreciation techniques normally employed by aeronautical stakeholders, finding an opportunity to construct a tool that accurately determines aircraft prices based on historical data.

After the evaluation of different regression algorithms, it was observed that the tree-based family provided the most accurate predictions, with the *Random Forest* approach being the one achieving the best balance between accuracy and variability. This model accomplished the lowest MAE error with a value of 110,297, and the largest R-squared parameter equal to 0.9585. Additionally, it corresponds to an easily constructible algorithm, with a low computational cost relative to the other methods.

The outstanding performance of the tree-based algorithms in comparison with the linear approaches suggests that, although limited, there exists a certain amount of non-linear relationships or complex patterns between the variables training the model, not being able to be captured by linear algorithms. On the other hand, by taking a look at more complex models such as the KNN, SVR, and ANN, although these approaches can also handle non-linearity, they require a thorough selection of hyperparameters and pre-processing techniques, which hinder the correct implementation of the models in comparison to tree-based algorithms.

Additionally, as derived from the pre-processing of the dataset, a relatively high number of features were included within the input data; this high-dimensionality can distort distances between points, degrading the performance of distance methods such as the KNN or SVR approaches through the so-called *curse of dimensionality*. Furthermore, the training dataset included some observations that, although perfectly feasible, represented non-common aircraft, which may be identified as outliers or induce some noise in the training of models like the SVR or ANN.

In this way, the Random Forest is the method that best recognizes patterns in the training set under scope, positioning itself as a model capable of handling non-linear relationships and tolerating high-dimensionality, noise, and outliers as a result of its elevated robustness. All of this, without requiring the implementation of complex pre-processing techniques and with low training and optimization times.

However, although the Random Forest outperformed other regression models, its predictive capability can be found to have some limitations with a MAPE metric of 23.81 %, showing a considerable deviation when predicting aircraft prices. After exploring the insights derived from the pre-processing stage, it can be inferred that the most probable cause of this constraint model's performance is associated with the quality of the training data:

- The 70% of the observations correspond to single-piston-engined airplanes located in the US and evaluated in dollars. However, the scope of the study was not limited to this typology of aircraft, resulting in the skewness of the target variable towards the right and the possible identification of a high number of outliers by the model. Although this tendency was tried to be mitigated through some feature transformation techniques, this bias in the information hinders the price prediction of those aircraft that lie outside the most frequent features.

- The propulsive configuration of aircraft was found to be one of the most relevant variables impacting the price of aircraft. However, due to the high number of missing values present in the columns indicating the usage and status of the engines and propellers, these features had to be deleted, losing information about the propulsion condition.
- Finally, the training data did not contain information about some component that could significantly impact the aircraft prices, such as electric systems, hydraulic systems, avionics, engine specifications, or any other onboard technology. This lack of information makes the model predict based on more general descriptions, such as the aircraft model, which was found to be the most relevant feature, but it does not provide particular details, which limits predictions' accuracy.

In this way, the limitation of the model's accuracy can be largely attributed to the quality of the data used for the training of the algorithms, which, although deeply processed, presented incomplete, biased, and moderately noisy information.

For this reason, as potential future steps, efforts should be focused on the acquisition of a higher-quality training dataset, representing the complete catalogue of general aviation aircraft, including detailed and consistent information about more significant aircraft components.

Apart from the improvement of data quality based on the acquisition of a more consistent and detailed dataset, the pre-processing step could also be enhanced by the implementation of some advanced techniques. An area of improvement can be found regarding the handling of null values, where additionally to the common imputation methods, some regression algorithms could be used such as the *K-Nearest Neighbors Imputation (KNNImputer)* or the *Multivariate Imputation by Chained Equations (MICE)*, which computationally estimate null values based on other observations in the dataset, detecting hidden patterns.

Additionally, although a low regularization coefficient was obtained for Lasso, Ridge, Elastic Net, and ANN approaches, indicating that a good selection of features was performed during the pre-processing stage, some more advanced techniques could be implemented for the detection and treatment of multicollinearity among variables. An example of more sophisticated feature selection method is the *Principal Components Analysis (PCA)*, corresponding to a highly powerful filter technique which combines meaningful features, reducing data dimensionality, at the same time that keeping the largest amount of variance within the data, and applying to all of the algorithms.

From a model implementation perspective, some further methods could be applied for the enhancement of the model's performance. For example, due to the high number of hyperparameters to be selected for the different algorithms, the *Bayesian Search* could be implemented instead of the standard grid search. This technique, instead of evaluating the set of hyperparameters defined by the data scientist and selecting the most accurate combination, makes preliminary intelligent decisions about the most optimal combinations of hyperparameters before their evaluation on the models. In this way, it significantly influences model performance by selecting the appropriate hyperparameters for the problem under study, at the same time that the computational cost is reduced, which can eliminate the convergence issues faced during the implementation of some of the algorithms.

After this discussion, it can be concluded that although the study shows some limitations in the predictions, it is noticeable that the selected model is capable of capturing relationships, allowing the prediction of aircraft prices within a tolerable range for the majority of the observations. This demonstrates the strong potential of regression machine learning models for supporting aircraft valuation and decision-making activities in the aeronautical industry. Indeed, a detailed description of the benefits and social implications of this study can be found in Ch. 7.

7 Environmental, Social and Economic Sustainability

Finally, this chapter will aboard the environmental, social, and economic aspects surrounding the development of this study, focusing on both the impact caused during the development of this project and the possible effects that the implementation of the model could have on our society.

7.1. Environmental Impact

As explored during previous chapters, the development of this dissertation has been based on the implementation of artificial intelligence models, which, although hailed as a revolutionary technology, poses a concerning risk: its huge environmental impact. Indeed, the development, implementation, and maintenance of predictive algorithms are characterized by a high consumption of energy resources and electronic components. The next paragraphs will provide more detailed descriptions of these environmental risks.

AI's Energy Consumption

The training of machine learning algorithms can be very intensive in terms of energy consumption, especially for those models that involve advanced algorithms, such as deep neural networks. In these cases, powerful GPUs (Graphical Processing Units) and TPUs (Tensor Processing Units) may be required, demanding a higher computational cost and an associated increase in the energy required. For more complex applications, the energy consumption can even become higher, this is the case of algorithms requiring long times to be trained due to high-dimensional problems, or applications requiring real-time predictions. Additionally, although this is not the case in the study, models can also be implemented in cloud servers whose infrastructure is located within data centers; in those occasions, the energy and water consumption associated with the cooling of the systems also needs to be considered.

This energy expenditure produces an inevitable release of greenhouse gas emissions, which is accentuated as the complexity of the model increases. Indeed, some research carried out by OpenAI highlights that since 2012, the power used to train algorithms has doubled every 3.4 months, being even more relevant in the last years [120]. In addition, some studies estimate that the Information and Communications Technology (ICT) sector will contribute to the 14% of the total CO_2 emissions by 2040 [65].

The relevance of AI emissions can easily be visualized by taking a look at the investigation developed by the University of Massachusetts that compares the amount of CO_2 released by the training of some complex AI models with other relevant sources of CO_2 emissions, represented in Fig. 7.1 [104].

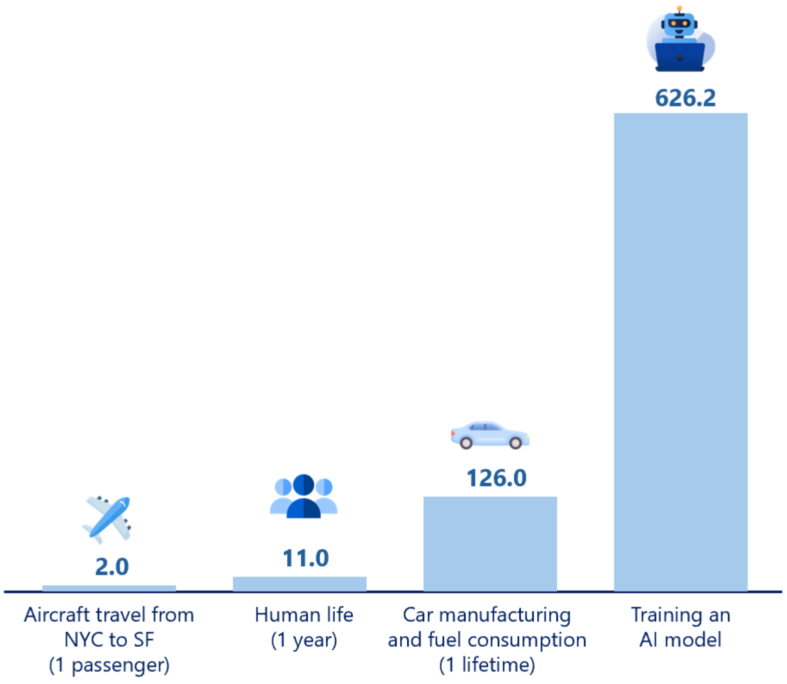


Figure 7.1: Comparison of CO_2 emissions footprint in thousands pounds (10^3 lbs) for aircraft travel, average human life, car manufacturing and usage, and the training of an AI model.

