NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DIOGO ABRANTES DE SOUSA

BSc Electrical and Computer Engineering

USING MACHINE LEARNING AND DEEP LEARNING TECHNIQUES TO PREDICT HOTEL BOOKING CANCELLATIONS

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
March, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

USING MACHINE LEARNING AND DEEP LEARNING TECHNIQUES TO PREDICT HOTEL BOOKING CANCELLATIONS

DIOGO ABRANTES DE SOUSA

BSc Electrical and Computer Engineering

Adviser: João Almeida das Rosas
Assistant Professor, NOVA University Lisbon

Co-adviser: João Paulo Ferreira Dias
CTO, Host Hotel Systems

Examination Committee

Chairs: Rui Manuel Leitão Tavares
Assistant Professor, NOVA University Lisbon

Ana Inês da Silva Oliveira
Assistant Professor, NOVA University Lisbon

Adviser: João Almeida das Rosas
Assistant Professor, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
March, 2024

Using machine learning and deep learning techniques to predict hotel booking cancellations

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.0.3) [**novathesis-manual**].

To my grandfather, António.

Acknowledgements

I would like to take this opportunity to express my sincere thanks to all those who who contributed to the completion of this project.

First of all, to João Dias, CTO of Host Hotel Systems, not only for the opportunity to carry out this dissertation in conjunction with HHS, but also for all the help and advice he has given me over the last year. Next, I would like to thank my supervisor João Rosas, for accepting this challenge and for the guidance he has given me throughout this project.

Next, I would like to thank the most important people in this whole process, my parents Manuel and Angela, and my sister Carolina. I will be eternally grateful for all the love, support and trust they have placed in me, especially over the last five years.

To Tatiana, my girlfriend, for all the love, patience and for never stopping believing in me.

To my grandparents, Francisco, Francisca and Diamantina, whom I miss very much.

To the friends that college has given me over the last five years, Caetano, Vaqueira, Tomás, André, Filipa, Eduardo and Hugo, thank you for sharing this journey with me. Every sleepless night will be rewarded, never doubt it.

I would like to thank all my coworkers at Host, especially the development team, for accompanying me over the last year.

Finally, I'd like to say a special thank you to my grandfather António. Thank you for all the love, all the stories, all the conversations and all the teachings. I hope you never stop following me, wherever you are.

”

“Big data is no use if you can’t turn it into knowledge.”

— Pedro Domingos, Master Algorithm
(Professor, researcher and engineer)

Abstract

Booking cancellations are one of the most concerning topics in the hotel industry. From the revenue lost from canceled bookings, the uncertainty created in occupancy rates or even the inaccuracy of the hotel's resource management, the negative impacts of booking cancellations are numerous, making it a sensitive subject for those working in the field. To combat these obstacles, hotels have been forced to adopt strategies such as creating cancellation policies or managing the sale of their rooms with overbooking. Although these strategies are effective in controlling the negative impacts caused by booking cancellations, they can also have a negative impact on a hotel's reputation and customer satisfaction, especially when it comes to overbooking.

This project aims to present a solution to help identify possible booking cancellations by creating predictive models using machine learning and deep learning methodologies. Based on the CRISP-DM paradigm and a real and current dataset of a Portuguese hotel chain, this project addresses various techniques used in the creation of binary classification models, from data processing to modeling and model evaluation.

Given the constant superiority of machine learning algorithms in models based on tabular data, making deep learning solutions still not very viable in this field, this project also seeks to test some deep learning architectures that have been developed in recent years.

The machine learning model created using the XGBoost algorithm showed promising results, with an AUC of 0.77 and a precision and recall of 0.93 and 0.54, respectively. Although the results are not yet excellent, they show that the use of this solution is viable and has room for further improvements.

Keywords: Machine learning, Deep learning, Predictive analytics, Booking cancellations

Resumo

O cancelamento de reservas é um dos temas que mais preocupações cria na indústria hoteleira. Desde a receita que é perdida pelas reservas canceladas, a imprevisibilidade criada na taxa de ocupação ou até a imprecisão na gestão de recursos dos próprios hotéis, os impactos negativos que advém do cancelamento de reservas são muitos, tornando-o um tema bastante sensível para quem trabalha na área. Para combater estas dificuldades, os hotéis viram-se obrigados a adotar estratégias como a criação de políticas de cancelamento ou a gestão da venda dos seus quartos em overbooking. Apesar de serem efetivas no controlo dos impactos negativos causados pelo cancelamento de reservas, as consequências da utilização destas estratégias também podem impactar negativamente a reputação dos hotéis e a satisfação do cliente, principalmente no tema do overbooking.

O presente projeto visa apresentar uma solução para ajudar na identificação de possíveis cancelamentos de reservas, com a criação de modelos preditivos assentes nas metodologias de machine learning e deep learning. Usando como base o paradigma CRISP-DM e um dataset real e atual de uma cadeia de hotéis portuguesa, este projeto aborda várias técnicas utilizadas na criação de modelos de classificação binária, desde a parte de processamento dos dados até à modelação e avaliação dos modelos.

Aliado à constante superioridade dos algoritmos de machine learning em modelos assentes em dados tabulares, tornando as soluções de deep learning ainda pouco viáveis neste campo, este projecto procura também testar algumas arquiteturas de deep learning que têm sido desenvolvidas nos últimos anos.

O modelo de machine learning criado com a utilização do algoritmo XGBoost apresentou resultados promissores, com um AUC de 0.77 e uma precisão e um recall de 0.93 e 0.54, respetivamente. Apesar dos resultados ainda não serem excelentes, mostram que a utilização desta solução é viável e tem espaço para continuar a ser explorada.

Palavras-chave: Aprendizagem automática, Aprendizagem profunda, Análise preditiva, Cancelamento de reservas

Contents

List of Figures

List of Tables

Acronyms

AdaBoost Adaptive Boosting *(p. 42)*
AI Artificial Intelligence *(pp. 5, 6, 10)*
API Application Programming Interface *(p. 42)*
AUC Area Under Curve *(pp. 18, 21)*

CRISP-DM Cross Industry Process for Data Mining *(pp. 4, 25, 55, 57)*

DL Deep Learning *(pp. 2, 22, 42, 45–49, 55–57)*
DNN Deep Neural Network *(p. 13)*

EDA Exploratory Data Analysis *(p. 31)*

HHS Host Hotel Systems *(p. 2)*

IQR Interquartile Range *(pp. 29, 30)*
IT Information Technology *(p. 2)*

KNN K-Nearest Neighbors *(pp. 27, 41)*

LightGBM Light Gradient Boosting Machine *(p. 42)*

ML Machine Learning *(pp. 2, 6, 42, 45, 46, 49, 55–57)*

ReLU Rectified Linear Unit *(p. 11)*
ROC Receiver Operating Characteristics *(p. 18)*

XGBoost eXtreme Gradient Boosting *(pp. 39, 42)*

1

Introduction

1.1 Context and Motivation

The way tourists plan and book their accommodations has changed significantly with the emergence of online travel agencies (OTAs) and booking platforms. Customers have now more convenience and flexibility thanks to the move to these platforms, which allow them to quickly compare costs, read reviews, and make appointments at any time and from any location. However, this flexibility combined with the cancellation policies created by these platforms that benefit the tourist has led to an increase in hotel booking cancellations. The unpredictable nature of booking cancellations presents major problems for hoteliers when it comes to allocating resources and revenue management. In addition to costing money from rooms sold, cancelled reservations interfere with forecasting and inventory control procedures. In an attempt to control the negative impacts caused by the increase in cancellations of their bookings, hotels have been forced to develop other types of strategies such as developing their own cancellation policies, creating dynamic pricing strategies or using overbooking sales. Although the use of these types of strategies has proved to be effective for hotel management, their strictness, especially when it comes to overbooking or cancellation policies, may not be so appealing to tourists, leading to cases of complaints that can affect the hotel's social reputation and image [1]. As a result, hotels began to feel the need to explore other solutions in order to control the flow of canceled bookings, preferably ones that didn't have such an impact on customer satisfaction.

Although some canceled bookings cannot be predicted, bookings that happen due to unpredictable circumstances or last-minute plans changes, it is believed that some of them may have a predictive component, derived from booking patterns or seasonal trends, for example. Combining this belief with recent advances in the field of artificial intelligence, especially in the use of machine learning and deep learning algorithms to create predictive models that work with tabular data, studies of solutions based on these methodologies have begun to surface. As the hospitality industry being an area where the collection of various types of data is very accessible and fast, the creation of this type of solution is gaining even more momentum and systems that use it can create a significant advantage

over the competition.

Despite the development of predictive models using ML and DL methodologies having very positive results in various areas, their use for predicting cancellations is still a little explored area, although some studies show that it could be a viable solution with potential. If the use of these predictive models proves to be effective in predicting cancellations, hotels that use them can outperform the competition with more optimized revenue management and operations processes. In addition, and to provide a better experience, hotels can streamline operations, optimize staff scheduling, and spend resources more wisely.

1.2 Objectives

This dissertation will be developed in partnership with **HHS - Host Hotel Systems**. HHS is an IT consulting and software development company, specialized in the hospitality industry. Host Hotel Systems' solutions cover different operational areas including reservation management, revenue management, inventory control, process automation, guest management, data analytics and more.

As HHS works on several aspects of hotel management, it was necessary to take advantage of all the data provided from its systems, especially in the development of AI powered products. With that, and focusing on predictive analysis, the main objective of this dissertation is to build a predictive model to forecast possible booking cancellations.

In addition, this dissertation also aims to showcase and test some deep learning architectures that have been developed in recently and which still have a relatively questionable use, unlike machine learning models After comparing and analyzing all the models built during this dissertation, a final model will be chosen and analyzed in greater detail, to try to understand its behavior and gain some insights into booking cancellations.

1.3 Document Organization

This document is organized in five chapters: Introduction, State of Art, Methodology, Results and Conclusion.

The Introduction chapter serves as the starting point of the document and sets the context for the research and establishes the motivation behind the chosen topic. The State of Art chapter delves into the existing literature and research relevant to the chosen research topic. It aims to provide a comprehensive review and analysis of the current state of knowledge in the field. This chapter examines the key theories, concepts, methodologies, and findings related to the research problem, highlighting any gaps or limitations in the existing literature. The Methodology chapter provides an explanation of the models creation, discussing the model's pipeline, step by step. The following chapter, Results and Discussion reports and discusses the results of the models and analyzes in more depth the best one. Finally, in the last chapter, the main conclusions are presented along with the exposure of this work's limitations and future work proposals.

2

State of Art

2.1 Impact of Booking Cancellations in Revenue Management

In the constantly evolving and highly competitive hospitality industry, optimizing profits and maintaining business growth depend heavily on efficient revenue management. The optimization of room sales and price decisions to take advantage of shifts in market trends and demand is a key component of revenue management techniques. But the frequency of cancellations is a big problem for hotel revenue management [1], requiring a sophisticated knowledge of how they affect operational procedures and methods for revenue optimization.

Numerous factors, such as pricing tactics, market dynamics, customer behavior, and external considerations like travel restrictions and economic situations, can all have an impact on booking cancellations. Cancellations are largely caused by consumer-driven issues, such as price sensitivity, uncertainty, and changing travel plans, since customers look for value and flexibility when making reservations. Furthermore, a culture of price comparison and flexible booking has been made possible by the growth of third-party booking platforms and online travel agents (OTAs), which has increased cancellation rates [2]. Furthermore, customers may be encouraged to make several appointments or take advantage of last-minute discounts by dynamic pricing schemes and special offers, which could result in a greater cancellation rate.

Booking cancellations have a complex effect on revenue optimization techniques, affecting not just price, inventory control, and customer relations indirectly, but also directly through revenue loss. Cancellations that leave rooms unoccupied and unable to be resold at the same price result in direct revenue loss through missed possibilities to make money [3]. In addition, cancellations cause confusion with inventory management procedures, requiring in-the-moment price and availability modifications to minimize revenue loss.

2.2 CRISP-DM: The pipeline for predictive analysis

CRISP-DM, which stands for Cross Industry Process for Data Mining, is a well-known methodology, published in 1999 by Chapman [4], that serves as a base for the most data science and machine learning projects, as can be seen in figure 2.2. This model is more focused on the data science workflow and not so much on team collaboration and challenges, such as Scrum or Kanban, which added to it is completeness [1] and usefulness for planning and documentation [5] makes it the most popular methodology for data science and machine learning projects.

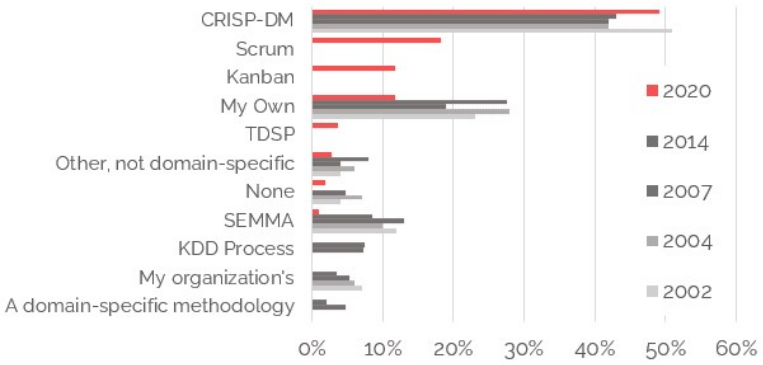

Figure 2.1: Most used methodologies for data driven projects survey. Adapted from [6].

The model is structured in six main phases and besides it sequentially, CRISP-DM workflow is not rigid, where moving back and forward through phases is often required, as represented in figure 2.2.

- **Business Understanding:** The initial phase is dedicated to understanding the context and requirements of the project and setting the objectives. it is important to think the problem in a business perspective and being able to turn it into an analytical problem.

- **Data Understanding:** The data understanding phase is responsible for the collection of the data and performs all exploratory analysis needed to get a complete knowledge of it. This phase is essential to understand if the data available meets the requirements or if other sources need to get explored to get more and different information.

- **Data Preparation:** Data preparation is the crucial step in every data science project since the results of the predictive model are always directly related to the quality of the dataset that is exposed to the algorithms. In this phase, different pre-processing methods are used in the raw data to create the final dataset.

- **Modelling:** Modelling is considered the most compelling phase of the entire pipeline since all the data related tasks are done and the final dataset is exposed to the algorithms. In this step, the algorithms that better fit the problem are chosen,

considering requirements of the data, and the different models are trained and configured.

- **Evaluation:** The models created in the modelling phase are evaluated and the best one is chosen. Since many models can lead to misleading results, it is important to have a critical approach and evaluate using different metrics to be sure of the real performance and robustness of them.

- **Deployment:** The final model is deployed and exposed to real data in a production environment.

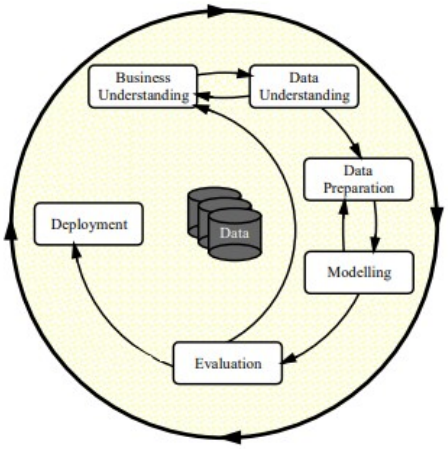

Figure 2.2: CRISP-DM workflow. Adapted from [5].

2.3 Artificial Intelligence Methodologies

Artificial Intelligence is a very fast developing field that aims to replicate in computers the human ability to think, reasoning and problem-solving processes, which has the potential to improve numerous aspects of our society, from automating simple and repetitive tasks to solving more complex scientific problems.

Machine Learning and Deep Learning may be responsible for the construction of powerful and complex AI's systems that have led to significant advances in several areas, including text recognition, computer vision, natural language processing, medicine, finance and others.

The choice between machine learning algorithms and deep learning algorithms depends on the nature of the problem, the availability of data, the complexity of the data, and the desired performance and interpretability trade-offs [7]. Machine learning algorithms can be more suitable for simpler tasks and situations with limited data, while deep learning algorithms excel in handling complex data and benefit from large amounts of labeled data.

2.3.1 Machine Learning

Machine Learning is an AI's sub field that allows computers to self-learn from past experiences (past data) without being explicitly programmed. This approach has become one of the most dominant subfield in AI due to its ability to handle complex and extensive data and identify hidden patterns, making it a powerful tool in data-driven decision making. Mathematical algorithms and models are the core of ML and input data is fed to these models, which analyze it, extract relevant features and use that information to make predictions.

ML has several paradigms that can be chosen and used depending on the problem to be solved, such as Supervised Learning, Unsupervised Learning and Reinforcement Learning.

2.3.1.1 Supervised Learning

Supervised learning is the most used paradigm in Machine Learning. The models that use this paradigm are trained over a set of labeled data, learning its patterns [8]. The goal is to build a model capable of labeling new data or predict a value, based on the trained set.

In Supervised learning two different type of problems can be handled: regression and classification problems. In both, the model can be fed with the same type of data, what differs is the output [9]. While in regression the model is responsible for predicting a continuous value (prediction of a salary or a house price), in classification the model predicts a label (prediction a flower's specie, for example).

2.3.1.2 Unsupervised Learning

Unsupervised learning is a prominent field within machine learning focused on extracting patterns, structures, and relationships from unlabeled data.

The goal of unsupervised learning is to identify inherent structures and relationships within structures and relationships within the data without any predefined target variable or specific guidance. By employing various statistical and computational techniques, unsupervised learning algorithms can uncover underlying patterns, cluster similar data points, detect anomalies, and reduce the dimensionality of complex datasets [10]. This form of learning has wide-ranging applications in diverse fields, including data mining, pattern recognition, natural language processing, and computer vision.

One of the fundamental techniques in unsupervised learning is clustering, where the algorithm groups similar data points together based on their intrinsic properties. By organizing the data into clusters, unsupervised learning algorithms can provide insights into the underlying structure and similarities among the data instances [11]. Clustering algorithms employ different distance measures and optimization methods to ensure that data points within a cluster are more similar to each other than to those in other clusters.

Another crucial aspect of unsupervised learning is dimensionality reduction. In many real-world scenarios, data sets consist of numerous features or variables, which can make analysis and interpretation challenging. Dimensionality reduction techniques aim to transform high-dimensional data into a lower-dimensional representation while preserving the essential information [12]. These techniques help eliminate redundant or irrelevant features, enhance computational efficiency, and facilitate visualization of the data.

Association rules is another significant unsupervised learning technique, following clustering, which identifies intriguing connections (associations, dependencies) among extensive collections of data elements. These elements are typically stored as transactions, which can be generated by an external process or extracted from relational databases or data warehouses. With the association rules algorithms demonstrating strong scalability and the continuous expansion of data volumes, association rules serve as a crucial data mining tool for extracting valuable insights from data [13].

2.3.1.3 Reinforcement Learning

Reinforcement Learning is another subset of ML that allows a system to learn through trial and error using feedback from its actions. During the learning time, an is expected to be able to choose the actions associated with a greater reward and avoid the ones that generate a punishment, maximizing a cumulative reward. The set of actions carried out by the agent is called policy and the main goal of RL systems is to learn an optimal policy.

This methodology is very popular in robotics industry, especially in self-driving cars field, recommendation systems and gaming [14].

2.3.1.4 Machine Learning Algorithms

A. Logistic Regression

Logistic Regression (LREG) is one of the most simple and traditional methods in machine learning to resolve binary classification problems. Logistic Regression works like a common linear regression but applies a sigmoid function (2.1) to model the output into a value between 0 and 1 [15]. All input variables are exposed to a weighted transformation (multiplying with the weights) and then added up. The final output represents the probability of the input sample to belong to a certain class [16].

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

B. Naive Bayes Classifier

As logistic regression, the Naive Bayes Classifier is another known and simple classifier in the machine learning industry. This classifier, uses the Bayes' theorem assumption that all

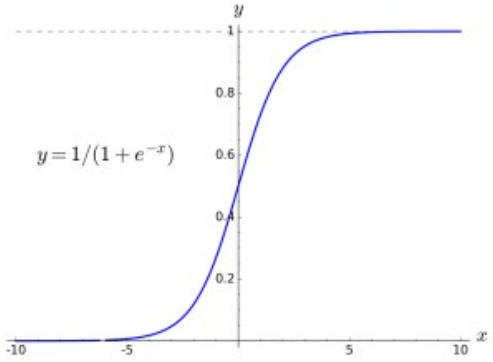

Figure 2.3: Graphic representation of sigmoid function [17]

features are conditionally independent and contribute equally to the final output, which simplifies the calculations.

$$\mathrm{P}\,(\mathrm{H}\,|\,\mathrm{X}) = \frac{P(X|H) \cdot P(H)}{P(X)} \tag{2.2}$$

Where,
H - hypothesis, represents one of the output classes
X - evidence, describe by measure on set of attributes
P(H | X) - the probability of hypothesis H, given the evidence X
P(H) - prior probability of H
P(X | H) - probability of X conditioned on H
The final result, R, is given by the ratio between the two probabilities of X belonging to one of the possible classes H [18].

$$\mathrm{R} = \frac{P(Hi|X)}{P(Hj|X)} = \frac{P(Hi) \cdot P(X|Hi)}{P(Hj) \cdot P(X|Hj)} \tag{2.3}$$

With equation 2.3, it is simple to verify that if R>1, the predicted class is *i*, otherwise the predicted class is *j*.

C. Decision Tree and Random Forest

The decision tree, besides being a classifier itself, is the base learner technique of the most powerful classification algorithms for tabular data. It is a rule-based algorithm that creates a tree-like structure of decisions based on the input features and starts with a single node (root node), which contains the dataset. A feature that provides the best split for the data is selected and the possible values of it become a branch (decision node). The data is divided into subsets based on the value of each input in the decision node and the process is recursively repeated until the stopping conditions are met (maximum depth reached, for example), known as leaf nodes. Once the tree is constructed, new inputs

can be classified (traversing the tree from the root node to a leaf node) determining the majority class in the corresponding node.

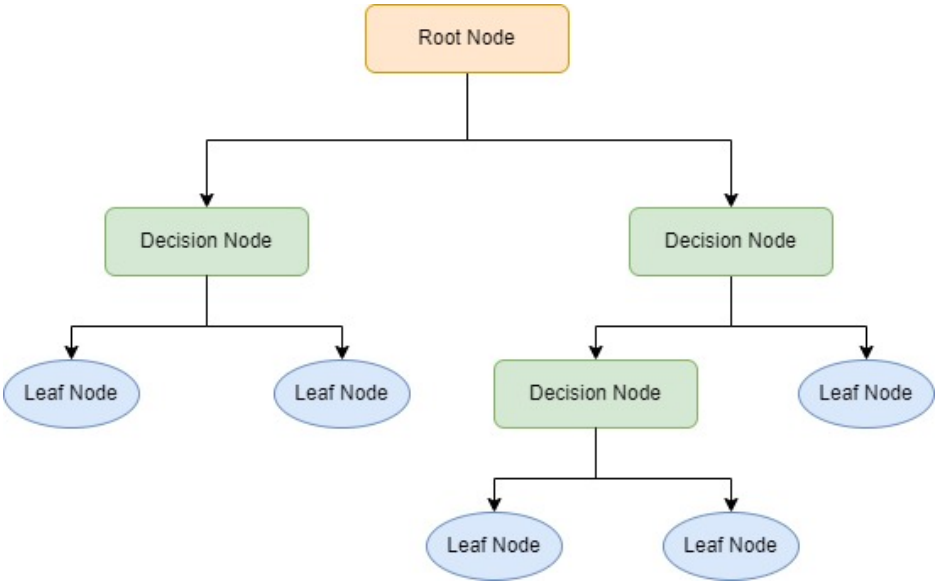

Figure 2.4: Architecture of a decision tree

Random Forest is one of the most used algorithms that uses ensemble techniques to make predictions. Ensemble classifiers combine the result of several classifiers (base learners), normally through a voting process [19]. In Random Forest case, the base learners are decision trees that are trained in a sample of the training data, and a different set of features is selected (bagging) for each tree to use as decision nodes. The final prediction of each tree is named as a "vote" and the class with more votes is selected as final result.

D. AdaBoost, LightGBM and XGBoost (Boosting algorithms)

On the other hand of bagging techniques used by algorithms such as Random Forest, another ensemble technique was developed, known as boosting, in other to improve the performance of these classifiers. While bagging algorithms train its base learners in parallel, boosting techniques use a sequential training in order to transform its weak learners into stronger ones, combining them following a specific strategy.

AdaBoost, which stands for Adaptive Boosting, was the first algorithm developed using a boosting technique. This algorithm trains a base learner over a weight-data sample created by evaluating the training error of the predecessor learner. This process is repeated recursively, minimizing the training error and combining all the learners outputs to make the final decision.

While AdaBoost tries to fit the new predictor with the residual error made by the previous one, gradient boosting algorithms use a strategy focused on optimizing the loss function. These algorithms try to get the best model since the beginning instead of correcting errors sequentially and its use has grown exponentially due to its high

performance in tabular data analysis. They are known as gradient boosting machines and LightGBM and XGBoost (Extreme Gradient Boosting) are two examples of these algorithms. While LightGBM is generally faster and more efficient, making it suitable for large and complex datasets, XGBoost is more focus on predictive performance, generally getting better results.

2.3.2 Deep Learning

Deep learning is a subfield of machine learning that focus on training artificial neural networks to learn and make predictions or decisions from complex and large-scale data. It involves building and training deep neural networks that are composed of multiple layers of interconnected nodes, called artificial neurons or units. These neural networks are designed to automatically learn hierarchical representations of data by progressively extracting higher-level features from lower-level ones.

Unlike traditional machine learning algorithms, deep learning models can automatically learn and discover intricate patterns and representations from raw or unstructured data, without requiring explicit feature engineering. This ability to automatically learn and extract features makes deep learning particularly effective in domains where the data has complex structures, such as images, audio, text, and sequences [20].

Deep learning has achieved significant breakthroughs in various domains, including computer vision, natural language processing, speech recognition, and recommendation systems. The power of deep learning lies in its ability to effectively model and learn from complex and high-dimensional data, enabling the development of highly accurate and sophisticated AI systems.

2.3.2.1 Artificial Neural Network

Artificial neural networks belong to a family of models that tries to replicate the functioning of the human brain, learning and processing information like it does. Their effective integration of several mathematical, logical, and computational techniques allows them to dig deeper on the input data, which frequently makes them quicker and more effective than other traditional machine learning models that do not use neural networks [21].

An ANN consists in interconnected neurons, organized in various layers. Each neuron receives an input from several neurons of the previous layer, and produces an output, which is sent to the neurons of the next one. This communication between neurons is weighted, and the learning process of a neural network relies on adjusting these weights. As represented in figure 2.5, a basic ANN architecture consists in an input layer, few hidden layers and an output layer. The input layer has one neuron for each dataset feature and receives the raw input ingested from the dataset. The outputs produced by the input layer are sent to the hidden layers, the intermediate ones, that process these signals weighting them and applying activation functions [22]. The activation function receives as input the sum of all weighted signals sent to the respective neuron and applies

a non-linear to create the neuron's output [23]. If this output is above a defined threshold, the neuron is "activated" and fires it to the next layer neurons.

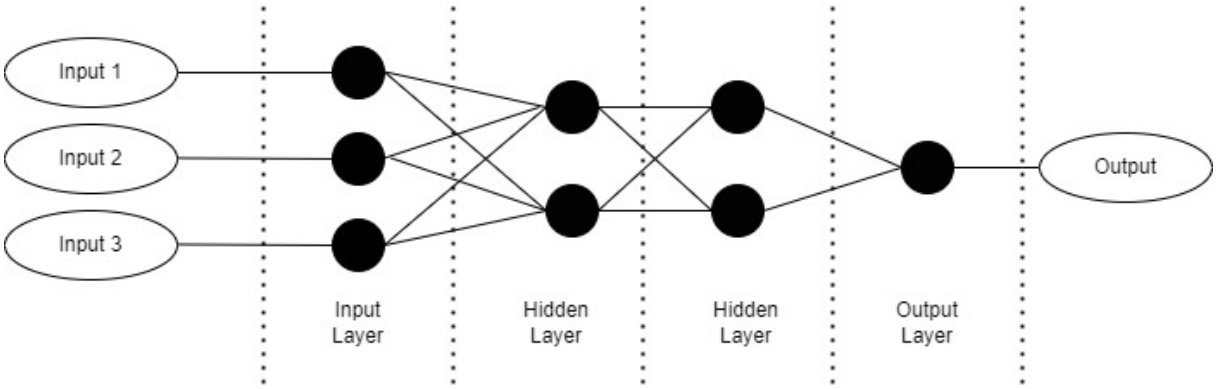

Figure 2.5: Basic ANN architecture

Choosing the right activation function is one of the most important processes when building an ANN. The most common functions are rectified linear unit (ReLU) and sigmoid function and both of them apply non-linearity to the neural network. ReLU converts all negative values to 0, blocking the information in these neurons, and keeps the positive ones. The sigmoid function ranges it values between 0 and 1 and is commonly used in classification tasks, when a probability is needed as an output. The behaviour of both functions can be observed in figure 2.6.

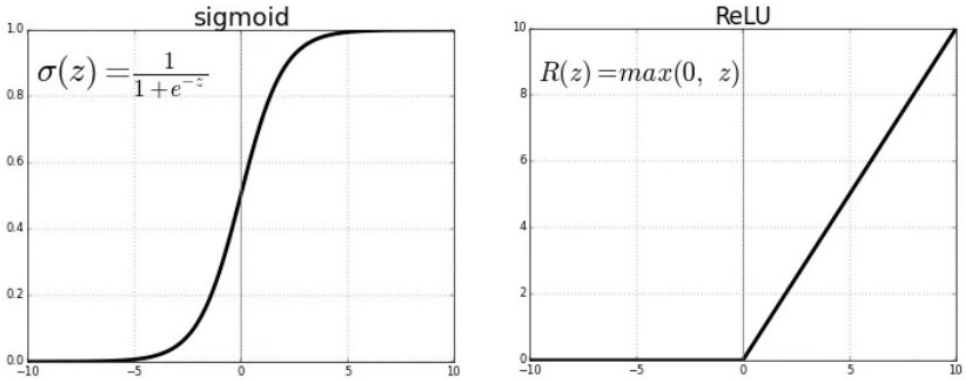

Figure 2.6: Sigmoid and ReLU functions. Adapted from [24]

2.3.2.2 Deep Learning Architectures for Tabular Data

NODE

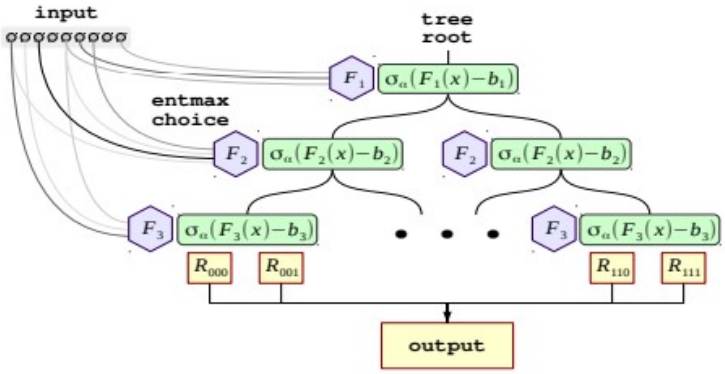

Figure 2.7: Representation of a neural oblivious decision tree. [25]

NODE, which stands for Neural Oblivious Decision Ensembles, it is an architecture that uses a neural implementation of oblivious trees [26]. An oblivious tree is a regular decision tree that grows symmetrically, which means that this type of trees use the same features combination and decision-threshold for all the internal nodes. Due to these conditions, ODT's become a much more weaker learners than regular decision trees. Its strength lies in their ensemble where the overfitting is greatly reduced, due to the low complexity of each learner, improving the final results specially when exposed to new data [25].