The development of the work of this study was not at the scale of those large commercial AI models; however, what was described in the previous paragraphs was rapidly observed during the training of the model. In some cases, some approaches required a couple of hours to be trained, with the computer constantly connected to an energy source, as the battery was consumed very rapidly. On top of that, it is also relevant to consider the computational time associated with the pre-processing of the data, because although those tasks were less computationally demanding, long hours were spent on them.

Electronic Components Expenditure

Apart from the high CO_2 footprint associated with the training of the model, AI can also harm the environment due to the large amount of processing and storage devices that it requires. Normally, these electronic components include materials such as lead, mercury, or cadmium, which correspond to non-renewable resources and pose an environmental risk during their extraction, production, and disposal [96].

The most critical phase during the lifetime of these hazardous chemicals is the one corresponding to their elimination, in which these so-called *e-waste* can pollute soil and water supplies, being detrimental to both human health and the natural ecosystems [97]. Indeed, a recent study performed by the World Economic Forum (WEF) estimates that the global amount of e-waste present in our environment will be above 120,000 metric tonnes by 2050 [23], which will have a special impact on those developing countries that lack recycling regulations. This provides insights into the environmental footprint of the electronic components used in machine learning applications, which is often not considered when developing these models.

For the concrete case of this study, a personal device was used, not requiring the utilization of specialized workstations. However, although the computational equipment was not so intensive, it still contained some detrimental materials that will need to be correctly managed during their disposal.

Under this context, it is clear that the implementation of machine learning models has a negative impact on the environment, however, this impact could be tolerated if it is balanced by the benefits they can offer.

In the case of the study, the machine learning algorithm is targeted to provide accurate price estimations for aircraft, which could play a substantial role in the optimization of aircraft valuation and lifecycle management. This facilitates decision-making regarding the acquisition, resale, and retirement of aircraft, which could reduce unnecessary manufacturing activities and waste, diminishing their associated CO_2 emissions. Additionally, more advanced and detailed models could improve the predictability of devaluation mechanisms, supporting strategies for aircraft disposal at the end of their life.

7.2. Socio-Economic Impact

In addition to the environmental impact, the development of AI solutions also involves relevant social and economic implications, which need to be considered and evaluated before the deployment of the applications.

Currently, AI tools are present in a wide variety of domains, such as health care, finance, education, and manufacturing, having the potential to significantly transform work dynamics by the optimization of tasks, the increase of efficiency, and the automation of labor-intensive operations.

Focusing on the ML model under the scope of this work, the prediction of aircraft prices positively supports numerous stakeholders within the aeronautical industry, as already anticipated in Sec. 1.1.2. The first main activity that could be positively impacted by the algorithm is the realization of buy-sell transactions, allowing private and corporate owners to accurately fix a price for the deal. Additionally, the accurate estimation of the aircraft price is a fundamental step during financial reporting activities, allowing the analysis of the profitability of aircraft assets and the identification of aircraft that need to be retired. Thus, the algorithm could provide the guidelines for the development of asset management strategies, with the objective of improving profitability and operational efficiency.

On the other hand, the model could also support tax deductions associated with the devaluation of aircraft, at the same time that assisting financial and insurance institutions in the assessment of risk and the offering of their solutions.

In general terms, the increased accuracy and transparency that this tool could offer might result in an enhanced market performance and competitiveness, promoting a more stable and predictable financial ecosystem for the aeronautical industry.

However, the rapid expansion of artificial intelligence has brought a wide range of subjects into public discourse. One of the most concerning topics is the possibility of employment displacement for certain technical functionalities, which may be totally or partially automated by intelligent agents. While AI can relieve professionals of repetitive activities and enable their focus on higher value-added activities, it may also require a reorientation of work capabilities, which may pose some challenges in terms of workforce adaptation, training, and equity when accessing the new opportunities.

Budget

The large computational work associated with the development of this thesis resulted in a non-significant consumption of resources. Thus, this project does not have a relevant capital investment associated; however, although minor, some expenses were incurred, mainly related to the time dedicated by the author and the computing tools used. In this way, two main budget pillars are identified:

Human resources

In terms of human resources, the author of this dissertation has been the main contributor and executor of the project, corresponding to a professional in their early career. Nevertheless, the recurrent involvement of the thesis tutor also needs to be taken into consideration, having a role as a senior expert in the aeronautical industry, and holding a position as professor at Universidad Europea de Madrid.

An estimation of the time dedicated by each of these resources has been summarized in the Tab. 7.1, allowing an approximation of the cost associated with human contributions. Note that the approximation of the cost per hour included for each of the profiles has been estimated based on studies performed by the Instituto Nacional de Estadística (INE) [14], and the research performed by the *Jobted* forum [69].

Role	Cost/hour [€/hour]	Dedicated time [hours]	Final Cost [€]]
Author	15	150	2,250
Tutor	38	15	570

Table 7.1: Cost of human resources associated with the development of this dissertation.

Computational resources

The development of this project was performed using the author's personal laptop, consisting of an Intel Core i9 processor, 16 GB of RAM, and a 512 GB SSD. The market price of this system was 1,100€, assuming a usable life of 5 years and a complete development time of 6 months, the project consumed 110€ related to this asset. No more costs were derived from computational needs, as the Python programming language was used in Visual Studio console, taking advantage of Jupyter Notebook development environments and some open-source libraries, all of which correspond to public solutions. Additionally, cloud storage and version control tools were used, but for all of them, only the free versions of the solutions were consumed. Furthermore, the writing of this report was performed in LaTeX in the TeXworks open editor, no assuming any further costs.

Finally, the power consumption was also determined by assuming high computational requirements (150W) during 75 hours for the model's training and medium requirements (65W) during the remaining 75 hours, taking into consideration the laptop specifications [5]. In this way, assuming a power cost of 0,20€/kWh, a total expense of 3,20€ was derived from the energy consumption, which is not a relevant contribution for the overall cost and can be neglected.

Accordingly to previous paragraphs, Tab. 7.2 includes a summarized view of the different costs associated with the development of the project and their final total amount:

Resources	Cost [€]
Author's dedication	2,250
Tutor's dedication	570
Laptop's usage	110
Total Cost	**2,930€**

Table 7.2: Project budget overview, including significant resources and their associated cost.

A

Developed Project's Material

The development of this thesis implied the building and production of different Python-coded scripts for both the pre-processing and model's implementation stages. With the objective of promoting transparency and guiding the construction of similar applications, the developed material has been stored in a public repository, accessible through this link.

B
Feature Engineering Impact Assessment

As covered throughout the body of this report, the selection of the most appropriate feature engineering techniques was performed by their isolated implementation and subsequent comparison with a baseline model for each of the algorithms considered. These baseline models, together with their obtained evaluation metrics, are included in Tab. B.1.

Model	**Hyperparameters**	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
Standard Linear	-	358,656	789,971	142,025	188.17 %	0.7025
Decision Tree	*max_depth* = 10 *min_samples_leaf* = 5 *min_samples_split* = 2	361,739	1,046,002	94,605	129,48 %	0.4783
K-Nearest Neighbors	*algorithm* = "auto" $k = 5$ $p = 1$ *weights* = "distance"	341,301	957,979	90,900	130.79%	0.5624
Support Vector Regression	$C = 10$ $\epsilon = 0.2$ γ = "scale" *kernel* = "linear"	529,692	1,480,734	128,456	147.51 %	-0.0454
Artificial Neural Networks	*activation* = "relu" $\alpha_{ANN} = 0.0001$ *hidden_layer_sizes* = (50,) α_{LR} = "constant" *solver* = "lbfgs"	508,304	112,8904	241,324	272.21 %	0.3924

Table B.1: Hyperparameters and their associated evaluation metrics (MAE, RMSE, medAE, MAPE, and R-squared) for the baseline models: Standard Linear Regression, Decision Tree, K-Nearest Neighbors, Support Vector Regression, and Artificial Neural Network.

The next sections present the evaluation metrics obtained for the different iterations performed on encoding, transformation, and scaling techniques. A detailed discussion and reasoning of the results obtained are included in Sec. 4.3.7.