The combination of various ODT's as weak learners, concatenating their outputs, is called a NODE layer and its output serves as input of the subsequent layer. This dense and sequential connection of layers represents a multi-layer architecture of neural oblivious trees ensemble, known as NODE architecture.

.

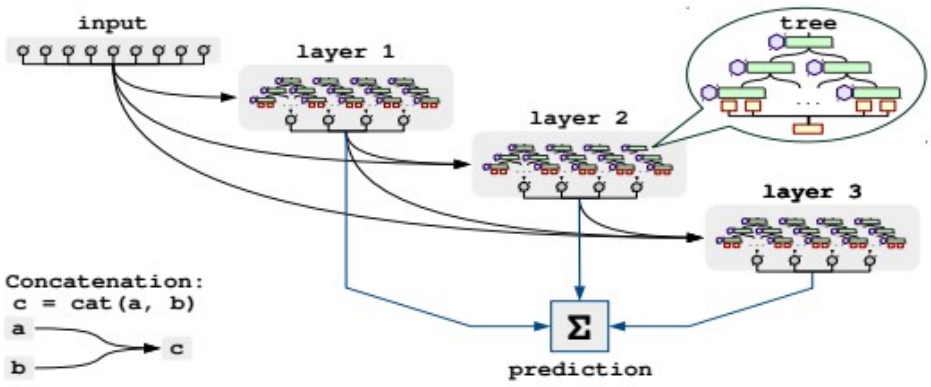

Figure 2.8: NODE architecture

TabNet

TabNet is another architecture that uses decision trees as it weak learner. Unlike NODE, TabNeT uses differentiable decision trees and the model strength lies in the mechanism used for feature selection. Instead of using all features available, TabNet has an attention mechanism to choose the best features for the split and assign different weights to the them, applying higher weights to the most important ones. The sequential stacking of decision trees allows the model to update the features importance in each decision step, leading to a better model training and better final predictions.

TabNet is specifically used for tabular data analysis, working on classification and regression tasks.

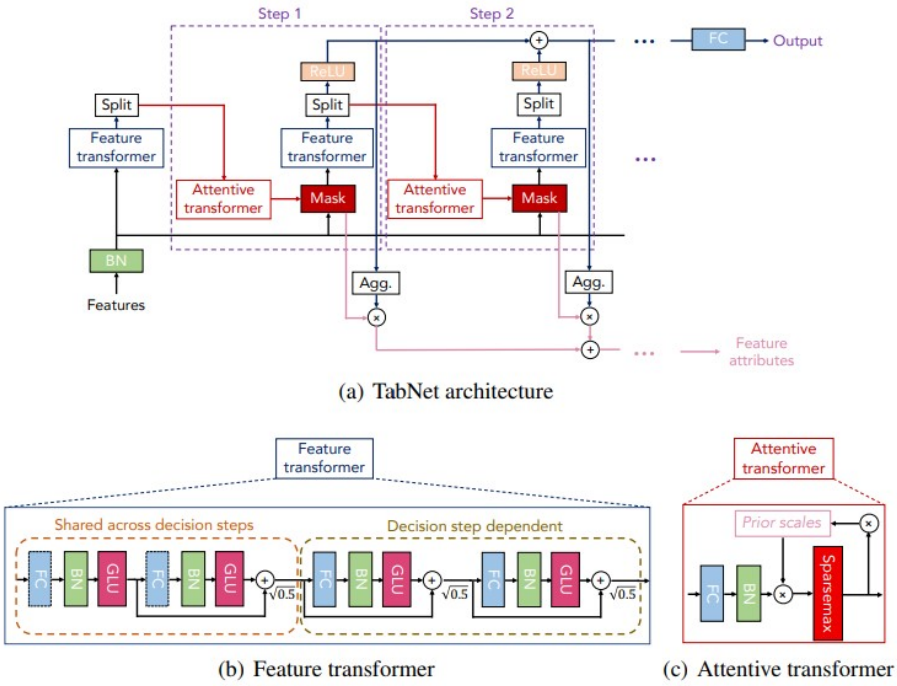

Figure 2.9: Overall architecture of TabNet model. [27]

Wide and Deep

Wide and Deep is an hybrid architecture that combines two different components: a wide model and a deep model. The wide model consists of a linear model, responsible for capturing simple interactions between features (normally between binary features) and adding a memorization component to the model. The deep part is a feed-forward DNN responsible for capturing more complex relationships in continuous and categorical features, adding a generalization component to the model. The outputs log odds of both are combined in a weighted sum, resulting in the final prediction [28]. By combining these two components, Wide and Deep models normally achieve high performance levels and is a technology that has been widely adopted in various industries.

Wide & Deep was originally developed by Google to be used in recommender systems. Nowadays this architecture is also used in other tasks, such as classification and regression.

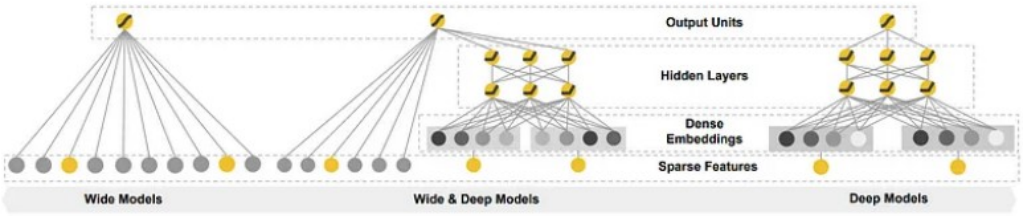

Figure 2.10: Wide models vs Wide and Deep models vs Deep models

DeepFM

Similar to Wide&Deep, Deep Factorization Machines (DeepFM) consist in a parallel merging of a deep component with an FM component. A multilayer perceptron (MLP) is utilized in the deep component to record high-order feature interactions and nonlinearities while the FM component is responsible for the low-order features interactions. The final prediction is the culmination of the outputs of these two components, which share identical inputs and embeddings [29]. It is important to note that DeepFM has characteristics with the Wide&Deep architecture, which is capable of capturing both generalization and memorization. By automatically identifying feature combinations, DeepFM decreases the effort required for hand-crafted feature engineering, which gives it an edge over the models that need this manual craft [30].

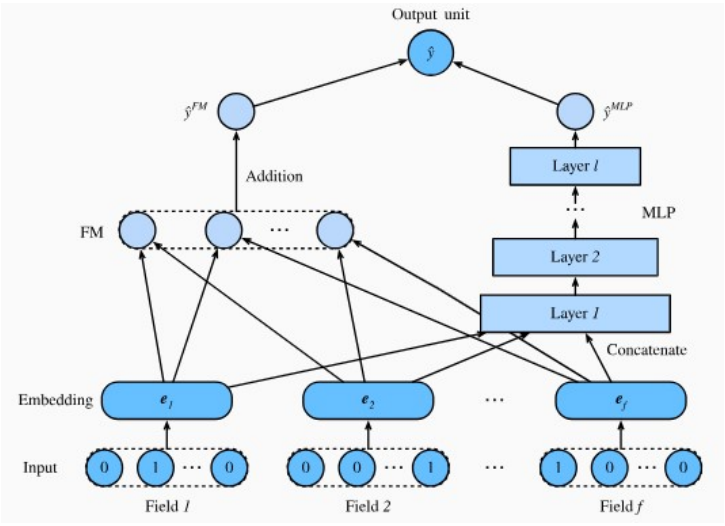

Figure 2.11: DeepFM architecture. Adapted from [30]

2.4 Imbalanced Datasets

In real world applications, most of the binary classifications that uses a machine learning approach found a very common issue within this type of problems: the class imbalance.

An imbalanced dataset is defined as a dataset that has an unequal distribution of the target class. Detection of fraudulent transactions, spam filtering, diagnosis of diseases or text categorization are some examples of problems that use extremely unbalanced datasets [31]. In all of these examples, the minority class is always the one that is more important to be identified correctly, and there is where the imbalance between classes becomes a problem because when building a machine learning model to make predictions over an imbalanced dataset, the expected behaviour is that the model gives more importance to the majority classes, becoming biased to classify this one and having a poor performance in the minority class.

It is important to have this problem in mind when working with imbalanced datasets, and some techniques can be applied to address this situation.

2.4.1 Resampling

Resampling is the most common and simple method to handle imbalanced datasets and can be applied by two different strategies: oversampling and undersampling.

Oversampling is defined by the random duplication of the minority class samples, balancing the distribution of both classes. In cases that the two classes are extremely imbalanced, applying random oversampling can lead to a lot of data duplicated and the chances of the model having an overfitting behaviour are increased [32].

Undersampling does exactly the opposite. This method selects a random subset of the majority class, ignoring the rest of the samples and balancing it with the minority class set. Applying this method can reduce significantly the dataset size, improving the training process performance but can lead to a loss of useful information contained in the discarded samples.

In order to balance the tradeoff between both techniques disadvantages, hybrid methods (using oversampling and undersampling at the same time) can also be applied [33].

2.4.2 SMOTE

SMOTE, which stands for Synthetic Minority Oversampling Technique, is another resampling technique. The standard resampling methods, discussed above, randomly resample the data which often does not lead to a gain of useful information. Otherwise, SMOTE techniques try to measure the minority class properties in order to create synthetic samples [34], overcoming the overfitting problem caused by random sampling.

Despite the advantages of using SMOTE, not all datasets are suitable for its application (categorical features, for example). To overcome these situations, several derivations of the

method have been developed, such as SMOTE-NC, SMOTE-ENN or Borderline SMOTE, each one adjusted to its own application.

When using resampling methods, especially those that generate synthetic data like SMOTE, it is important to be sure of the timing of the resampling application. The data should be first divided into the training and test/validation set and the resampling should be only applied in the training set. Doing this, the synthetic data will not have samples originated by the test set.

2.4.3 Threshold-Moving

Imbalanced datasets handling techniques are not all part of the data pre-processing phase. The threshold-moving case is a method that switches the way of how the binary classification values are attributed. The baseline threshold applied in standard predictions is usually 0.5, whereas if sample probability is >0.5, the sample belongs to class 1, otherwise it belongs to class 0.

In some cases, the analysis of the tradeoff between true positive rate and false positive rate, using an AUC-ROC curve plotting, shows that the standard decision threshold of 0.5 is not the optimal one to evaluate the model performance. Increasing or decreasing the threshold can improve the prediction results, specially if the metric used to evaluate the model is precision or recall.

2.4.4 Algorithm Selection

The selection of the prediction algorithm is also a moment that can make impact in the performance of the model when using imbalanced datasets, especially if no resampling techniques were used during the pre-processing phase. Normally, boosting algorithms with tree-based weak learners perform well on these datasets, due to the sequential training of the learners, which increases the importance given to the minority class. Algorithms such as XGBoost, LightGBM or even Random Forest are normally good options to handle imbalanced datasets.

2.5 Evaluation Metrics

2.5.1 Accuracy

Accuracy is probably the most famous and intuitive metric for classification problems. This metric simply calculates the ratio between the correct predictions and the total predictions.

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \tag{2.4}$$

Despite its easy and intuitive use, accuracy it is not always a reliable indicator to evaluate model's performance. In imbalanced datasets (datasets where the target variable classes

distribution is not balanced) this metric can be misleading, since the model may be predicting perfectly the majority class but failing in minority class prediction.

2.5.2 Confusion Matrix

In order to avoid misleading, the best way to evaluate a model's performance is analyzing the confusion matrix of the prediction results. Confusion matrix is a tabular representation of the results, divided by four groups (if the target variable is binary): True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

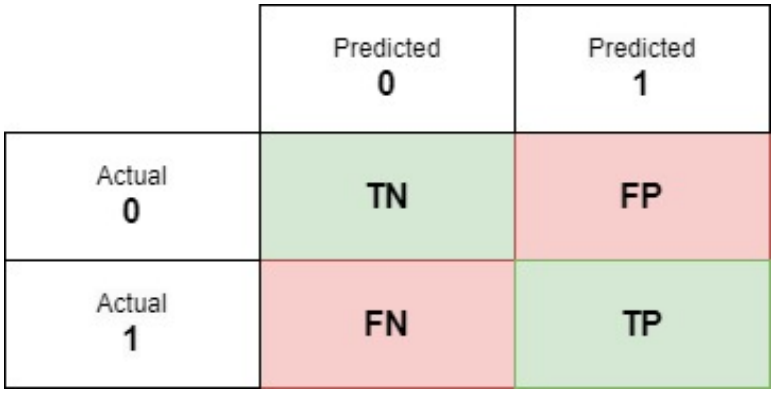

Figure 2.12: Confusion matrix example.

True Negative (TN) - Negative sample predicted correctly.
False Positive (FP) - Negative sample predicted incorrectly, as positive.
True Positive (TP) - Positive sample predicted correctly.
False Negative (FN) - Positive sample predicted incorrectly, as negative.

With this representation, better and more complete conclusions can be made about the results, using some metrics that use the confusion matrix groups, such as, precision, recall ans specificity.

2.5.3 Precision

Precision metric, as the name implies, calculates the precision of the model in its positive class predictions. It calculates the ratio between actual positive predictions and all positive predictions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.5}$$

2.5.4 Recall

Recall, that can be referenced as sensitivity or true positive rate, measures the ability of the model to identify positive cases or, in other words, evaluates the model's effectiveness to correctly predict all actual positive cases.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.6}$$

2.5.5 Specificity

Specificity, or true negative rate, complements recall. This metric calculates the effectiveness of the model to predict actual negative cases.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.7}$$

2.5.6 F1 Score

In most of classifications models, specially those based on imbalanced datasets, it is impossible to improve recall and precision at the same time, and there is always a need to give up in one metric in order to improve the other. This problem is called Precision-Recall trade-off.

The goal of F1 Score is exactly to combine precision and recall in one metric, calculating the harmonic mean between them.

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.8}$$

2.5.7 AUC-ROC Curve

AUC-ROC curve is a classification metric that stands by the combination of AUC (Area Under the Curve) and ROC (Receiver Operating Characteristics). This metric plots true positive rate (y-axis) against false positive rate (x-axis) and calculates the area below the curve. This area represents the ability of the model to distinguish between classes.

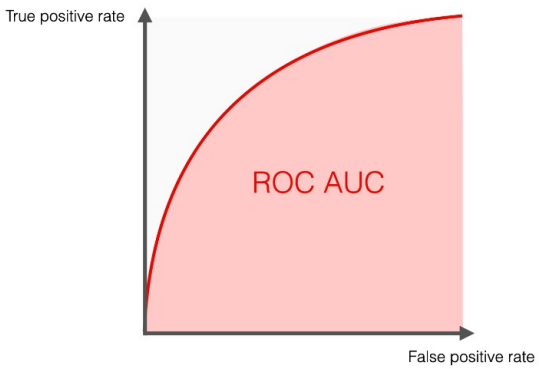

Figure 2.13: AUC-ROC curve example [35]

An AUC of 1.0 means a perfectly fitted model, with total distinction between classes. In the other side, an AUC of 0.5 means the opposite which leads to random predictions.

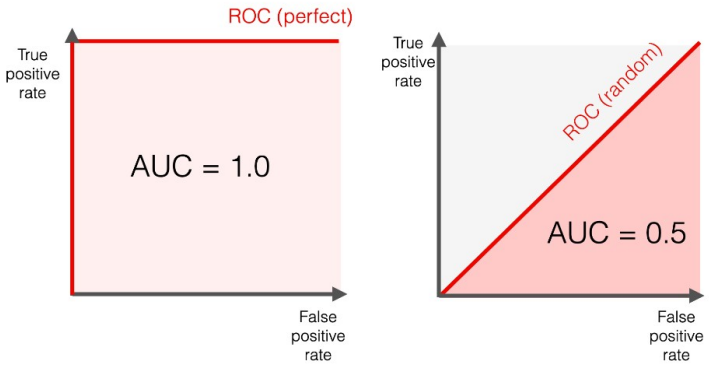

Figure 2.14: Graphic representation of AUCs with 1.0 and 0.5 values [35]

2.6 Underfitting and Overfitting

2.6.1 Relation between Bias, Variance, Underfitting and Overfitting

Bias and variance are the key concepts behind the underfitting/overfitting problem and understanding their fundamentals is essential for assessing the viability of a machine learning model.

Bias, in a machine learning context, refers to the error of trying to make assumptions about a complex problem with a simplistic model. This error is caused by the inefficiency of the model to understand the relationships between the independent variables and the output target which leads to a poor performance of the model in training and test sets. This behavior is characterized by a high bias, which indicates underfitting.

In other hand, variance is the sensitivity of the model to fit the training set to closely. A model with an high variance is normally an excessively complex model and trained on noisy data which leads to a good performance on the training set but a failure when it tries to generalize on unseen data (test set) [36]. This is the definition of overfitting.

When building a model, is extremely necessary to pay attention to these concepts and try to find a balance between them, which is normally defined by bias-variance tradeoff. The ideal tradeoff is the point where the total error (sum of squared bias and variance) is minimized [37].

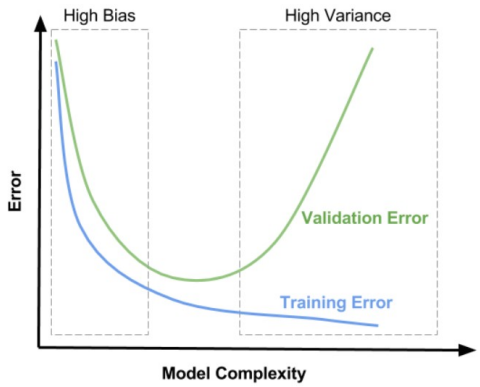

Figure 2.15: Representation of bias-variance tradeoff. [38]

2.6.2 Techniques to avoid underfitting and overfitting

The causes of underfitting and overfitting are not always due to the techniques used during the model training. In some cases, the problems can be caused right from the construction and structuring of the dataset, the methods used in pre-processing and feature engineering phases or even from the high complexity of the problem.

That said, it is important to explore the entire model's pipeline as much as possible in order to understand the reasons behind the underfitting/overfitting behaviour and to figure out how to correct them.

Underfitted models

Underfitted models are much easier to understand than overfitted ones. Most of the time, the causes are related to the dataset size or simplicity. The features presented on the data, may not be enough to describe the complexity of the problem or, conversely, the model may be too simple for the complexity of the data. In the pre-processing phase, differences in data scale between train and test datasets can be another problem.

To avoid underfitting, it is important to be sure that enough data is collected and the features used are the correct ones. Optimizing the hyperparameters of the model, to increase it complexity, is another method to train the ideal model and reduce the error.

Overfitted models

Unlike an underfitting model, understanding what might be causing overfitting can be more trickier due to the wide variety of possibilities. In some cases, increasing the training data or reducing the complexity of a model can be enough. Using ensemble algorithms like Random Forest is another solution due to the capacity of this algorithms to process complex models and to improve generalization, always align with an adjusted hyperparameters optimization. If the model complexity is too high, performing better feature engineering methods (creating new variables or deleting others) can be effective as well. Another solution is to identify and remove outliers that can produce noise to the data.

In ANN, a common approach to reduce the overfitting of a model is to use a dropout layer. Adding this layer to the network is a regularization technique that randomly drops a set of neurons and connections (for each input) during the training phase, improving the generalization of the model. This technique is mainly effective in large and complex networks, that need a lot of training data [36].

2.7 Related Work

2.7.1 Booking Cancellations Forecast Using Machine Learning

Ivanov, et al. [39], recognized that forecasting is a crucial aspect of revenue management, especially demand forecasting and since 2016, Nuno, et al. [1], has developed some studies focusing on hotel booking cancellations forecast with ML-based models. In the research mentioned, the authors built a dataset with over 120 thousand entries of hotel bookings information from four different hotels and applied ML algorithms to build a model to forecast cancellations. Overall, the models developed by the authors achieved pretty good results, with Accuracy ranging from 0.78 to 0.89, Precision from 0.57 to 0.80 and AUC from 0.78 to 0.93, with XGBoost being the algorithm that achieved better results. The dataset used during this research was published afterwards, in Kaggle platform, which proved to be very important in booking cancellation forecasting literature review, since this type of information is extremely sensitive and hard to collect.

The publishing of the dataset on an online platform allowed more authors to join in the research of this case of study and many papers started to be published with proposed models for booking cancellations forecast. Saputro and Nanang, in [40], also achieved an Accuracy score of 0.88 with Random Forest and Extra Tree Classifier but in their research, they addressed the problem of Recall metric. Besides the high accuracy obtained by their models, the Recall value was still ranging from 0.70 to 0.79. Since an unbalanced dataset is being used, this research exemplifies that the high accuracy results are being highly influenced by the prediction of non-cancelled bookings, which shows that the Recall metric has to be used in the evaluation of the models. Adil, et al. [41], addressed this imbalance problem and performed some pre-processing techniques in the dataset in order to study the impact of it in the prediction results. The combination of Random Forest algorithm with a dataset balanced with SMOTE-ENN technique, achieved scores of 0.96 in Accuracy, 0.95 in Recall and 0.96 in AUC.

In contrast to all the studies mentioned above, Huang, et al. [42], used deep learning to build their models. In this research, a back propagation and a regression neural network were developed and achieved a Recall score of 0.69 and 0.75 respectively. Besides the different approach applied by the authors, the results did not exceed those obtained with machine learning models.

Despite all these studies showing that cancellation forecasting is possible, applying these models to real cases still raises some questions. As mentioned before, a large percentage of these studies use the dataset created by Nuno in [1] as a base, which raises the question of the applicability of the models to data acquired by different hotels and sources. In [43], Falk and Vieru studied the nature of booking cancellations behaviour and concluded that cancellations are mainly related to variables like booking lead time, country of residence and the channel where the booking was made. If the structure of the public dataset cannot be matched, researchers can use this study to get an idea of the

main factors influencing the cancellation of a booking.

2.7.2 Machine Learning vs Deep Learning on Tabular Data

Besides the huge impact of deep learning in AI field during the recent years when dealing with image, audio and text data, his superiority to machine learning algorithms, especially tree-based ones, on structured and tabular data is still questionable [44].

Borisov et al. [45], in 2021, reviewed the results of many deep learning architectures in five different datasets and compared them to tree-based ML algorithms. In this research, algorithms such as XGBoost, CatBoost and LightGBM outperformed the deep learning architectures in four datasets, with SAINT being the only DL architecture to get better results in HIGGS dataset [46]. Although it is not clear, this better performance in HIGGS dataset can be explained by the dataset's dimension with over 11 million samples in contrast to the other datasets with few dozen samples. Another conclusion that can be made with this research is about the differences in training and inference time. For the low/medium size datasets, both times were generally lower for ML algorithms, but in HIGGS dataset, the inference time of XGBoost was clearly higher than the others.

In 2022, Grinsztajn et al. [44] tried to identify the main reasons for the constant outperformance of tree-based models on tabular data. Firstly, the authors referred that "the lack of an established benchmark" and the small size of most of the available datasets do not allow a clear and constant evaluation of deep learning methods. To suppress this problem, the authors created a benchmark with 45 datasets with specific characteristics, where datasets that contain different column types, real world data, datasets with at least four features and 3000 samples and datasets that cannot be resolved with simple models (a single decision tree or a regression, for example) were used. This step was crucial to maintain the criteria and value of the evaluations made further. The models used in the research for comparison were XGBoost, GradientBoostingTrees and Random Forrest for the tree-based side and a MLP model, Resnet and a FT-Transformer for the deep side, all created by Gorishniy et al. [47]. Since the model's performance matched the results of the previous research, where tree-based models got better results, the authors tried to analyze in more depth in order to understand why the tree-based models outperformed the deep ones. Firstly, the authors demystified that one of the main weaknesses of neural networks was the categorical variables. Borisov et al. [45] stated that "one of the critical obstacles for deep learning with tabular data is categorical variables", which was not the case in this research when the models were tested only on categorical variables. Finally, the authors discovered that deep neural networks-based models are "not robust to uninformative features" since the performance gap between both increased whenever these were added. This statement is very important for the study of deep learning models in tabular data, since uninformative features are very common in these datasets. It is important to note that the conclusions made in this research were only considering balanced datasets with accuracy as evaluation metric. The authors mentioned that other evaluation metrics

should be used, especially when dealing with imbalanced datasets, to get even more complete conclusions.

Finally, Shwartz-Ziv and Armon [48], as well as having proven the best results with XGBoost and mentioned the higher computational costs of training deep learning models, they also proposed an ensemble of neural networks with XGBoost that outperformed single models in seven of the eleven datasets tested. The authors also highlighted the importance of studying the constraints of each model when using them in real life applications.

3

Methodology

The pipeline adopted for the development of this project is based on CRISP-DM methodology with six main phases: data cleaning, data understanding, data preparation, modelling, evaluation and best model selection and analysis.

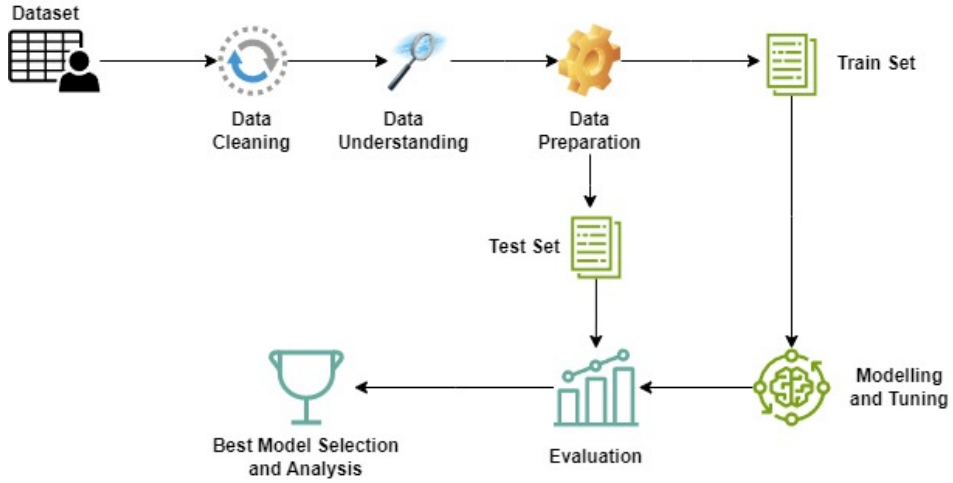

Figure 3.1: Representation of the proposed methodology

3.1 Dataset

The dataset used for the model's development was extracted from an hotels chain present in HHS' systems. This dataset is a csv file with 51660 bookings from three different hotels within the same group, that has a check-in date ranging from January 2022 and April 2023. The main reason for choosing this start time window is the attempt to avoid the impact on booking's behavior caused by Covid-19 pandemic. The end of the window was set to the day before data extraction, since the final status of bookings with check-in after April is uncertain.

The dataset contains 23 initial features of mixed types available, with ReservationStatus being the target feature, as shown in table 3.1.

Table 3.1: Reservations Dataset Description

Variable	*Type*
HotelId	*Categorical*
ResIdDate	*Categorical*
SegmentCode	*Categorical*
SubSegmentCode	*Categorical*
DistChannelCode	*Categorical*
Adults	*Numerical/Discrete*
CheckIn	*Categorical*
CheckOut	*Categorical*
ChannelRoomTypeCode	*Categorical*
ReservationStatus	*Categorical*
AmountAfterTax	*Numerical/Continuous*
Children	*Numerical/Discrete*
Infants	*Numerical/Discrete*
Source	*Categorical*
Voucher	*Categorical*
daystochekin	*Numerical/Discrete*
los	*Numerical/Discrete*
arrival date week number	*Categorical*
checkin date week number	*Categorical*
rate	*Categorical*
Gender	*Categorical*
Nationality	*Categorical*

3.2 Data Cleaning

The predictive performance of any model is directly related to the quality and relevance of the data presented in the dataset what turns the data cleaning phase into the most important phase of any model's building pipeline. During this phase some initial pre-processing steps are made, such as the removal of duplicate samples, the outliers and missing values handling and some syntax corrections are also made. The reason for performing this phase before the data understanding is because the author believes that the data should be cleaned before the exploration part, so that it can be carried out on data with a structure close to that which will be exposed to the models.

3.2.1 Removing duplicated and useless samples

During data collection process, the duplication of data samples are very often, especially when data from different sources are combined. The presence of duplicated samples may misrepresent the true characteristics of the dataset and can introduce bias during the training phase. Looking to the table 3.2, 37.96% of the dataset is duplicated data, which would have several negative impacts on model training if it wasn't removed.

Total Samples	Duplicated Samples	Percentage of Duplicates (%)
51660	19608	37.96

Table 3.2: Percentage of Duplicated Samples in the Dataset.

Two data samples are considered duplicated when the value in the ResIdDate feature, which contains the booking's registration date down to the millisecond, is exactly the same in both, as is the rest of the booking variables.

3.2.2 Handling Missing Values

Since most machine learning methods cannot handle missing values independently, this is a necessary step in data cleaning phase. The most common techniques to handle missing values are direct deletion and imputation.

Imputation occurs when the missing value is substituted by another calculated value. Normally, statistical methods are used in the column to calculate the value to input, such as median, mean or mode. These are simple imputations but they can be very efficient in skewed datasets. Other more advanced techniques can be used like KNN imputation, where the value to input is chosen on the basis of information obtained by observing the k-nearest neighbors, or Linear Regression imputation, where the value to input is calculated by applying a linear regression using the other features as independent variables.

Direct deletion is normally used when the percentage of null values is extremely high or low. This technique is divided into another two: listwise deletion and column-wise deletion. Listwise deletion is used when the percentage of null values in that feature is very

low and the samples that contain null values are not necessary, leading to their deletion. In contrast, column-wise deletion is used when the feature has an high percentage of null values, making it an useless feature. In this case it is preferable to delete the column from the dataset.

	Null Values	Null Values (%)
HotelId	0	0.000
ResIdDate	0	0.000
SegmentCode	4	0.012
SubSegmentCode	4	0.012
DistChannelCode	5	0.015
Adults	0	0.000
CheckIn	0	0.000
CheckOut	0	0.000
ChannelRoomTypeCode	0	0.000
ReservationStatus	0	0.000
AmountAfterTax	0	0.000
Children	0	0.000
Infants	0	0.000
Source	17126	53.412
Voucher	29746	92.805
daystochekin	0	0.000
los	0	0.000
arrival_date_week_number	0	0.000
checkin_date_week_number	0	0.000
rate	19	0.059
Gender	7564	23.600
Nationality	898	2.802

Table 3.3: Percentage of Missing Values by Feature.

Analysing the missing values in the dataset, represented in table 3.3, different types of behaviours can be observed. Features SegmentCode, SubSegmentCode and DistChannelCode have an extremely low percentage of null values with values around 0,012% and 0.015%. In this case, a listwise deletion will be applied, deleting the samples from the dataset. Features Voucher and Gender, in contrast, have a very high null values percentages, with values of 92,805% and 23,600%, making it preferable to drop them from the dataset instead of the samples. In Nationality and Rate, the null values will be imputed with the most common value, since "PT" has a frequency of 30% in Nationality column, and "APA|BB" a frequency of 60%. In these cases, the most common category have a high dominance in the dataset, and imputing the few null values with it will not impact negatively the feature's distribution. The last modification in this phase is related to the feature Source. This feature has a null values percentage of 53,412% and the normal process would be to delete the column from the dataset. It turns out that this variable can add value to the model, even with this high percentage of null values. In this case, the

null values will be replaced by a new "UNK" category, representing an unknown value. Later on, this feature will be analysed in more detail to see if this modification has had a positive impact on the model.