B.1. Evaluation of Categorical Encoding Techniques

Model	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
One-Hot Encoding	358,656	789,971	142,025	188.17 %	0.7025
Ordinal Encoding	607,626	1,259,228	405,865	371.50 %	0.2440
Binary Encoding	550,951	1,164,370	266,465	300.76 %	0.3536
Count Encoding	668,506	1,292,208	501,677	455.63 %	0.2039
Target Encoding	153,677	345,939	72,098	85.28 %	0.9429

Table B.2: Effect of Encoding techniques in the performance of Linear Regression models.

Model	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
One-Hot Encoding	361,739	1,046,002	94,605	129.48 %	0.4783
Ordinal Encoding	398,842	1,083,125	88,031	162.93 %	0.4407
Binary Encoding	417,982	1,164,758	81,559	127.01 %	0.3532
Count Encoding	517,096	1,226,008	174,784	252.25 %	0.2833
Target Encoding	136,971	404,572	33,542	27.39 %	0.9220

Table B.3: Effect of Encoding techniques in the performance of Tree-Based models with *max_depth = 10*, *min_samples_ leaf = 5*, and *min_samples_split = 2*.

Model	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
One-Hot Encoding	341,301	957,979	90,900	130.79 %	0.5624
Ordinal Encoding	429,517	1,104,321	101,042	227.90 %	0.4186
Binary Encoding	378,054	1,093,974	81,660	127.58 %	0.4294
Count Encoding	552,134	1,307,879	171,990	238.12 %	0.1844
Target Encoding	144,237	390,789	41,070	36.18%	0.9272

Table B.4: Effect of Encoding techniques in the performance of KNN model with *algorithm = "auto"*, $k = 5$, $p = 1$, and *weights = "distance"*.

Model	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
One-Hot Encoding	529,692	1,480,734	128,456	147.51 %	-0.0454
Ordinal Encoding	520,562	1,472,383	122,341	146.71 %	-0.0336
Binary Encoding	529,535	1,480,369	128,464	147.65 %	-0.0449
Count Encoding	530,708	1,481,358	128,422	148.87 %	-0.0463
Target Encoding	-	-	-	-	-

Table B.5: Effect of Encoding techniques in the performance of SVR model with $C = 10$, $\epsilon = 0.2$, $\gamma =$ *"scale"*, and *kernel = "linear"*.

Model	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
One-Hot Encoding	508,304	112,8904	241,324	272.21 %	0.3924
Ordinal Encoding	564,743	1,332,571	231,310	274.57 %	0.1534
Binary Encoding	581,366	1,175,737	338,427	254.86 %	0.3409
Count Encoding	658,986	1,360,269	317,417	406.29 %	0.1178
Target Encoding	125,613	406,040	29,376	31.53%	0.9214

Table B.6: Effect of Encoding techniques in the performance of Artificial Neural Network model with *activation = "relu"*, $\alpha = 0.0001$, *hidden_layer_sizes = 50*, $\alpha_{LR} =$ *"constant"*, and *solver = "lbfgs"*.

B.2. Evaluation of Feature Transformation Techniques

Model	MAE	RMSE	medAE	MAPE	R^2
Logarithm	293,676	736,447	88,887	94.59 %	0.7414
Reciprocal	699,922	2,237,968	96,663	88.03 %	-1.3880
Squared	516,460	910,471	305,578	312.92 %	0.6048
Squared Root	311,618	753,384	93,507	123.71 %	0.7294
Box-Cox	300,716	791,688	91,357	89.55 %	0.7012

Table B.7: Effect of Transformation techniques in the performance of Linear Regression models.

Model	MAE	RMSE	medAE	MAPE	R^2
Logarithm	330,099	1,060,407	85,038	88.41 %	0.4639
Reciprocal	329,342	1,069,056	86,834	69.83 %	0.4551
Squared	414,345	1,081,808	131,583	205.87 %	0.4420
Squared Root	344,669	1,070,161	86,469	103.00 %	0.4540
Box-Cox	322,774	994,877	85,184	86.77%	0.5281

Table B.8: Effect of Transformation techniques in the performance of Tree-Based models with *max_depth = 10, min_samples_ leaf = 5,* and *min_samples_split = 2.*

Model	MAE	RMSE	medAE	MAPE	R^2
Logarithm	340,869	1,054,684	90,394	96.60%	0.4696
Reciprocal	356,608	1,156,594	85,813	79.87 %	0.3622
Squared	352,328	905,531	100,000	158.53 %	0.6090
Squared Root	338,112	1,000,822	86,605	113.39 %	0.5224
Box-Cox	342,318	1,068,304	91,011	93.36%	0.4559

Table B.9: Effect of Transformation techniques in the performance of KNN model with *algorithm = "auto"*, k = 5, p = *1,* and *weights = "distance"*.

Model	MAE	RMSE	medAE	MAPE	R^2
Logarithm	289,036	738,747	81,728	95.78%	0.7398
Reciprocal	658,871	1,591,049	204,462	93.37 %	-0.2069
Squared	565,791	1,523,402	134,485	152.32 %	-0.1065
Squared Root	380,599	1,234,922	87,221	112.54 %	0.2729
Box-Cox	352,661	1,309,670	90,638	86.68%	0.3807

Table B.10: Effect of Transformation techniques in the performance of SVR model with C = *10,* $\epsilon = 0.2$, γ = *"scale"*, and *kernel = "linear"*.

Model	MAE	RMSE	medAE	MAPE	R^2
Logarithm	431,507	1,246,828	108,146	131.20%	0.2588
Reciprocal	666,802	1,594,370	212,521	99.98 %	-0.2120
Squared	1,422,108	1,758,342	1,476,182	1,539.97 %	-0.4741
Squared Root	435,312	1,110,794	118,626	146.32 %	0.4117
Box-Cox	412,481	1,289,423	92,778	98.19%	0.2073

Table B.11: Effect of Transformation techniques in the performance of Artificial Neural Network model with *activation = "relu"*, $\alpha = 0.0001$, *hidden_layer_sizes = 50*, α_{LR} = *"constant"*, and *solver = "lbfgs"*.

B.3. Evaluation of Scaling Techniques

Model	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
Standard Scaler	-	-	-	-	-
Min-Max Scaler	358,656	789,971	142,025	188.17 %	0.7025
Max-Abs Scaler	358,656	789,971	142,025	188.17 %	0.7025
Robust Scaler	358,656	789,971	142,025	188.17 %	0.7025
Normalizer	-	-	-	-	-
Quantile Transformer	290,905	743,774	88,442	87.34%	0.7270

Table B.12: Effect of Scaling techniques in the performance of Linear Regression models.

Model	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
Standard Scaler	360,616	1,044,987	94,605	129.42%	0.4794
Min-Max Scaler	361,739	1,046,002	94,605	129.48 %	0.4783
Max-Abs Scaler	361,739	1,046,002	94,605	129.48 %	0.4783
Robust Scaler	360,616	1,044,987	94,605	129.42%	0.4794
Normalizer	370,015	1,080,106	90,225	123.70%	0.4438
Quantile Transformer	325,097	1,028,484	89,030	74.97%	0.4779

Table B.13: Effect of Scaling techniques in the performance of Tree-Based models with *max_depth = 10*, *min_samples_ leaf = 5*, and *min_samples_split = 2*.

Model	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
Standard Scaler	306,414	821,456	95,640	140.19 %	0.6783
Min-Max Scaler	299,986	843,042	82,880	127.04 %	0.6611
Max-Abs Scaler	305,871	854,668	92,870	131.45 %	0.6517
Robust Scaler	297,742	835,855	97,240	135.27 %	0.6669
Normalizer	303,489	849,820	94,460	131.82 %	0.6557
Quantile Transformer	300,782	883,404	94,107	89.22%	0.6148

Table B.14: Effect of Scaling techniques in the performance of KNN model with *algorithm = "auto"*, *k* = 5, *p = 1*, and *weights = "distance"*.

Model	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
Standard Scaler	337,533	742,220	145,914	179.57 %	0.7373
Min-Max Scaler	2,179,167	2,432,641	2,659,459	2,280.14 %	-1.8215
Max-Abs Scaler	2,162,192	2,411,777	2,732,955	2,256.33 %	-1.7733
Robust Scaler	339,997	832,999	95,078	124.94 %	0.6692
Normalizer	597,304	1,475,651	280,764	310.75 %	-0.0382
Quantile Transformer	281,724	697,925	83,337	93.35%	0.7596

Table B.15: Effect of Scaling techniques in the performance of SVR model with *C = 10*, $\epsilon = 0.2$, γ = *"scale"*, and *kernel = "linear"*.