3.2.3 Handling Outliers

As mentioned at the beginning of this chapter, data quality is crucial for the positive performance of any predictive model. The presence of samples with characteristics that lie far from the rest of the dataset greatly diminishes the quality of it, which can have a significant impact during the training phase. This type of sample is known as outliers and typically occurs due to errors in data recording or when rare events occur. There are various techniques to identify the presence of outliers with statistical methods like the calculation of z-scores or the use of interquartile range (IQR) being the most famous.

Interquartile range (IQR) is a statistical measure and it is defined by the difference between the third and first quartiles. IQR helps the calculation of two extremes of the dataset called lower fence and upper fence, that defines the window of acceptable values. All values that fall outside this window are considered outliers.

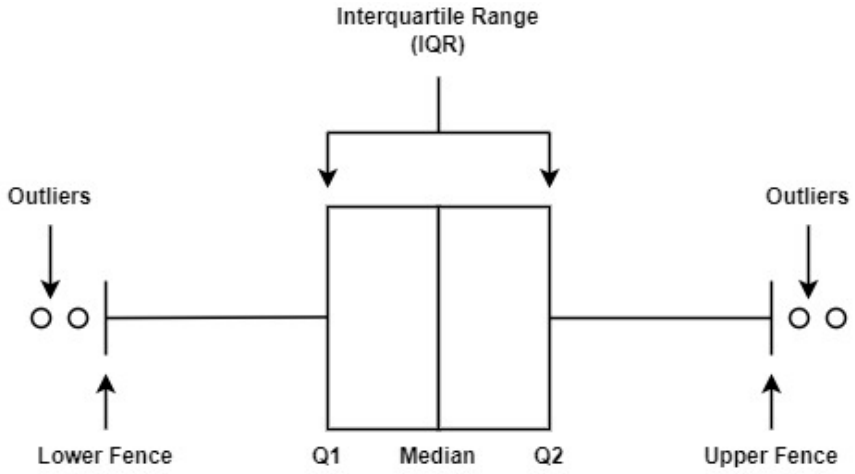

Figure 3.2: IQR

$$\text{IQR} = \text{Q3} - \text{Q1} \tag{3.1}$$

$$\text{Lower Fence} = \text{Q1} - (1.5 \cdot \text{IQR}) \tag{3.2}$$

$$\text{Upper Fence} = \text{Q3} + (1.5 \cdot \text{IQR}) \tag{3.3}$$

Before applying the method, it is important to note that the feature Adults was engineered to be the sum of Adults, Children, and Infants. It was done due to the high percentage of bookings without Children and Infants (98% and 99%), making it more sense to aggregate the three features for the application of the method.

To use this method in bookings dataset, some particularities need to be addressed carefully. Normally, the multiplier applied to IQR in equations 3.2 3.3 is 1.5, but in this case it would lead to a low values in upper fences, causing the identification of a large number of outliers that theoretically might not be outliers. The upper fences for a multiplier of 1.5 are 3.5 for Adults, 130.5 for daystochekin and 6 for los, as shown in table 3.4. Using these fences, bookings made 131 days in advance or bookings with a length of stay of 7 nights would be considered as outliers, what it is not true. Although these bookings aren't that common, they do not happen because of mistakes either.

To address this situation, a multiplier of 5.5 was chosen in order to increase the upper fences values. With this multiplier value, bookings of up to 7 people, lasting up to 20 days and booked up to 232 days in advance were considered acceptable. If any of these requirements were not met, the booking was considered an outlier and was removed from the dataset.

	Q1	Median/Q2	Q3	IQR	LF (3.5)	UF (1.5)	UF (5.5)
Adults	1.0	2.0	2.0	1.0	-4.5	3.5	7.5
daystochekin	3.0	19.0	54.0	51.0	-276.0	130.5	348.0
los	1.0	2.0	3.0	2.0	-15.5	6.0	20.5

Table 3.4: Statistical values for IQR method.

Another particularity is that Q1 values are very low, leading to negative values in lower fences after applying this method. Since these features need to be strictly positive, which means that negative values rely to recording errors, lower fences were forced to be zero in all features. The application of IQR, with the specified modifications, in this dataset identified and deleted 616 samples with daystochekin and los being highly impacted features, what can be seen with the comparison of features' boxplots, before and after removing outliers.

3.2.4 Syntax and Final Corrections

During the outlier removing phase, most of the inconsistencies in the data were cleared up, leaving only a few details in certain features unresolved.

Looking at the TotalGuests feature, for example, four samples with a zero were found, what is not possible and probably occurred as a result of a data entry error.

Another case that needs to be handled is the samples with a value of "$0.00" in AmountAfterTax feature. There are 409 cases in which no amount paid has been registered, which can be explained by internal bookings or errors. Since AmountAfterTax will be an important feature further, bookings that do not have this information are considered useless. Another change made in this feature was the cast to all the values to the float accordingly and divided by the length of stay. With this change, the feature now represents

the amount paid by night, and not for the entire stay, improving the usefulness of the feature.

Finally, it will be necessary to work on the target variable in order to provide better analysis in the next phase. Originally, the ReservationStatus feature has three categories represented by the integers 6, 7 and 20. The integers 6 and 20 represent canceled bookings and non-canceled bookings, respectively, while 7 represents no-shows. For the purposes of this model, canceled reservations and no-shows will be treated equally, so bookings with a ReservationStatus of 6 will be mapped as "Not-Canceled", while canceled bookings and no-shows will be mapped as "Canceled".

All this data cleaning has left the dataset with a total of 31018 samples.

3.3 Data Understanding

Data understanding, or in some cases exploratory data analysis, is another typical phase in every machine learning model, especially when dealing with predictive analysis over a tabular dataset. This phase is very important to understand the nature of the data and understand how each feature can impact the target, by analysing each one using statistical metrics or plotting them directly. With this visualization, relationships between features can be also studied and insights for feature engineering phase can be established during EDA.

Starting with target feature distribution, figure 3.3 shows that the dataset is highly imbalanced, with just 9.51% of the samples being targeted as canceled bookings. This is a very important analysis since this imbalance can impact the training phase, as referenced in section 2.4.

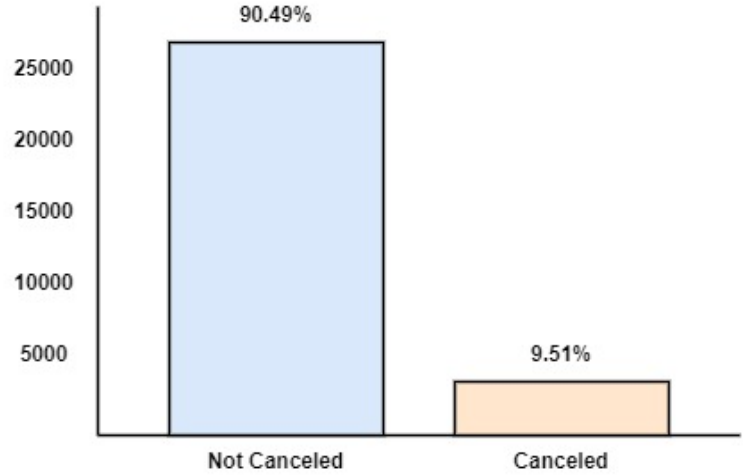

Figure 3.3: Target feature distribution, ReservationStatus

In figure 3.4, the evolution of booking count and cancellation ratio by week can be monitored. The peek in booking count, observed in the first thirteen weeks, is explained by the datetime window used to collect the data, referenced in section 3.1. As this slightly

changes the actual distribution of bookings during the year, the data collected in the first thirteen weeks of 2023 could be removed, but only if it is really necessary as it will not have a direct impact on the cancellation ratio. Looking at the cancellation ratio, it is interesting to note that the highest peaks of cancellations per week are found in the first and last 8 weeks of the year, representing the months of January, February, November and December, with canceled bookings reaching 22% of total bookings. The lowest cancellation peaks are between weeks 18 and 22 and between weeks 30 and 36, representing the month of May and the end of July to the beginning of October.

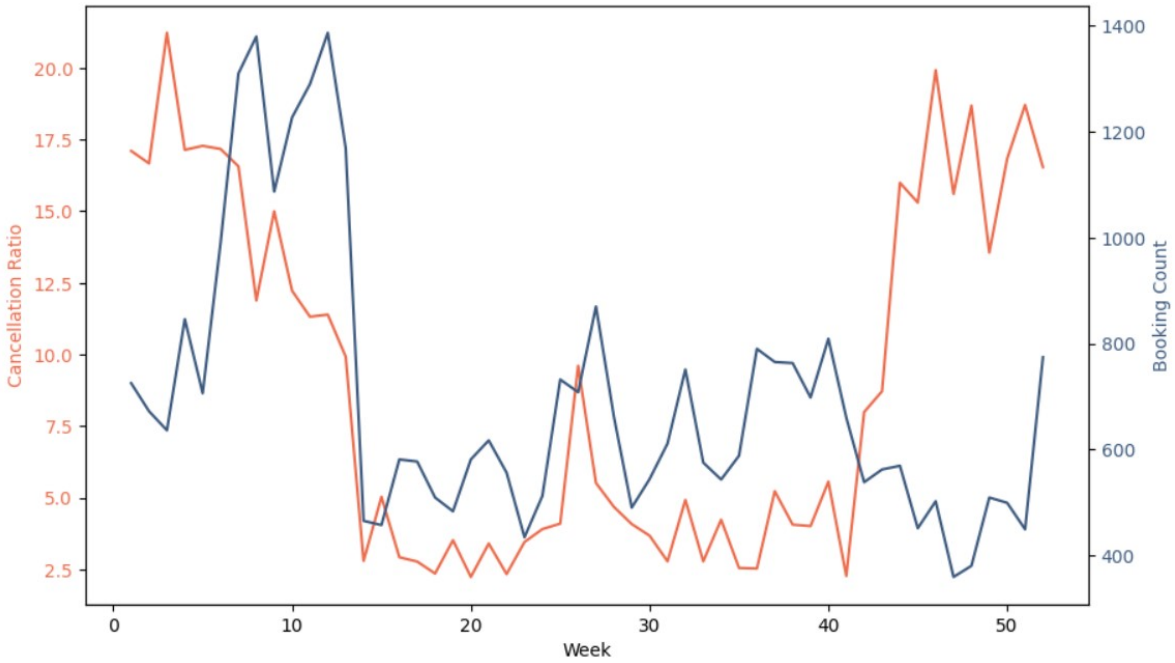

Figure 3.4: Cancellation Ratio and Booking Count by Week

Since the data was collected from Portuguese hotels, is expected that Portuguese nationality dominates the dataset. More precisely, 29% of the bookings comes from Portugal, followed by United States with 13.3%, and United Kingdom with 9%. The others 49.7% are distributed among the rest of the nationalities with France, Brazil, Deutschland, Spain, Italy, Canada, and Netherlands being the big group. Looking to figure 3.5, which illustrates the cancellation ratio by nationality, it is noticeable that there aren't relevant differences between the cancellation behavior for the nationalities with more presence in the dataset. Some high cancellation ratios can be observed mainly in nationalities from Africa and Asia due to the few bookings registered from there. Myanmar, for example, has just two registered bookings in the entire data, both canceled, leading to a cancellation ratio of 100%. Costa Rica, Tanzania, Reunion and Zimbabwe have cancellation ratios between 50% and 67% but there are only fifteen records of this nationalities combined. The only nationality with a bigger representation in the data, which has a higher ratio than normal is Portuguese, with values of 17%.

For all these points mentioned, Nationality feature seems not to be a feature with a good predictive value and could be discarded in the feature selection phase. If it is used, it will have to go through feature engineering processes, also because it is a categorical variable with high cardinality. United States and United Kingdom have cancellation rates lower than average, with values of 4.43% and 7.0%, respectively.

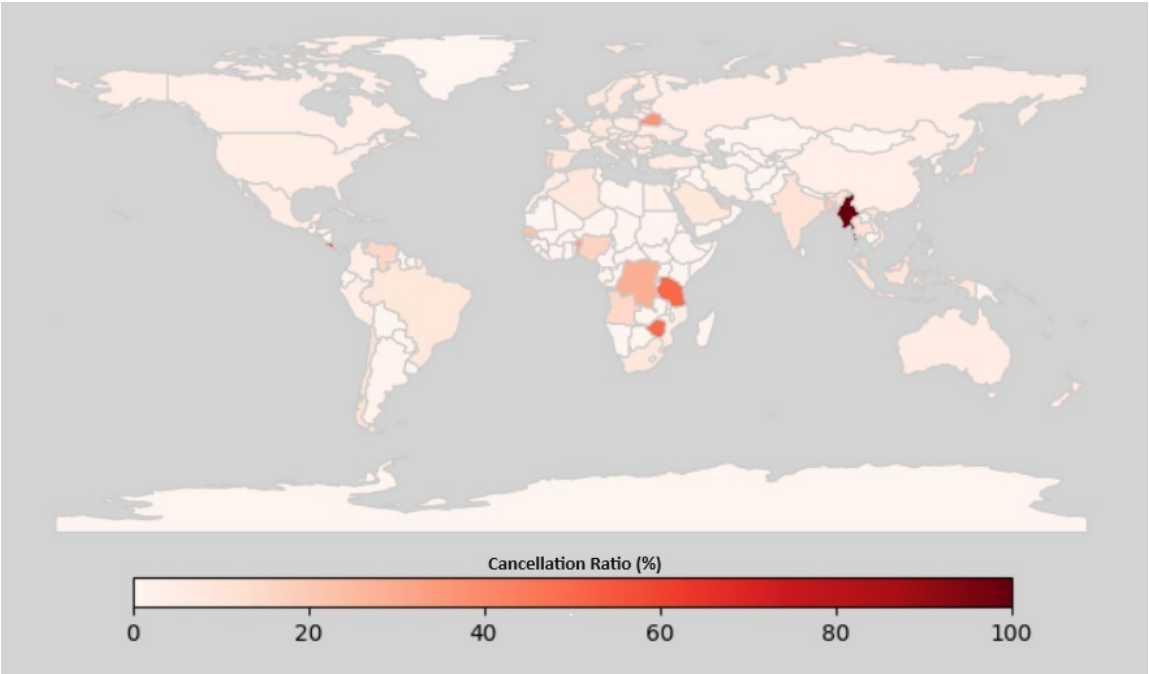

Figure 3.5: Cancellation Ratio by Nationality

The authors in [43] stated that the source channel through which the booking was made could be one of the reasons that most influence the cancellation behaviour. Looking to the distribution of Source feature, plotted in figure 3.6, the dominance of bookings with the

source unknown or from Availpro is clear, with 83% of the entire dataset being distributed over these two categories. Analyzing the cancellation ratio differences between Source categories, Availpro is the only category with a cancellation ratio higher than normal, with a value of 17,26%. Others category, have a significantly low cancellation ratio, with just 2.89% of the bookings being canceled. When the Source is unknown, the cancellation ratio is 8.86% which is a value close the 9.49% of the entire dataset. Slight differences in the distribution of the target feature can be observed between the categories which indicates that Source could have some prediction value. Despite this, the feature does not seem to have a strong impact on the dataset on its own and may need other features to increase its predictive value.

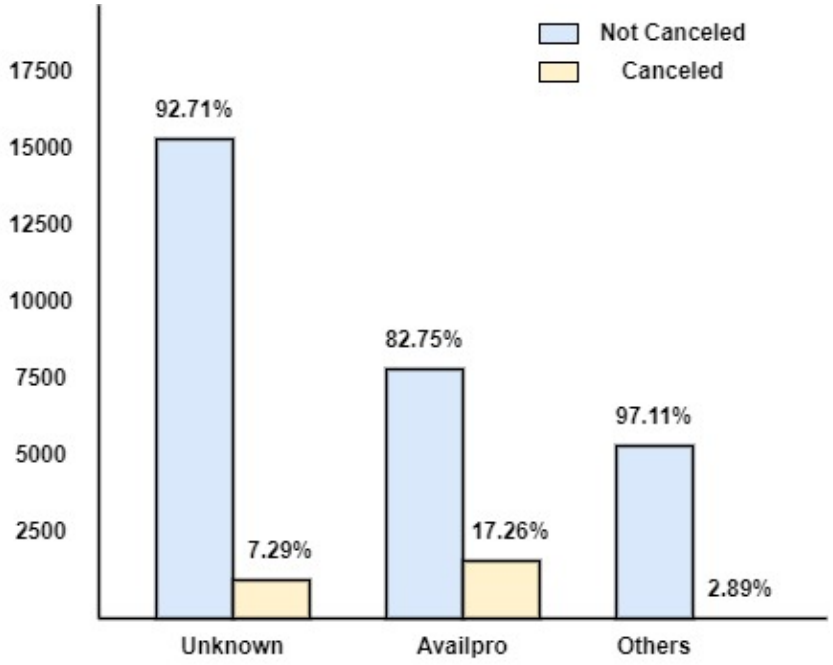

Figure 3.6: Cancellation Ratio by Source.

3.4 Data Splitting

In data splitting phase, is where the dataset is normally divided into two main sets: the training set and the test set.

The training set is the one which the model uses to learn the patterns and relationships of the data, in order to predict the output. Since the training phase requires the most data, normally 60% to 80% of the data is addressed to this set. The test set is used to make the final evaluations and conclusions about the model's performance, with the rest 20% to 40% of the data. It is important that this set keeps the structure of the training set and be representative of the data to which the model will be exposed, allowing to understand the real predictive performance of the model.

The main reason to perform data splitting before feature engineering phase is to avoid

data leakage during the model's training. Data leakage happens when the model is trained with information that can potentially leak the target output. Creating new features, scaling data, or making feature selection over the entire dataset (before splitting it) can lead to what is called train-test contamination, making the model well-fitted to the used sets but will probably perform poorly in unseen data.

During the training phase it is important to have an independent set where the model can be tested with different hyperparameters combinations, in order to choose the combination that best fits the data. This process is called cross-validation and is made over the validation set, that normally contains 10% to 20% of the training data and. This split will be discussed further on in this chapter.

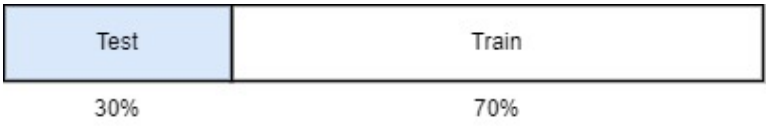

Figure 3.7: Data splitting

The splitting ratio applied in the dataset was 70% for the training set and 30% for test set, as shown in figure 3.7.

3.5 Feature Engineering and Selection

3.5.1 KMeans Clustering for Room Type

Since the dataset contains data from three different hotels, with different rooms, prices and locations, it is important to be careful when mixing information that depends on the hotel. Looking to the booking's price (AmountAfterTax), for example, what is considered an expensive booking to Hotel 1, can be normal for Hotel 2. These kinds of inconsistencies in booking behavior between hotels cannot be modeled as if they didn't exist and so they must be normalized.

To normalize the rooms between hotels, a new dataframe was created where each row represents different rooms. Then, four columns were added as characteristics of the room. The first one is the mean room's price, using AmountAfterTax feature. Since this feature used it variables as strings, all the values were processed to the float accordingly. Then, the mean AmountAfterTax value was calculated and divided by the mean price of the hotel which it belongs and added in the corresponding room's row. The second feature of the new dataframe is Count, representing the number of bookings registered in each room divided by total hotel's bookings. The last two features are Avg_LOS and Avg_Guests, representing the mean of los and TotalGuests for each room.

After getting the dataframe ready, the idea was to implement a clustering algorithm in the data to create clusters between the different rooms. To choose the right number of clusters that best fit the data, the K-Elbow method was used and can be observed in figure 3.8. Looking to the graphic, an "elbow" curve can be observed when k (number of

clusters) is 3 with a distortion score of 16.263, representing the optimal distortion score by k clusters.

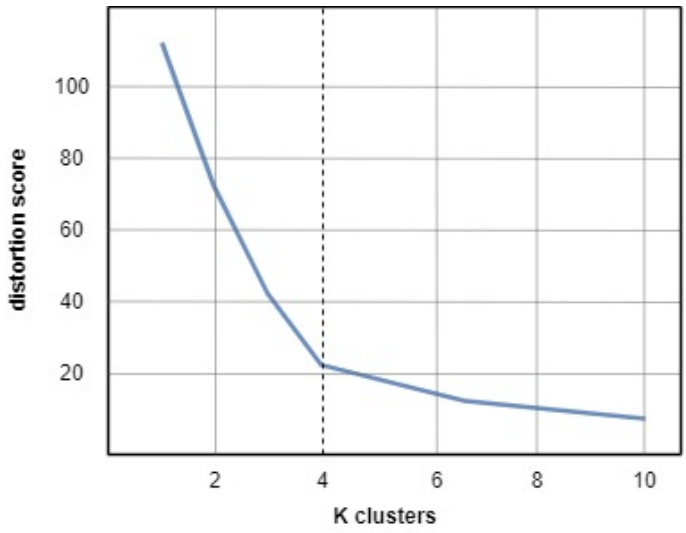

Figure 3.8: Choosing the optimal number of clusters with K-Elbow method

Then, KMeans algorithm was applied (with n_clusters parameter configured with a value of 3) and the clusters were formed. Looking to the statistical stats, AmountAfterTax and Count features highly impacted the clusters creation, while Avg_LOS and Avg_Guests did not show major differences between clusters. Since the data has two features that can clearly separate it in three clusters, a two-dimensional plot can be made to observe the clusters. Analyzing figure 3.9, which plots the clusters over AmountAfterTax (x-axis) and Count (y-axis), some conclusions can be made: cluster 0 represents the most expensive rooms, cluster 1 the low/medium price rooms with higher demand and cluster 2 the low/medium price rooms with lower demand.

Finally, in the original dataset, the values of feature ChannelRoomTypeCode were mapped with the corresponding cluster and the feature name were also replaced by RoomCluster.

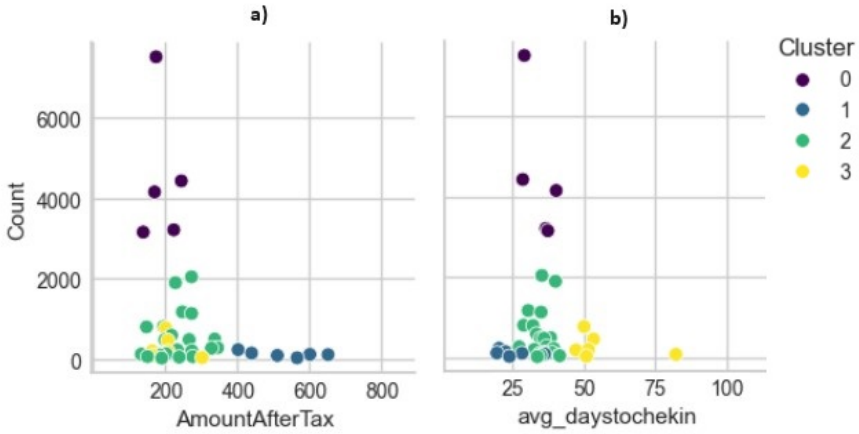

Figure 3.9: Clusters representation

3.5.2 Encoding Categorical Features

Feature encoding is the process of representing categorical features in numerical values or in a format that can be understood by machine learning algorithms. In recent years, some algorithms and architectures (like Catboost algorithm or deep learning architectures referenced in 2.3.2.2) can already process categorical features in their raw form, but most of them still need a pre-processing of non-numerical features. Usually, there are two types of categorical data: ordinal data and nominal data. Ordinal categories have an inherent order between them, such as the "GroupType", and normally they are encoded into a ranking to highlight that order. Nominal features, like the features "HotelId" or "Nationality", are the opposite, they do not assume any type of order and in this case, it is very important to choose the right encoding method to not impose a false perception of ranking. For the dataset used in this study, three methods of encoding will be considered: label encoding, mean target encoding and one-hot encoding.

Label encoding is the simplest form of encoding where each category is converted in an unique integer value, and a ranking can be created between them. For this reason, label encoding should be only used with ordinal features to prevent a misinterpreted ranking if the feature does not have one.

Mean target encoding is a technique where the frequency of the feature classes inside the positive target class is calculated and used to encode it. Using table c) of figure 3.10 as example, HotelId feature has three classes. For class Hotel_1, there are just one sample and is target as canceled, so the value that is used to encode Hotel_1 is 1.0. In opposite, Hotel_3 has also just one sample, but in this case targeted as not canceled, so the value 0.0 is used to encode it. Finally, Hotel_2 has two samples, one not canceled and one canceled. In this case, half of the samples of Hotel_2 are canceled, so the value that is used to encode it is 0.5. This encoding is very useful because the way the method is used creates an explicit relation and value between the class encoded and the target, making it perform better than the other techniques. Despite this, if two classes have the same effect in the target, they will be encoded with the same value, causing a loss of information between these classes and other features. Another advantage of this method is the considerable cardinality decrease.

One-hot encoding is a method where n new binary columns are created, representing the n cardinality levels existing in the feature. Besides this method being widely used to build machine learning models, specially because preserves all the information of the original feature, it can introduce a big problem when used over high dimensional data, normally mentioned as dimensionality curse. If it used over a feature with high cardinality, the model complexity will increase significantly, being more propitious to suffer of overfitting.

At this stage, the dataset contains nine categorical features that need to be encoded. As the variables SegmentCode, SubSegmentCode, ChannelRoomTypeCode, Source and RoomType show differences in the effect of their classes on the target, or a high cardinality

Sample	HotelId	ReservationStatus
1	Hotel_1	Canceled
2	Hotel_2	Not Canceled
3	Hotel_3	Not Canceled
4	Hotel_2	Canceled

a) Original Set

Sample	HotelId	ReservationStatus
1	1	Canceled
2	2	Not Canceled
3	3	Not Canceled
4	2	Canceled

b) Label Encoding

Sample	HotelId	ReservationStatus
1	1.0	Canceled
2	0.5	Not Canceled
3	0.5	Not Canceled
4	0.0	Canceled

c) Mean Target Encoding

Sample	Hotel_1	Hotel_2	Hotel_3	ReservationStatus
1	1	0	0	Canceled
2	0	1	0	Not Canceled
3	0	0	1	Not Canceled
4	0	1	0	Canceled

d) One-Hot Encoding

Figure 3.10: Representation of different encoding techniques.

as in the case of ChannelRoomTypeCode, the method used for their encoding was mean target encoding. On the other hand, as the HotelId and GroupType variables have a low cardinality and similarities in the frequency of their classes in the cancellation, the method chosen for their encoding was one-hot-encoding. Finally, as the DistChannel variable only contains two classes and the classes in the GroupType variable have a rank relationship with each other, label encoding was preferred for these cases.

3.5.3 Feature Selection

Before feeding the machine learning algorithm with the training set, it is important to test some feature selection techniques to understand if every feature is really useful and brings predictive value. Using redundant features in the predictive model, beyond the increase of computational costs and time consumption, can reduce the model's generalization and impact it performance.

The most common feature correlation technique, and the one that will be used in this

study, is the calculation of Pearson's Correlation Coefficient. Pearson's Correlation is an indicator of the linear relationship between two variables, that varies between -1 and 1. A correlation value greater than zero is a positive relationship, which indicate that both variables grow in the same direction, i.e. when one goes up, the other also goes up. A correlation value less than zero indicates a negative relationship or an inverse correlation, which means that the variables grow in opposite directions. Values close to -1 and 1 are considered strong relationships, meanwhile values close to 0 indicates that there is no relationship at all.

With X and Y being the variables in comparison, and n the number of total data points present in the training set, the Pearson's correlation coefficient is given by:

$$r = \frac{n\sum_{i=1}^{n}(X_i \cdot Y_i) - \sum_{i=1}^{n} X_i \cdot \sum_{i=1}^{n} Y_i}{\sqrt{\left[n\sum_{i=1}^{n}(X_i^2) - \left(\sum_{i=1}^{n} X_i\right)^2\right] \cdot \left[n\sum_{i=1}^{n}(Y_i^2) - \left(\sum_{i=1}^{n} Y_i\right)^2\right]}} \tag{3.4}$$

Looking to table 3.5, which shows the top six features with the highest Pearson correlation coefficients, it is clear that only the ChannelRoomTypeCode, SubSegmentCode, Source, SegmentCode and DistChannelCode variables have any significant values. The first thing that stands out is that four of these five variables were encoded using mean target encoding. As mentioned in the previous section, this encoding creates an explicit relationship with the target, making the variables encoded using this method outstanding when calculating the coefficient. In turn, the large number of discrete variables present in the dataset makes it harder to apply the model, since it assumes that the data is continuous and linear. As a result, although this method is one of the most widely used in the feature selection phase, it does not seem to be the most suitable for the structure of this dataset.

Table 3.5: Percentage of Missing Values by Feature.

	Coefficient
ChannelRoomTypeCode	0.52
SubsegmentCode	0.2
Source	0.18
SegmentCode	0.13
DistChannelCode	0.11
Adults	0.053

To overcome the difficulties faced in Pearson's coefficient calculation, other methods can be used to try to understand the impact of each feature on the target. In this study, the alternative applied will be to use the methods for calculating the feature importance score provided by some algorithms based on decision trees, such as Random Forest or XGBoost. This method is based on training one of these models with default parameters and using its capabilities to identify the variables that were most important in its training. As well as analyzing the direct impact that the variables had on the model, this method is applicable to any type of variable, making it easier to use and the results more reliable.

A Random Forest model was then trained and the results of the feature importances can be seen in figure 3.11. As expected, the results obtained by applying this method differ significantly from those obtained by calculating the Pearson correlation. The only variable that remained in the top six was ChannelRoomTypeCode, which proved to have a major impact on training the model, with a value of 0.29. The rest of the top six is occupied by the variables daystochekin, checkin_date_week_number, AmountAfterTax, the dummy variable _Hotel2 and los, with values decreasing from 0.13 to 0.6. Despite these differences, the results showed again that the variables do not have a major impact on target prediction on their own, excluding of course, ChannelRoomTypeCode.

The results of this method are much more reliable given the context of the problem, not only for the reasons mentioned above, but also because the models that were trained in the modeling phase also have decision trees as a weak learner.