Model	**MAE**	**RMSE**	**medAE**	**MAPE**	**R^2**
Standard Scaler	344,439	1,095,529	99,710	139.67 %	0.4278
Min-Max Scaler	322,570	776,537	125,027	138.83 %	0.7125
Max-Abs Scaler	351,360	772,332	123,067	168.66 %	0.7156
Robust Scaler	253,475	541,108	101,983	134.01 %	0.8604
Normalizer	733,053	1,448,219	537,565	582.52 %	0.0000
Quantile Transformer	295,413	853,675	83,097	81.59%	0.6403

Table B.16: Effect of Scaling techniques in the performance of Artificial Neural Network model with *activation* = *"relu"*, α = 0.0001, *hidden_layer_sizes* = *50*, α_{LR} = *"constant"*, and *solver* = *"lbfgs"*.

C
Models' Validation

This appendix includes the graphs obtained for the validation of the stability of the different models implemented and described in Sec. 5.

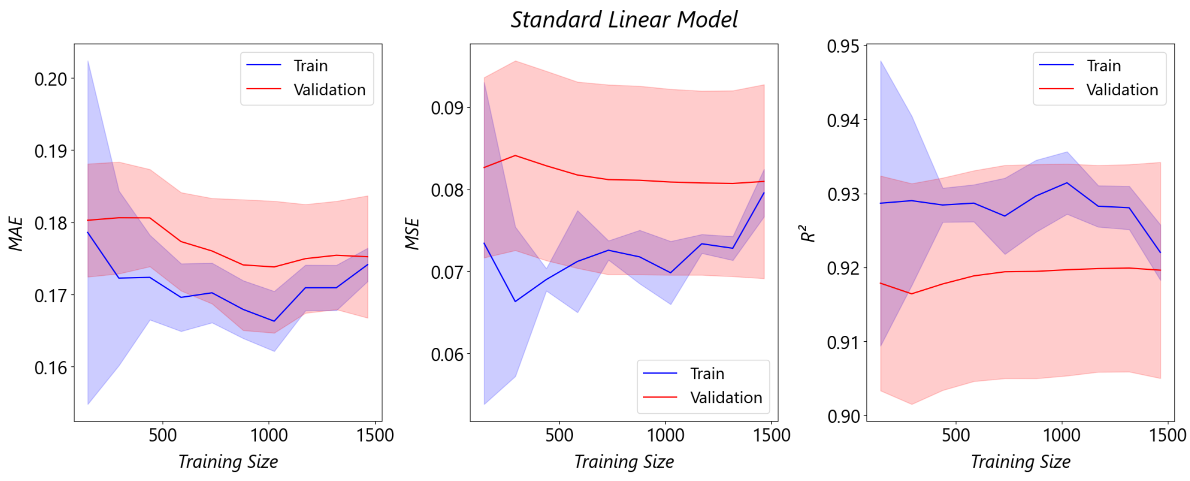


Figure C.1: MAE, MSE, and R-squared train and validation curves for the verification of the Standard Linear model's stability.

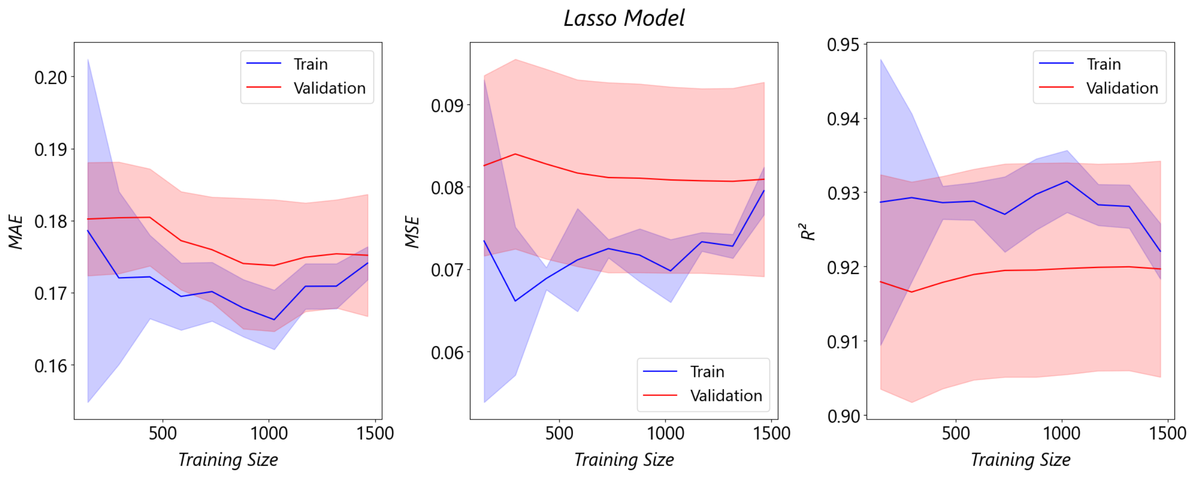


Figure C.2: MAE, MSE, and R-squared train and validation curves for the verification of the Lasso model's stability.

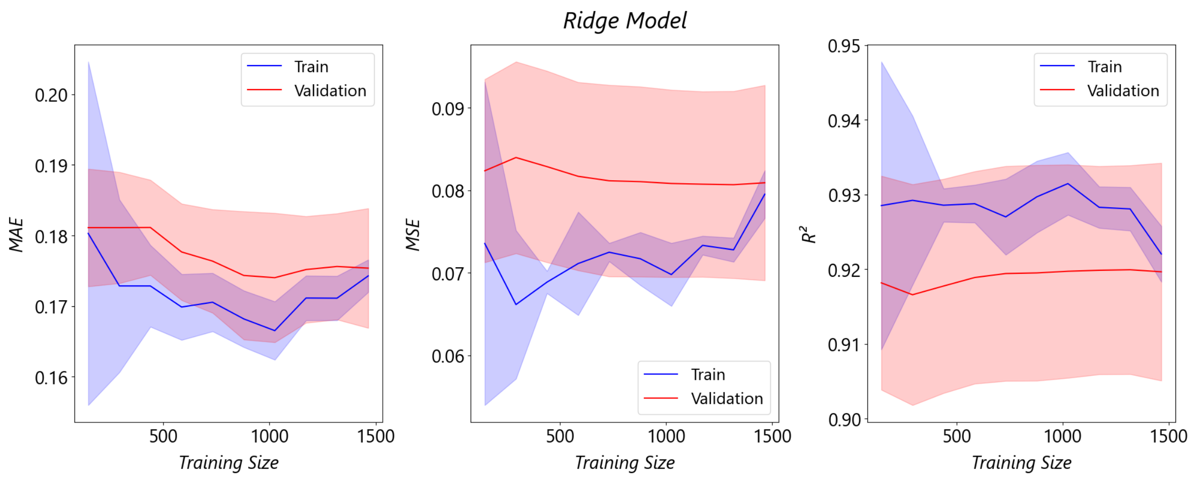


Figure C.3: MAE, MSE, and R-squared train and validation curves for the verification of the Ridge model's stability.

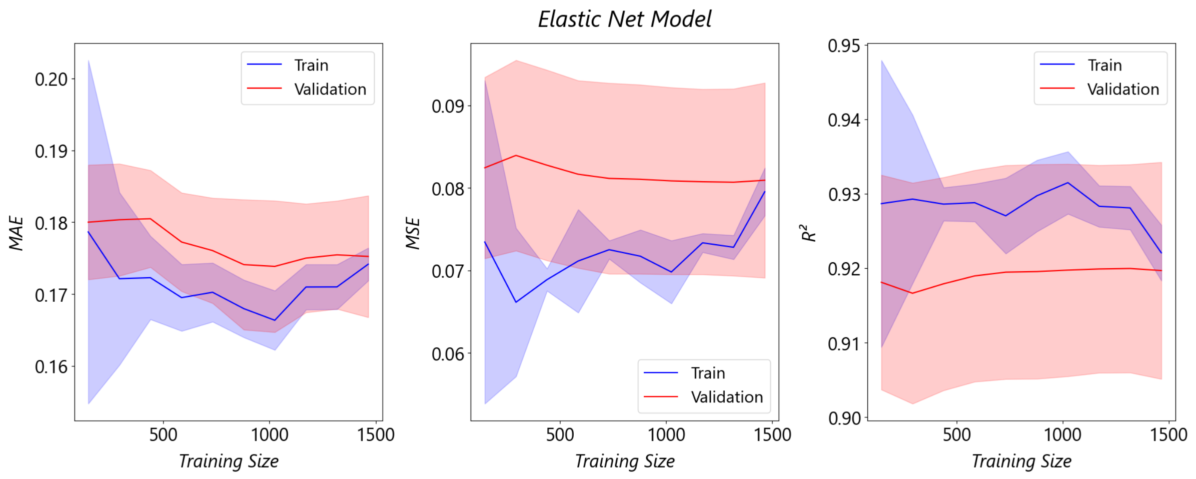


Figure C.4: MAE, MSE, and R-squared train and validation curves for the verification of the Elastic Net model's stability.