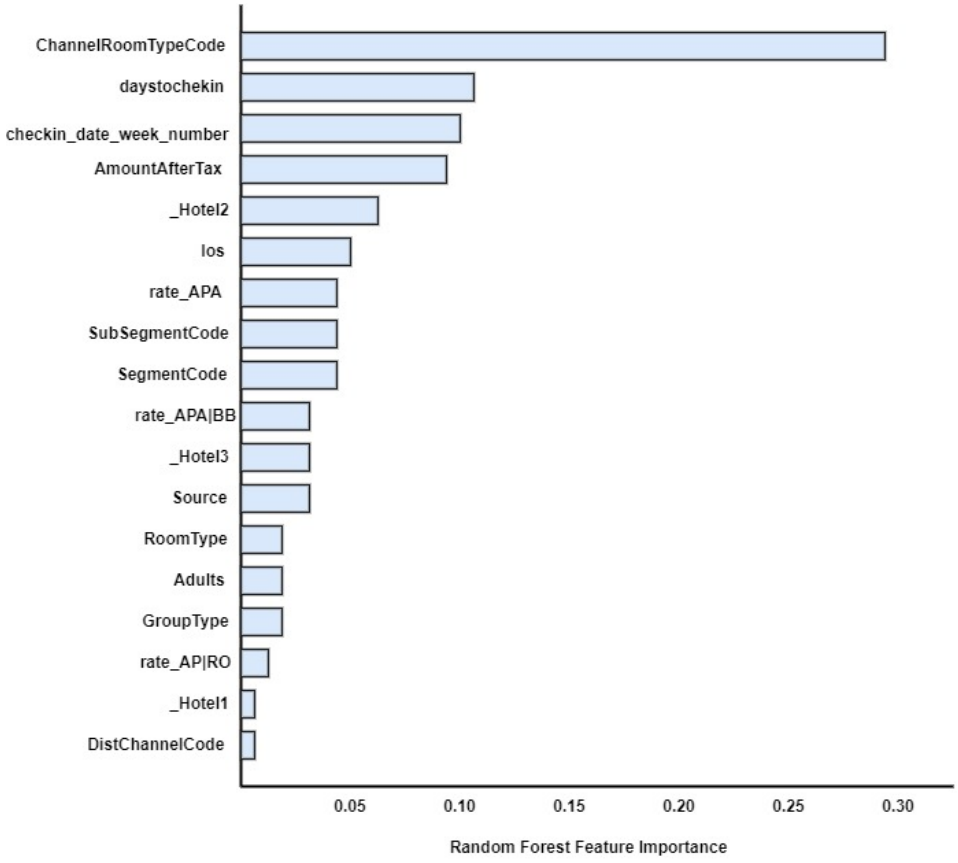

Figure 3.11: Results of the feature importances calculated by the default model of Random Forest

3.6 Addressing Class Imbalance

The distribution of the target variable ReservationStatus, shown in figure 3.3, highlighted the high level of class imbalance present in the dataset, with 90.49% of the samples being from negative class (not canceled) and 9.51% from positive class (canceled). Class imbalance is a topic that needs attention because it can lead to misleading results, if not carefully handled. Currently, a lot of techniques to handle class imbalance have been studied, some of them exposed in section 2.4, and the biggest conclusion is that there is no perfect formula, different techniques should be tested in order to understand which ones fits the problem better.

In addition to the various metrics that will be used to evaluate the models in chapter 4, a second training set, with synthetic data, will be created from the original using the SMOTE-ENN technique, since authors in [41] got better results with the training set generated from it. Subsequently, the results of the models obtained from this new training set will be compared with the results of the models obtained from the original one.

SMOTE-ENN is the combination of two techniques used to handle imbalanced datasets, SMOTE and ENN. SMOTE, was already explained in section 2.4.2 and is, in short, a technique that uses the KNN algorithm to oversample the minority class. In turn, Edited Nearest Neighbors (ENN) is used to identify and eliminate noisy samples. Also using the same algorithm, this technique assesses whether the target of the sample in question is different from most of the targets of its neighbors and, if so, considers it to be a noisy sample and removes it. Figure 3.12 shows how this method works, as well as the influence of each of the techniques that compose it.

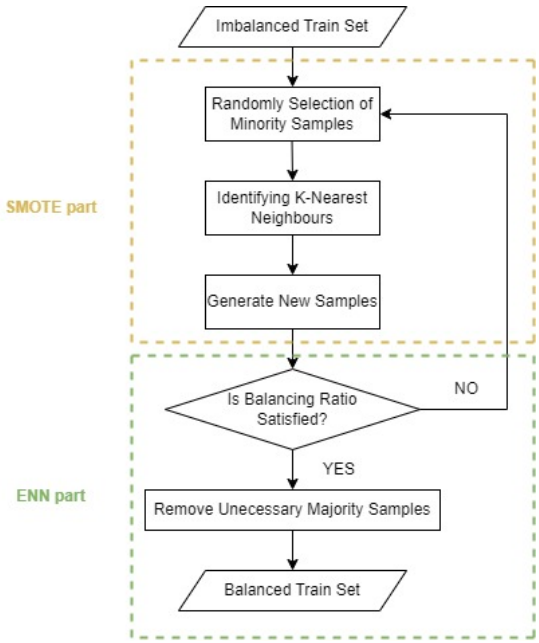

Figure 3.12: Flow diagram of SMOTE-ENN.

3.7 Modelling

Modelling is normally the most interesting phase in every predictive model building. After getting the dataset split, cleaned and prepared, the training set is exposed to various algorithms in order to let them understand the problem and identify useful patterns for the target prediction. This process is usually claimed as training phase, but before it is important to understand which algorithms fits the problem better. There are dozens of ML algorithms and DL architectures that can be used in binary classification, some of them exposed in sections 2.3.1.4 and 2.3.2.2, but training all of them would have computational costs. With this, the modelling phase is structured in two principal processes: models selection and the training of them.

Since this study deals with non-linearity data, where to low correlation exists between the features by their own and the target variable, it requires algorithms that can lead with complex problems, and the ones that are likely to achieve the best results are those that use ensemble and boosting techniques in their architecture. For these reasons, and looking to section 3.5.3, the ML based algorithms that seem to meet the needs imposed by the problem are Random Forrest, LightGBM, AdaBoost, XGBoost and CatBoost, and are the ones that were trained and tested. For the DL models, Wide&Deep, DeepFM, AutoInt and TabNet were trained, since are the ones available in deeptables API.

After selecting the models to be used, the training phase begins with hyperparameter tuning and the cross-validation technique that are essential components during training phase to ensure the robustness and generalization of the models.

The hyperparameters are external settings of each algorithm that are configured before the training and tested in different combinations to identify the one that gets a better performance [49]. Depending on the data and the problem, the best combination can vary significantly, which means that the combination that fits one dataset, should not be used in another without being tested. Each combination is trained and tested in different sets of the data, and this process is made in different iterations to change the train and validation folds, as shown in figure 3.13.

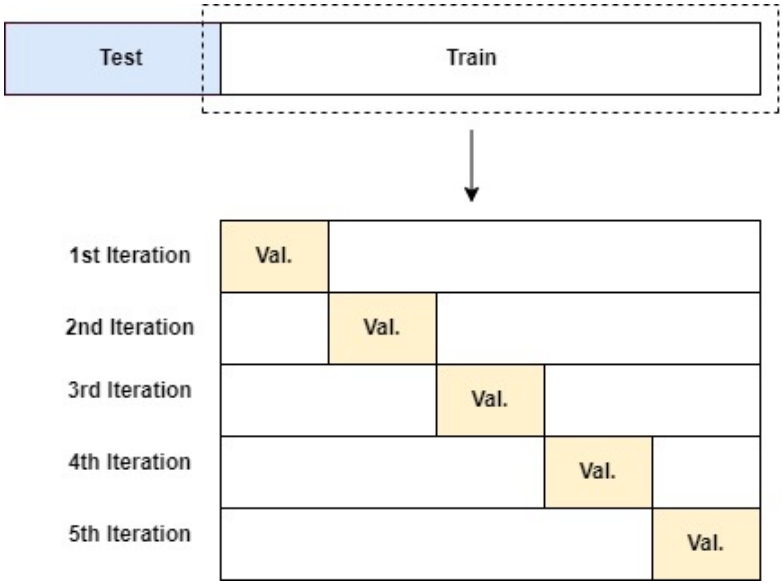

Figure 3.13: Cross validation representation.

In this research the tuning method used was GridSearchCV by sklearn API because of it exhaustive search over all values configured in param_grid parameter. In contrast to RandomizedSearchCV, by sklearn as well, every single combination is tested which increases the computational costs but provides more complete and reliable results. This method has a cross-validation process implemented, and the number of different folds to be tested can be configured in cv parameter. For this study, the value chosen for cv was 5, which means that the data was divided in five folds (four for training and one for validation) in five different iterations for each combination. The different hyperparameters combinations for each model can be observed in appendix A.

4

Results and Discussion

After the development and implementation of all models, this chapter reports and analysis the results and behavior of each one and a more depth analysis of the CatBoost model will be made in order to understand the business value of using it in a production environment. Finally, the results of the models trained over the resampled train set will be compared and analysed as well.

4.1 Comparison Between ML And DL Results

As mentioned in section 3.7, five different ML models and four DL architectures were trained over a fivefold cross validation to guarantee that the best hyperparameters combination for each one were used. After the training, the same test set was applied to all the models to analyze and compare the performance metrics between them. Taking into account the nature of the problem, specially because of the high imbalance verified in the target feature, the metrics considered were accuracy, precision, recall and AUC.

Table 4.1 shows the prediction results of all models in the different sets.

Table 4.1: Results

	Train				**Test**				**Val.**
	Acc.	**Pre.**	**Rec.**	**AUC**	**Acc.**	**Pre.**	**Rec.**	**AUC**	**AUC + std**
Random Forest	0.951	0.987	0.488	0.744	0.950	0.980	0.482	0.740	0.919 ±0.007
XGBoost	0.959	0.968	0.586	0.792	0.952	0.925	0.543	0.769	0.915 ±0.009
Adaboost	0.996	0.999	0.960	0.980	0.946	0.770	0.609	0.795	0.874 ±0.006
LightGBM	0.957	0.959	0.575	0.786	0.952	0.921	0.541	0.768	0.914 ±0.010
CatBoost	0.961	0.940	0.620	0.808	0.952	0.874	0.589	0.790	0.776 ±0.008
Wide&Deep	0.950	0.974	0.478	0.738	0.950	0.972	0.500	0.750	-
DeepFM	0.950	0.956	0.494	0.746	0.950	0.943	0.516	0.756	-
AutoInt	0.943	0.929	0.429	0.713	0.944	0.912	0.460	0.728	-
TabNet	0.952	0.987	0.500	0.750	0.951	0.973	0.498	0.748	0.894 ±0.007

Assuming that the baseline accuracy score is 0.905, which represents the accuracy score if all bookings from test set were predicted as non-canceled, all the models have added value in predicting cancellations, with all the accuracy values showing significant increases, ranging from 0.944 to 0.952, with LightGBM and CatBoost getting the higher accuracy score.

For precision and recall, it's easy to observe that precision got better results across all the models, which means that most of the bookings identified by the model as canceled, are actually canceled bookings. Adaboost were the only model with less convincing results for precision with a value of 0.770. All the other models achieved results ranging from 0.874 to 0.980, with Random Forest achieving the best score. The recall results show that all models were just able to identify half of the cancellations, with values ranging from 0.460 to 0.609.

For AUC, the models showed close and satisfactory results, which means that the models are managing to find relationships that separate a canceled booking from a non-canceled one, with CatBoost and Adaboost getting the best results with 0.796 and 0.790. The weakest results appear in the DL section, with 0.728 for AutoInt and 0.748 for TabNet. Besides Adaboost achieved the best result, an high variance between the train and test metrics of can be observed, which could imply the existence of overfitting.

The discrepancies observed in the validation results, which were the results obtained by cross-validation of GridSearch, suggest that the hyperparameters tuned during the cross-validation process might be overly optimized for the particular folds used. Besides that, the low values of standard deviation and the similar results between train and test sets, reflect the solidity and consistency of the models, which is a good indicator of the models performance. For Wide&Deep, DeepFM and AutoInt models, no validation values are shown as their implementation via deeptables API does not support an integration with GridSearch.

Looking to the learning curves of ML models, shown in image 4.1, the overfitting in AdaBoost can be proved. In this type of curve, the training and the cross-validation scores are supposed to converge to each other, reaching a point of stability. This behavior is not observed in Adaboost's learning curve, where the training score is always close to 1, which is a clearly evidence of overfitting. All the other curves show the supposed behavior, specially Random Forest that almost reached a perfect fit between both curves. This fit matches the results obtained in table 4.1, where Random Forest test results are very similar to the training ones, showing that the model generalized well to unseen data. All the other models didn't reach a perfect convergence, proving the small differences shown between the metrics of both sets, but their differences aren't too significant.

Another point that can be observed by analyzing the learning curves, is that the stability that is being reached in the models means that all of them are close to their training potential with the available data, indicating the need to obtain new features so that higher prediction values can be achieved.

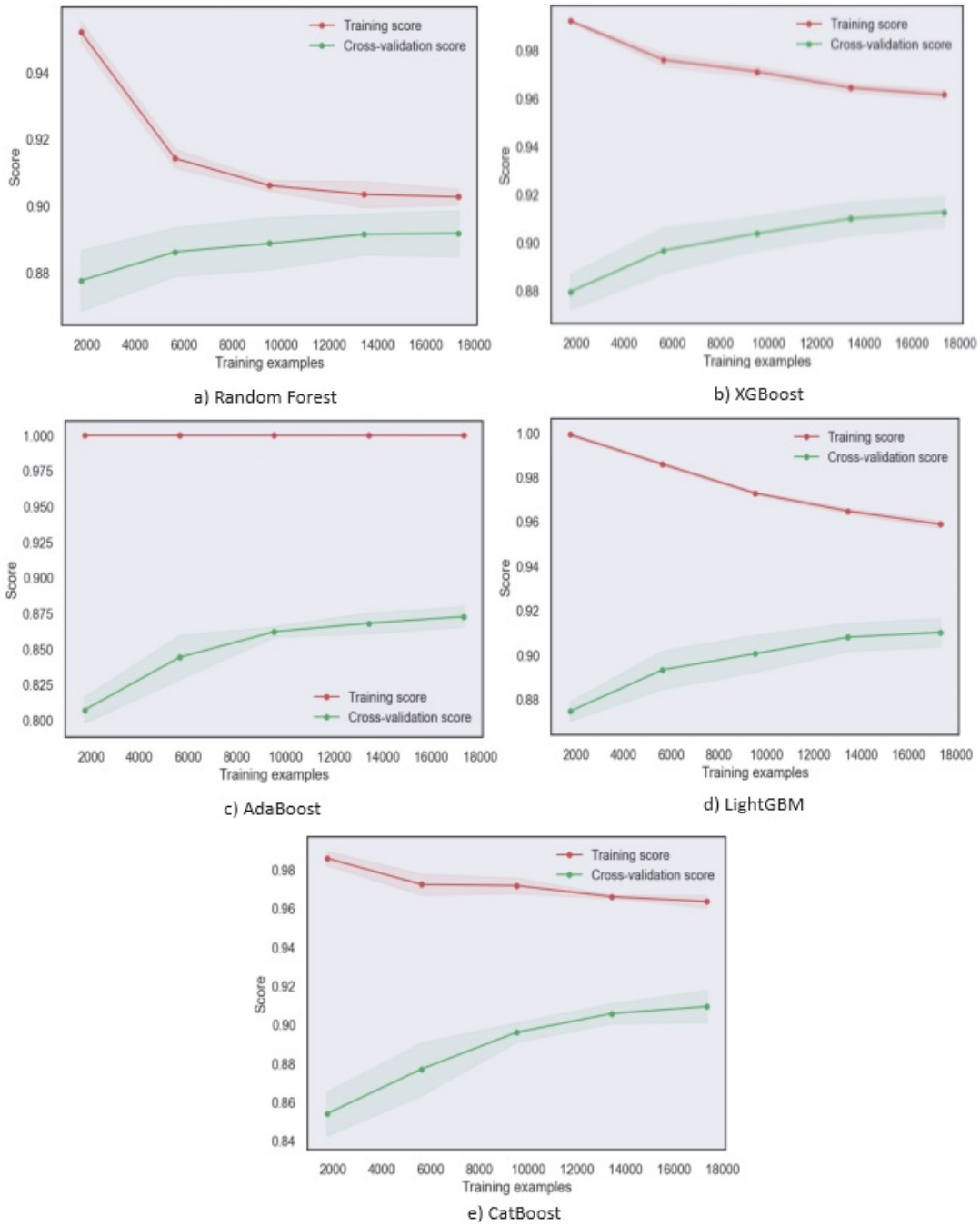

Figure 4.1: Machine learning models learning curves.