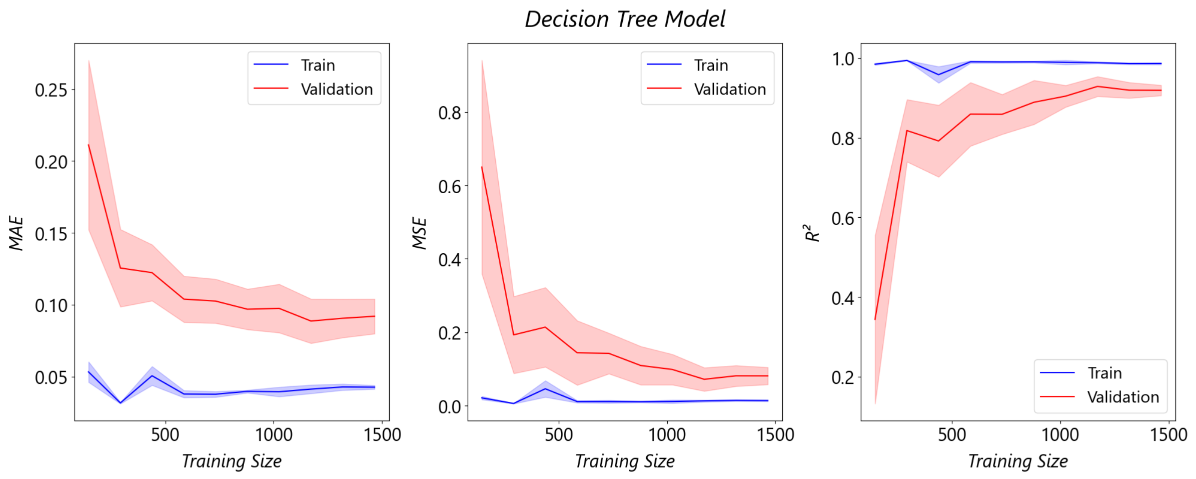


Figure C.5: MAE, MSE, and R-squared train and validation curves for the verification of the Decision Tree model's stability.

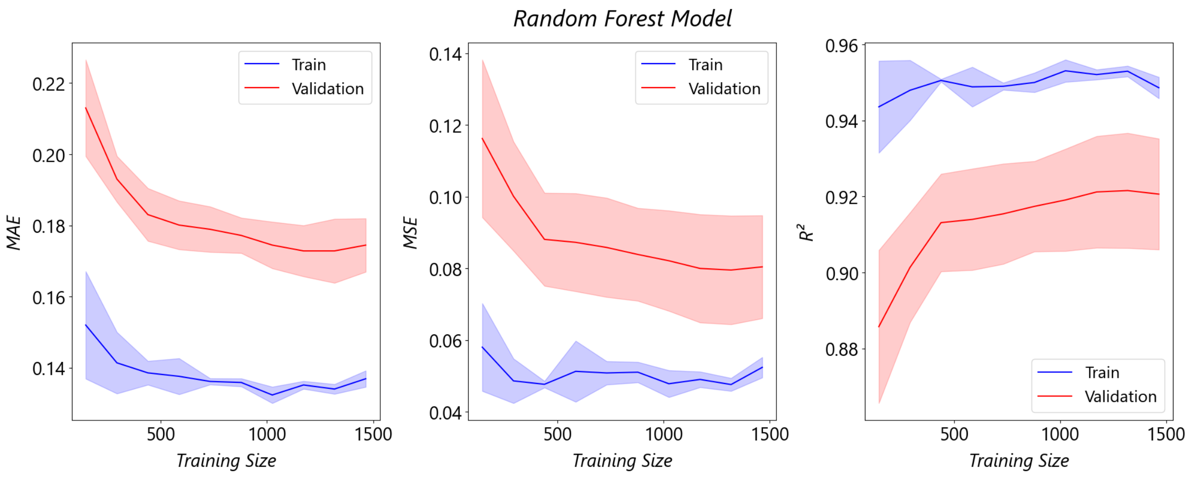


Figure C.6: MAE, MSE, and R-squared train and validation curves for the verification of the Random Forest model's stability.

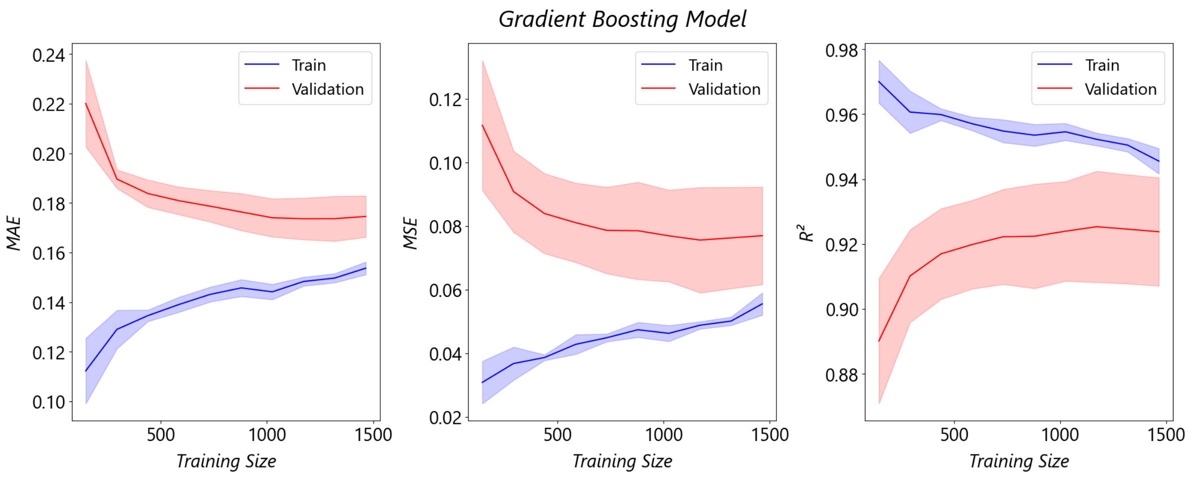


Figure C.7: MAE, MSE, and R-squared train and validation curves for the verification of the Gradient Boosting model's stability.

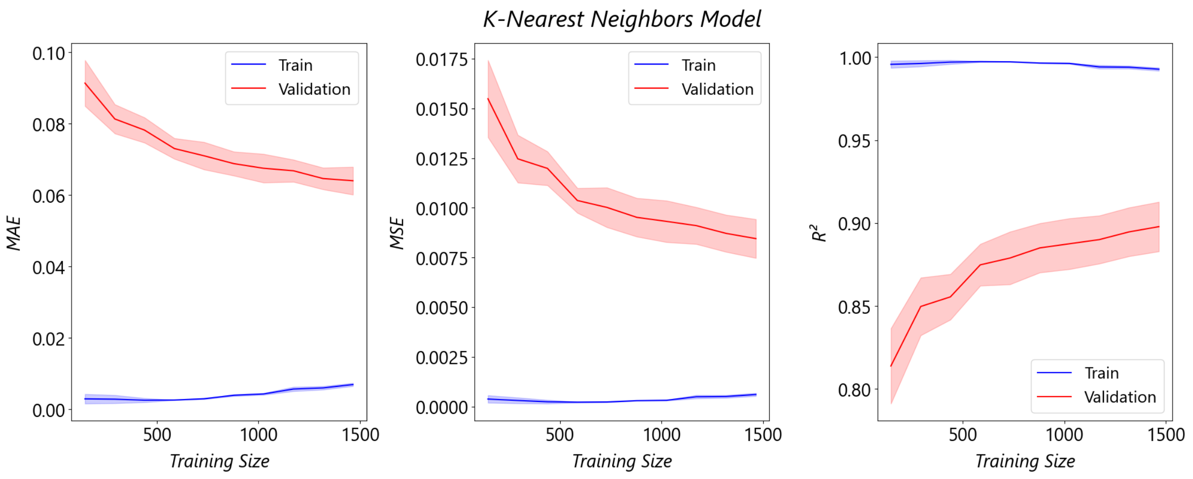


Figure C.8: MAE, MSE, and R-squared train and validation curves for the verification of the K-Nearest Neighbors model's stability.

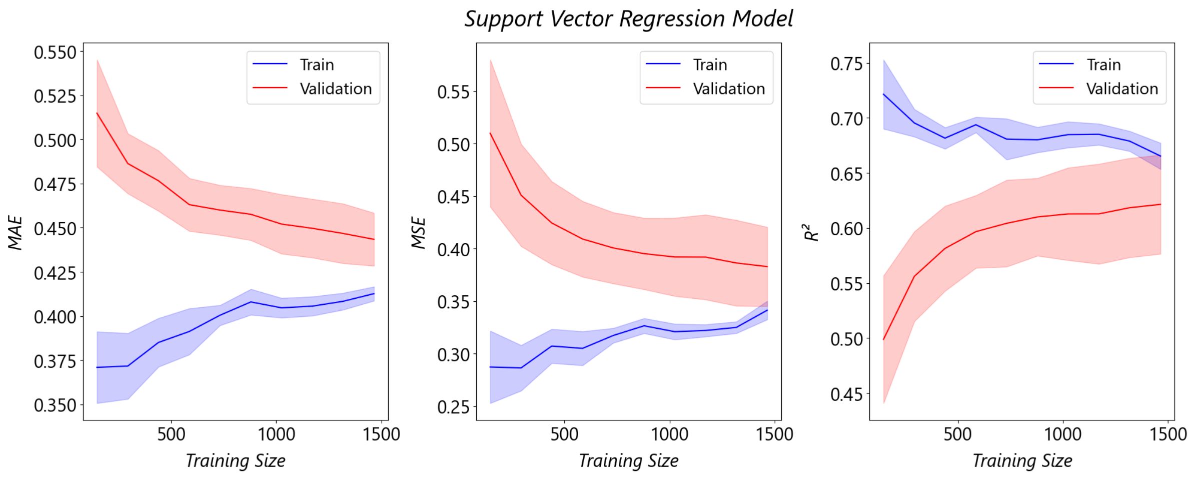


Figure C.9: MAE, MSE, and R-squared train and validation curves for the verification of the Support Vector Regression model's stability.

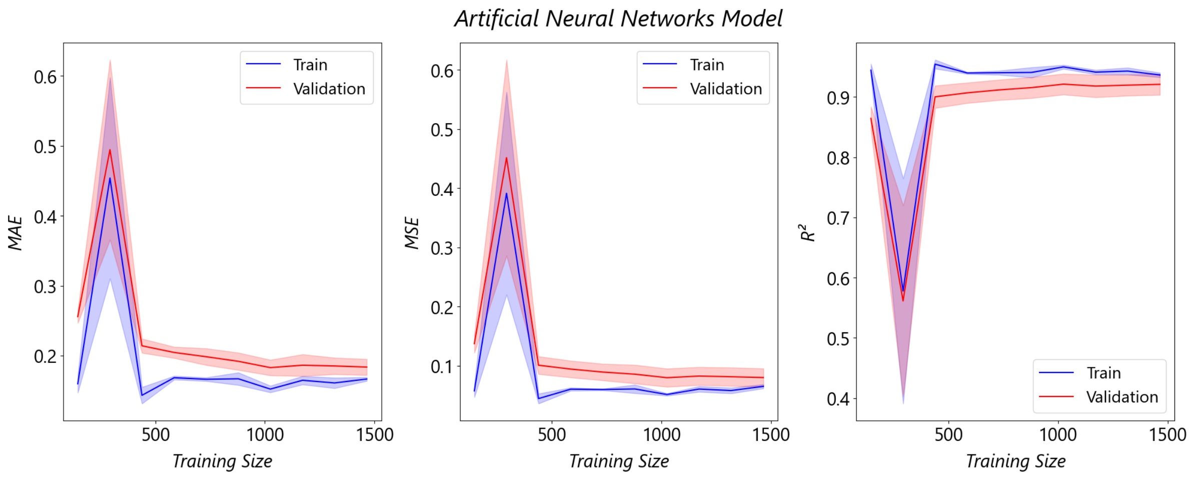


Figure C.10: MAE, MSE, and R-squared train and validation curves for the verification of the Artificial Neural Network model's stability.

Bibliography

[1] Avbuyer. URL https://www.avbuyer.com/aircraft/turboprops/beechcraft/1900d/370223.

[2] Airways. The history of commercial flight: How global travel took off, 2023. URL https://www.airwaysmag.com/legacy-posts/how-global-travel-took-off#:~:text=A%20New%20Era%20of%20Aviation,specifically%20built%20to%20carry%20passengers.

[3] AOPA. Aircraft maintenance: Major repairs and overhauls for propellers, 2018. URL https://www.aopa.org/news-and-media/all-news/2018/january/09/propeller-maintenance-major-repairs-and-overhauls#:~:text=Almost%20all%20propellers%20have%20time,to%20seven%20years%20in%20service.

[4] Dario Radecic Appsilon. Top 10 machine learning evaluation metrics for regression - implemented in r, 2023. URL https://www.appsilon.com/post/machine-learning-evaluation-metrics-regression.

[5] ASUS. Vivobook pro 15 oled (k6502), 2025. URL https://www.asus.com/es/laptops/for-creators/vivobook/asus-vivobook-pro-15-oled-k6502/techspec/.

[6] Latitude 33º Aviation. What is aircraft depreciation?, 2024. URL https://l33jets.com/resources/blog/what-is-aircraft-depreciation/#:~:text=The%20depreciation%20rates%20for%20aircraft,vary%20depending%20on%20several%20factors.

[7] AWS. What is overfitting?, 2025. URL https://aws.amazon.com/what-is/overfitting/.

[8] Academia Balderix. Percentiles (estadística), 2022. URL https://es.statisticseasily.com/glosario/%C2%BFQu%C3%A9-es-el-rango-intercuartil-IQR%3F/.

[9] Abhishek Barai. Normal distribution and machine learning, 2020. URL https://medium.com/analytics-vidhya/normal-distribution-and-machine-learning-ec9d3ca05070.

[10] Codificando Bits. ¿cómo hacer el análisis exploratorio de datos? – guía paso a paso, 2024. URL https://codificandobits.com/blog/analisis-exploratorio-de-datos/.

[11] Codificando Bits. La función de activación, 2025. URL https://codificandobits.com/blog/funcion-de-activacion/.

[12] Aviation Benefits Beyond Borders. Economic growth, 2023. URL https://aviationbenefits.org/economic-growth/.

[13] DataScientest. Cross-validation : definición e importancia en machine learning, 2024. URL https://datascientest.com/es/cross-validation-definicion-e-importancia.

[14] Instituto Nacional de Estadística (INE). Encuesta de estructura salarial (ees), 2024. URL https://www.ine.es/dyngs/Prensa/EES2022.htm.

[15] Blog de Zendesk. What is artificial intelligence (ai)? a complete guide, 2024. URL https://www.zendesk.es/blog/what-is-artificial-intelligence/.

[16] David Seme Dev. Exploratory data analysis using data visualization techniques, 2023. URL https://dev.to/daveseme/exploratory-data-analysis-using-data-visualization-techniques-4n48.

[17] Ifihanagbara Olusheye Dev. Problem definition (1/6), 2021. URL https://dev.to/ifihan/problem-definition-1-6-207p.

[18] Learn Statistics Easily. ¿qué es el riq (rango intercuartil)?, 2022. URL https://www.probabilidadyestadistica.net/percentiles/.

[19] ICAO Environment. Carbon offsetting and reduction scheme for international aviation (corsia), 2024. URL https://www.icao.int/environmental-protection/CORSIA/Pages/default.aspx.

[20] Eurocontrol. *European Aviation in 2040. Challenges of Growth.* 2018.

[21] Eurocontrol. *Eurocontrol Aviation Outlook 2050. Main Report.* 2022.

[22] Foqum. Knn, 2024. URL https://foqum.io/blog/termino/knn/.

[23] Platform for Accelerating the Circular Economy (PACE). A new circular vision for electronics time for a global reboot, 2019. URL https://www3.weforum.org/docs/WEF_A_New_Circular_Vision_for_Electronics.pdf.

[24] Geeks for Geeks. Feature transformation techniques in machine learning, 2022. URL https://www.geeksforgeeks.org/feature-transformation-techniques-in-machine-learning/.

[25] Geeks for Geeks. Comparing randomized search and grid search for hyperparameter estimation in scikit learn, 2022. URL https://www.geeksforgeeks.org/comparing-randomized-search-and-grid-search-for-hyperparameter-estimation-in-scikit-learn/.

[26] Geeks for Geeks. Gradient boosting in ml, 2023. URL hhttps://www.geeksforgeeks.org/ml-gradient-boosting/.

[27] Geeks for Geeks. Regresión de vectores de soporte (svr) utilizando núcleos lineales y no lineales en scikit learn, 2023. URL https://www.geeksforgeeks.org/support-vector-regression-svr-using-linear-and-non-linear-kernels-in-scikit-learn/.

[28] Geeks for Geeks. Tipos de ia según sus funcionalidades, 2024. URL https://www.geeksforgeeks.org/types-of-ai-based-on-functionalities/#selfaware-ai-the-future-of-artificial-intelligence.

[29] Geeks for Geeks. Univariate, bivariate and multivariate data and its analysis, 2024. URL https://www.geeksforgeeks.org/univariate-bivariate-and-multivariate-data-and-its-analysis/.

[30] Geeks for Geeks. What is data acquisition in machine learning?, 2024. URL https://www.geeksforgeeks.org/what-is-data-acquisition-in-machine-learning/.

[31] Geeks for Geeks. Métricas de regresión, 2024. URL https://www.geeksforgeeks.org/regression-metrics/.

[32] Geeks for Geeks. Problemas de gradientes evanescentes y explosivos en el aprendizaje profundo, 2024. URL https://www.geeksforgeeks.org/vanishing-and-exploding-gradients-problems-in-deep-learning/.

[33] Geeks for Geeks. Pasos para construir un modelo de aprendizaje automático, 2024. URL https://www.geeksforgeeks.org/steps-to-build-a-machine-learning-model/.

[34] Geeks for Geeks. Las mejores bibliotecas de python para el aprendizaje automático, 2024. URL https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/.

[35] Geeks for Geeks. Cómo trabajar con datos faltantes en pandas, 2024. URL https://www.geeksforgeeks.org/working-with-missing-data-in-pandas/.

[36] Geeks for Geeks. ¿what is variance?, 2024. URL https://www.geeksforgeeks.org/variance/.

[37] Geeks for Geeks. Cross validation in machine learning, 2025. URL https://www.geeksforgeeks.org/cross-validation-machine-learning/.

[38] Geeks for Geeks. Feature engineering Scaling, normalization, and standardization, 2025. URL https://www.geeksforgeeks.org/ml-feature-scaling-part-2/.

[39] Geeks for Geeks. Python | regresión de árbol de decisión con sklearn, 2025. URL https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/.

[40] Geeks for Geeks. Ensemble learning, 2025. URL https://www.geeksforgeeks.org/a-comprehensive-guide-to-ensemble-learning/.

[41] Geeks for Geeks. Feature selection techniques in machine learning, 2025. URL https://www.geeksforgeeks.org/feature-selection-techniques-in-machine-learning/.

[42] Geeks for Geeks. Python para el aprendizaje automático, 2025. URL https://www.geeksforgeeks.org/python-for-machine-learning/.

[43] Geeks for Geeks. Introducción al aprendizaje automático: qué es y sus aplicaciones, 2025. URL https://www.geeksforgeeks.org/introduction-machine-learning/.

[44] Geeks for Geeks. Detect and remove the outliers using python, 2025. URL https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python/.

[45] Geeks for Geeks. Ml | preprocesamiento de datos en python, 2025. URL https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/.

[46] Geeks for Geeks. Función de activación de relu en el aprendizaje profundo, 2025. URL https://www.geeksforgeeks.org/relu-activation-function-in-deep-learning/.

[47] Geeks for Geeks. Linear regression in machine learning, 2025. URL https://www.geeksforgeeks.org/ml-linear-regression/.

[48] Geeks for Geeks. Lasso vs ridge vs elastic net | ml, 2025. URL https://www.geeksforgeeks.org/lasso-vs-ridge-vs-elastic-net-ml/.

[49] Greek for Greeks. Encoding categorical data in sklearn, 2024. URL https://www.geeksforgeeks.org/encoding-categorical-data-in-sklearn/#3-ordinal-encoding.

[50] Greek for Greeks. Codificación one hot en el aprendizaje automático, 2025. URL https://www.geeksforgeeks.org/ml-one-hot-encoding/.

[51] Tableau from Salesforce. What is the history of artificial intelligence (ai)?, 2024. URL https://www.tableau.com/data-insights/ai/history.

[52] Chirag Goyal. Feature transformations in data science: A detailed walkthrough, 2024. URL https://www.analyticsvidhya.com/blog/2021/05/feature-transformations-in-data-science-a-detailed-walkthrough/.

[53] Graph. Machine learning en python, 2025. URL hhttps://www.grapheverywhere.com/machine-learning-en-python/.

[54] Sky Aviation Holding. What is aircraft tbo and why is it important?, 2022. URL https://skyaviationholdings.com/what-is-aircraft-tbo-and-why-is-it-important/.

[55] IATA. *Global Outlook for Air Transport. Deep Change.* 2024.

[56] IATA and KPMG. *Aircraft acquisition cost and depreciation.* 2015.

[57] IBM. What is clustering?, 2021. URL https://www.ibm.com/think/topics/clustering.

[58] IBM. Understanding the different types of artificial intelligence, 2023. URL https://www.ibm.com/think/topics/artificial-intelligence-types.

[59] IBM. What is the k-nearest neighbors (knn) algorithm?, 2024. URL https://www.ibm.com/think/topics/knn.

[60] IBM. What is machine learning?, 2024. URL https://www.ibm.com/think/topics/machine-learning.

[61] IBM. What is random forest?, 2025. URL https://www.ibm.com/think/topics/random-forest.

[62] Jacob Murel Ph.D. (IBM). What is reinforcement learning?), 2024. URL https://www.ibm.com/think/topics/reinforcement-learning.

[63] Niklas Donges Built In. 4 desventajas de las redes neuronales, 2023. URL https://builtin.com/data-science/disadvantages-neural-networks#:~:text=Neural%20Networks%20Are%20Computationally%20Expensive,to%20train%20completely%20from%20scratch.

[64] Index. Understanding data types in python programming, 2024. URL https://www.index.dev/blog/understanding-python-data-types.

[65] Anders Nordgren Emerald Insight. Artificial intelligence and climate change: ethical issues, 2023. URL https://www.emerald.com/insight/content/doi/10.1108/jices-11-2021-0106/full/html.

[66] Alicia Tuovila Investopedia. Depreciation: Definition and types, with calculation examples, 2024. URL hhttps://www.investopedia.com/terms/d/depreciation.asp#:~:text=Depreciation%20is%20an%20accounting%20practice,value%20has%20been%20used%20up.

[67] Eda Kavlakoglu IBM Jacob Murel Ph.D. What is feature engineering?, 2024. URL https://www.ibm.com/think/topics/feature-engineering.

[68] Weiwei Jiang. Applications of deep learning in stock market prediction: Recent progress. *Expert Systems with Applications*, 184:115537, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.115537. URL https://www.sciencedirect.com/science/article/pii/S0957417421009441.

[69] Jobted. Sueldo del ingeniero en españa, 2024. URL https://www.jobted.es/salario/ingeniero?

[70] IBM Julie Rogers. What is data cleaning?, 2024. URL hhttps://www.ibm.com/think/topics/data-cleaning#:~:text=Data%20cleaning%2C%20also%20called%20data%20cleansing%20or%20data,in%20raw%20data%20sets%20to%20improve%20data%20quality.

[71] Paul W. Mielke Kenneth J. Beery, Janis E. Johnston. *A Primer of Permutation Statistical Methods*. Springer Cham. doi: https://doi.org/10.1007/978-3-030-20933-9.

[72] Santi Seguí Laural Igual. *Introduction to Data Science*. Springer, 2024.

[73] Kamil Wrzosek LOT. International aircraft code: Understanding aircraft registration numbers, 2023. URL https://www.lot.com/mt/en/explore/inspirations/aviation-trivia/what-is-aircraft-registration.

[74] Medium. Understanding k-nearest neighbors (knn) regression in machine learning, 2023. URL https://medium.com/@nandiniverma78988/understanding-k-nearest-neighbors-knn-regression-in-machine-learning-c751a7cf516c.

[75] Medium. Simple linear regression in python, 2023. URL https://medium.com/@shuv.sdr/simple-linear-regression-in-python-a0069b325bf8.

[76] Medium. Understanding decision tree regressor: An in-depth intuition, 2024. URL https://farshadabdulazeez.medium.com/understanding-decision-tree-regressor-an-in-depth-intuition-a1d3af182efd.

[77] Anna Braun Medium. How to collect data for a machine learning model, 2021. URL https://medium.com/codex/how-to-collect-data-for-a-machine-learning-model-2b152752a15b.

[78] Danny Butvinik Medium. Feature selection — exhaustive overview, 2021. URL https://medium.com/analytics-vidhya/feature-selection-extended-overview-b58f1d524c1c.

[79] Gozde Madendere Medium. Data visualization for exploratory data analysis (eda), 2023. URL https://medium.com/@gozdemadendere/data-visualization-for-exploratory-data-analysis-eda-ddf850539575.

[80] GVinod Kumar Medium. Data transformations in ml: Different transformations in machine learning: Log transformer, reciprocal transformer, square transformer, square root transformer, box-cox transformer, yeo johnson transformer, right-skewed and left-skewed data., 2024. URL https://medium.com/@vinodkumargr/09-data-transformations-in-ml-different-transformations-in-machine-learning-log-transformer-d794d61d143d.

[81] Jonte Dancker Medium. A brief introduction to feature scaling, 2022. URL https://medium.com/@jodancker/a-brief-introduction-to-feature-scaling-e396356937b8.

[82] Kemal Toprak Ucar Medium. How to calculate the correlation between categorical and continuous values, 2023. URL https://medium.com/@ktoprakucar/how-to-calculate-the-correlation-between-categorical-and-continuous-values-dcb7abf79406.

[83] Krishnakanth Naik Jarapala Medium. Categorical data encoding techniques, 2023. URL https://medium.com/aiskunks/categorical-data-encoding-techniques-d6296697a40f.

[84] Mehmet Ali TOR Medium. Evaluation metrics for regression problems, 2024. URL https://medium.com/@mehmetalitor/evaulation-metrics-for-regression-problems-e07d082d8fb1.

[85] Mohammed Alhamid Medium. What is cross-validation?, 2020. URL https://medium.com/data-science/what-is-cross-validation-60c01f9d9e75.

[86] Nandini Verma Medium. An introduction to support vector regression (svr) in machine learning, 2023. URL https://medium.com/@nandiniverma78988/an-introduction-to-support-vector-regression-svr-in-machine-learning-681d541a829a.

[87] Sho Shimoda (Medium). Unsupervised learning (including clustering, dimensionality reduction), 2023. URL https://medium.com/@syantien/unsupervised-learning-including-clustering-dimensionality-reduction-5669438cace6.

[88] Shrinath Bhat Medium. A comprehensive guide to outlier treatment, 2023. URL https://medium.com/@bhatshrinath41/quick-guide-to-outlier-treatment-ada01f8cfc5.

[89] SnapWise Medium. Techniques for converting categorical data into numerical data, 2023. URL https://medium.com/@rafiemon71/techniques-for-converting-categorical-data-into-numerical-data-f1c9d0a3863f.

[90] Sowhardh Honnappa Medium. Top 100 open source datasets for data science, 2020. URL https://medium.com/analytics-vidhya/top-100-open-source-datasets-for-data-science-cd5a8d67cc3d.

[91] Supriyo Ain (Medium). What is association rule learning?, 2023. URL https://medium.com/@ainsupriyofficial/what-is-association-rule-learning-a6ef399fdc01.

[92] Tristen Wallace Medium. Bivariate data exploration with matplotlib & seaborn, 2024. URL https://medium.com/@tristenwallace/bivariate-data-exploration-with-matplotlib-seaborn-6ff43de5735a#:~:text=Bivariate%20plots%20investigate%20relationships%20between,the%20distributions%20of%20different%20variables.

[93] Novajet. Aircraft depreciation explained, 2024. URL https://www.novajet.com/knowledge-sharing/aircraft-depreciation-explained/.

[94] Idan Novogroder. Data preprocessing in machine learning: Steps & best practices), 2024. URL https://lakefs.io/blog/data-preprocessing-in-machine-learning/.

[95] Nvidia. Xgboost, 2024. URL https://www.nvidia.com/en-us/glossary/xgboost/.

[96] Alokya Kanungo Earth Org. The green dilemma: Can ai fulfil its potential without harming the environment?, 2023. URL https://earth.org/the-green-dilemma-can-ai-fulfil-its-potential-without-harming-the-environment/.

[97] Rozina Khattak Earth Org. International e-waste day 2024: The environmental impact of e-waste, 2024. URL https://earth.org/environmental-impact-of-e-waste/.

[98] Platzi. Importancia de la definión del problema en machine learning, 2024. URL https://platzi.com/clases/1178-scikit/8823-importancia-de-definir-el-problema-en-machine-le-3/.

[99] Javat Point. Distribución normal: Qué es, cómo se calcula y ejemplos, 2024. URL https://www.javatpoint.com/data-visualization-in-machine-learning.

[100] Scribbr Pritha Bhandari. Missing data | types, explanation, & imputation, 2021. URL https://www.scribbr.com/statistics/missing-data/.

[101] Tanoy (Quora). Step-by-step guide for implementing machine learning algorithms), 2024. URL https://www.quora.com/Can-anyone-provide-a-step-by-step-guide-for-implementing-machine-learning-algorithms-in-a-real-world-application-emphasizing-best-practices-and-potential-challenges.

[102] Medium Rajat Sharma. Practical examples of data cleaning using pandas and numpy, 2024. URL https://medium.com/pythoneers/practical-examples-of-data-cleaning-using-pandas-and-numpy-5f59021f0144.

[103] Sebastian Raschka. *Python Machine Learning*. Packt Publishing Ltd., 2015.

[104] Karen Hao MIT Technology Review. Training a single ai model can emit as much carbon as five cars in their lifetimes, 2019. URL https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/.

[105] Paula Rodo. Data visualization in machine learning, 2024. URL https://economipedia.com/definiciones/distribucion-normal.html.

[106] Shaun Turney Scribbr. Chi-square (x-square) tests | types, formula & examples, 2023. URL https://www.scribbr.com/statistics/chi-square-tests/.

[107] Skymart. Glossary: Msn - manufacturer's serial number, 2014. URL https://skymartsales.com/glossary/msn.php#:~:text=Serial%20numbers%20can%20trace%20which,745%20first%20flight%20Jan%202004.

[108] SourceOne. Exploring aircraft lifespans and retirement decisions, 2024. URL https://blog.sourceonespares.com/exploring-aircraft-lifespans-and-retirement-decisions#:~:text=Regional%20Jets%20and%20Turboprops&text=Regional%20jets%20may%20have%20a,30%20years%20with%20proper%20maintenance.

[109] Orlando Spencer. Why are margins so low in the airline industry?, 2023. URL https://www.orlandospencer.com/blog/41/why-are-margins-so-low-in-the-airline-industry.

[110] Medium Subha. Handling missing values in dataset — 9 methods that you need to know, 2024. URL https://medium.com/@pingsubhak/handling-missing-values-in-dataset-7-methods-that-you-need-to-know-5067d4e32b62.

[111] Stratos Joel Thomas. What is the lifespan of a private jet, 2024. URL https://www.stratosjets.com/blog/private-jet-lifespan/#:~:text='%20Just%20like%20vehicles%2C%20your%20private,25%2C000%20hours%20in%20the%20sky.

[112] Byron Vargas. Limpieza de datos en python: métodos esenciales y mejores prácticas, 2024. URL https://www.byronvargas.com/web/como-se-limpia-en-python/#:~:text=Veamos%20a%20continuaci%C3%B3n%20algunos%20m%C3%A9todos%20clave%20para%20llevar,identificar%20y%20eliminar%20registros%20duplicados.%20...%20M%C3%A1s%20elementos.

[113] Alakh Sethi Analytics Vidhya. Support vector regression tutorial for machine learning, 2024. URL https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/.

[114] Analytics Vidhya. How to handle missing data in python? [explained in 5 easy steps], 2024. URL https://www.analyticsvidhya.com/blog/2021/05/dealing-with-missing-values-in-python-a-complete-guide/.

[115] Aniruddha Bhandari Analytics Vidhya. Feature scaling: Engineering, normalization, and standardization (updated 2025), 2025. URL https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/.

[116] Chirag Goyal Analytics Vidhya. Guide on outlier detection methods, 2025. URL https://www.analyticsvidhya.com/blog/2021/05/feature-engineering-how-to-detect-and-remove-outliers-with-python-code/.

[117] Harika Analytics Vidhya. Detecting and treating outliers | treating the odd one out!, 2025. URL https://www.analyticsvidhya.com/blog/2021/05/detecting-and-treating-outliers-treating-the-odd-one-out/.

[118] Suvarna Analytics Vidhya. Difference between skewness and kurtosis, 2025. URL https://www.analyticsvidhya.com/blog/2021/05/shape-of-data-skewness-and-kurtosis/.

[119] Aman Analytics Vihya. Feature selection in machine learning, 2025. URL https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/.

[120] Scientific Computing World. The true cost of ai innovation, 2024. URL https://www.scientific-computing.com/analysis-opinion/true-cost-ai-innovation.