For the DL models, the interpretation of learning curves are slightly different, using different metrics for x and y axis. In x-axis, instead of using the number of data in the training set, the metric used is the number of epochs which are a complete pass through the entire training set. For the y-axis, the metric used is the validation loss, that represents loss value computed by the chosen loss function during training and validation. The ModelConfig class of deeptables API automatically uses binary cross-entropy as the loss function when working in a binary classification problem.

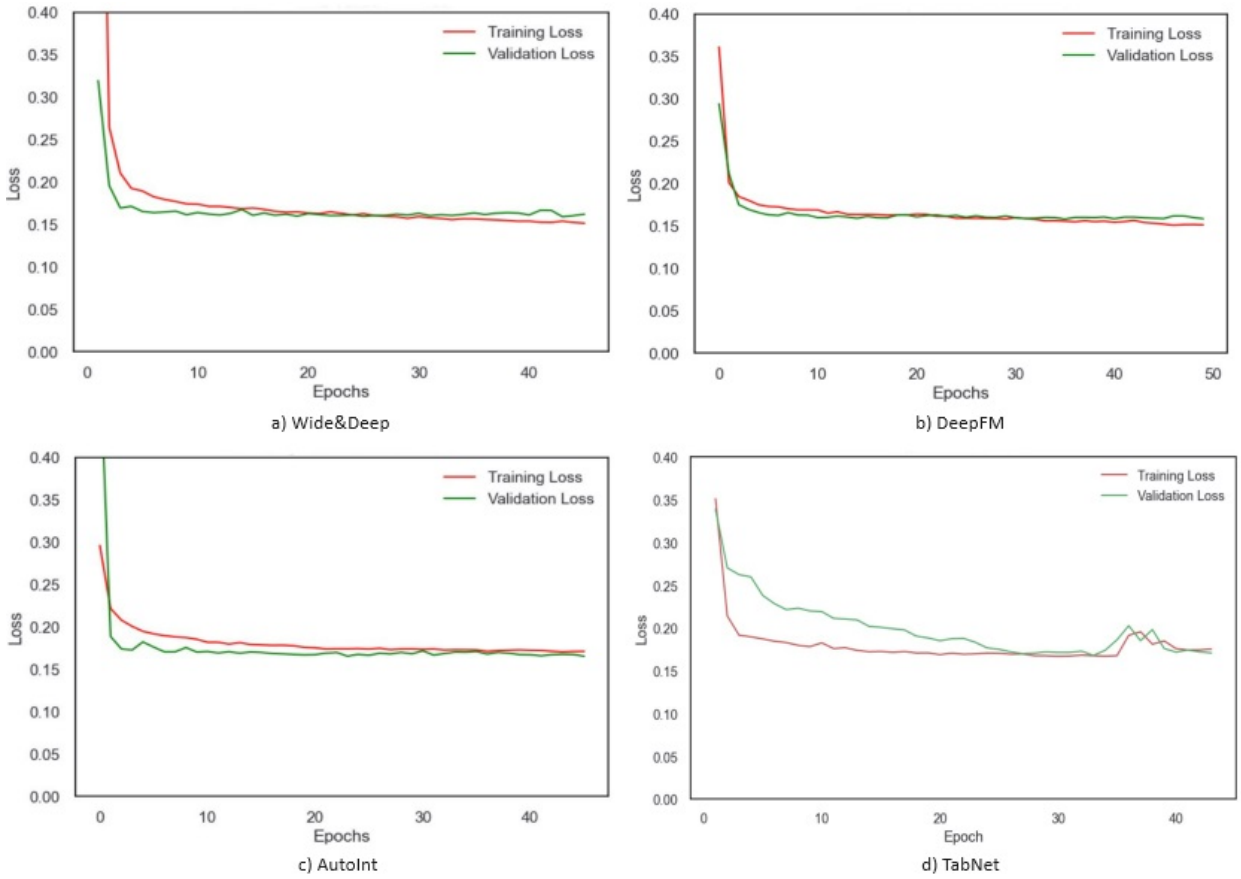

Figure 4.2: Deep models loss curves.

The visualization of loss curves in figure 4.2, shows that the curves of both sets converge to each other for all the models, reaching to a stability point where the gap between them were minimal. This behavior demonstrates the good fit of the models to the dataset without signs of overfitting, meaning that the performance of the unseen data is similar to the one observed during the training process. This point matches the results of table 4.1 for the DL side, where the performance metrics of both sets are also similar.

Comparing the four graphics, one of the differences that can be observed is that in the models built with the deeptables API, Wide&Deep, DeepFM and AutoInt, the validation loss is lower than the training one before the converge of both. One reason for this can be the default regularization that is configured in the ModelConfig class of the API that is just applied during the training process, sacrificing the training loss in order to improve the validation/test loss. This theory can be proved by the better results achieved by the test set prediction in these three models. Another interesting point that can be analysed is the number of epochs needed by the models to reach an almost perfect converge of the curves. Deeptables API models, specially Wide&Deep and DeepFM, reached this fit in less epochs, between the 10th and 20th epochs, while TabNet needed almost thirty epochs to reach a converge stability, demonstrating more difficulties to fit the data.

It's important to note that the use of fifty epochs were just for visualization reasons. The models were all trained with an early stopping system of ten epochs, meaning that the training process was interrupted if no improvements in the loss value were visible after ten epochs. This type of optimization is used to prevent overfitting scenarios as can be seen in the last epochs of Wide&Deep and DeepFM where the model is too trained over the training set and the validation loss starts to increase.

Comparing the metrics results of both methodologies, the superiority of ML models is not so clarifying, with some DL models coming close to the results obtained by the most popular models or even surpassing them, as in the case of Random Forest which had lower results than Wide&Deep, DeepFM and TabNet but in another aspects, such as complexity, the use of ML models can be very profitable. Table 4.2 shows the computational time needed by each model to be trained over the training set and to predict the test set and the differences are evident, specially in training times. All DL models needed several seconds to be trained, with AutoInt and TabNet breaking the barrier of 50 and 80 seconds, while some ML models didn't even need a second to be trained, such as LightGBM, XGBoost or Random Forest. In a conceptual context, these differences may not be important, but it is clear the impact that these differences could have if these models were deployed into a production environment, where they will be trained and called upon constantly.

Model	Training Time (s)	Prediction Time (s)
LightGBM	0.074	0.008
XGBoost	0.130	0.003
Random Forest	0.899	0.067
AdaBoost	3.776	0.072
CatBoost	19.526	0.033
DeepFM	20.716	0.200
Wide&Deep	35.775	0.200
AutoInt	56.159	0.489
TabNet	87.981	0.202

Table 4.2: Times statistics.

4.2 Analysis of XGBoost model in more depth

Before taking a predictive model into production, it is important to make sure that the model chosen meets the requirements of the problem and that, above all, its results make sense from a business perspective. To do this, a more in-depth and detailed analysis of the selected model need to be made, to get better and more complete conclusions about the model behaviour.

Taking into account the study of the models made in the last section, and considering all the parameters evaluated, XGBoost and LightGBM seem to be the most complete and valuable models, not only in terms of their predictive performance, but also in terms of

their speed of training and prediction. With this, XGBoost was the model selected and analysed, using it confusion matrix, to get a better understanding how the model was predicting the test set and to see if it predictions meet the expectations.

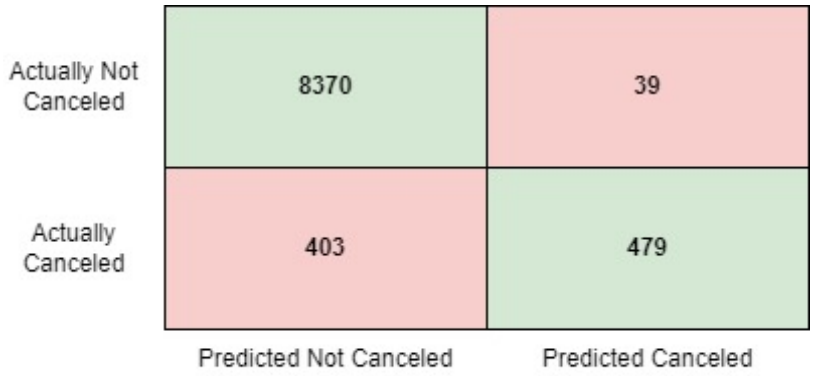

Figure 4.3: XGBoost confusion matrix.

Looking to figure 4.3, that plots the predictions made by XGBoost model in the test set, the results of the model in table 4.1 can be visualized in a way that is easier to interpret. There are two principal observations that can be made with the confusion matrix, the number of false positives and the number of false negatives.

Starting with false positives, the confusion matrix shows that only 39 out of 518 were wrongly predicted as canceled, following the value of 0.925 in precision metric. This is a very good result, specially in business terms, meaning that the model has 92.5% confidence when it identifies a cancellation. In other hand, the model did not identify 403 out of 882 cancellations, following the 0.543 result of recall. This means that the model has some difficulties identifying certain types of cancellations, with only half of them being canceled, and this is exactly the point where room for improvements is found. Despite the high number of false negatives, given the nature of the problem, it is more important to have a low number of false positives since a booking wrongly identified as canceled could have higher costs than an unidentified cancellation.

There are several reasons why the model may be having difficulty identifying a higher number of cancellations, and their origins may lie right at the root, in the dataset used to train the models. The analysis made in the last section, over the graphics obtained in 4.1, showed that the models are reaching their potential so more valuable features need to be collected to get better results. Features like the cancellation policy or the changes that are made in the booking showed to be useful in the studies exposed in chapter 2.7 but unfortunately this dataset does not contain that information. The class imbalance is another factor that may be making it difficult to identify cancellations, as the training set may not have enough cancellation samples for all the relationships to be understood by the model. Another factor has to do with the fact that certain cancellations cannot be predicted at all. Personal problems, health problems or even flight delays and cancellations are reasons why bookings are canceled, and these events are not taken into account by the

predictive models created in this dissertation.

Finally, another factor that may not help the model may be related not only to the need to find other variables, but also to the uninformative variables being used in the dataset. To understand this point, the gain and split importance of the features in the model can be analysed to realize which features are playing a more important role in model training. Gain importance represents the improvement in accuracy resulting from a split on a particular feature. When a decision tree is built, at each node, the algorithm evaluates different features and selects the one that maximizes the gain. Features with higher gain values are considered more important because they lead to larger improvements in accuracy when making splits in the decision trees. Split importance measures the number of times a feature is used to split the data across all trees in the ensemble. Essentially, it quantifies how often a feature is chosen for splitting.

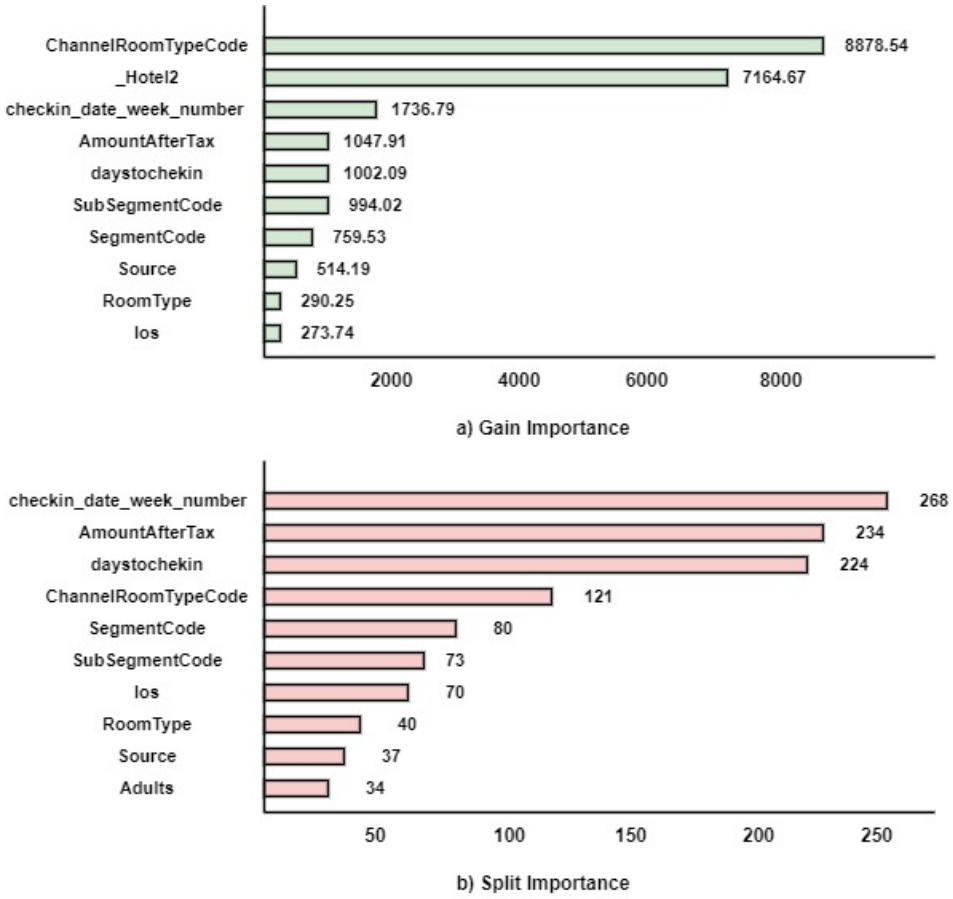

Figure 4.4: XGBoost gain and split importances.

Considering gain and split importances, figure 4.4 shows the impact of each feature in the training of the XGBoost model. The first point to take into account, is that features ChannelRoomTypeCode, checkin_date_week_number, AmountAfterTax and daystochekin appear in the top five of both metrics making it clear how important these features are to the model. In the opposite side, features like DistChannelCode, _Hotel1, rate_APA | BB or rate_APA | RO do not seem to have much impact on the model, showing

practically zero values. These types of features are considered uninformative features, as they do not add any value to the predictions and can even be removed from the model training. A different behavior to the previous ones can be seen in the _Hotel2 variable, which has a very high gain score while being one of the variables with the lowest split score. This type of behavior happens mainly with categorical variables that are encoded using one-hot encoding, such as this case. The fact that the booking belongs to Hotel2 proves to be quite important in classifying it as canceled or not, but it turns out that only 15% of the bookings are from this hotel, making it a minor variable in the splitting of the decision trees.

These types of evaluations are very valuable, not only in technical terms but also in business terms, as they show in detail which variables in the dataset are most important for predicting cancellations or which can be discarded and can even be compared to the conclusions made at the data exploration phase, in section 3.3.

4.3 Resampling Results

As mentioned in section 3.3, the dataset suffers from an high class imbalance with 90.51% being not-canceled bookings and 9.49% being canceled ones. This can cause various problems in the training phase, since the model cannot be able to capture all the relations that can identify a booking as canceled, turning down the recall results as shown in the previous section.

To prevent this, in section 2.4.1, a technique called SMOTE-ENN was described and implemented in order to resample the training set and analyze the results on the test set. As shown in figure 4.5, the original train set consists in 19620 not canceled samples and 2059 canceled, while the resampled train set has 13971 not canceled and 11420 canceled. It is important to note that the resampling is not applied to test set to keep the natural distribution of the target variable in this set and ensure that evaluations are carried out in a context that is close to the real scenario.

Figure 4.5: Target distribution before and after resampling.

After the resampling, an XGBoost model was tuned and fitted to the new training set and the predictions were made over the original test set. The results, in table 4.3, show that besides the increase of ROC-AUC metric from 0.769 to 0.811, the precision decreased drastically from 0.925 to 0.587. This means that 418 out of 1011 samples identified as canceled were wrongly predicted, making its use meaningless in a business context. The only metric that actually improved was recall, from 0.543 to 0.672, but much at the cost of a drop in precision, making the trade-off between the two not favorable for the problem. A last observation that can be made with the results is that the model trained with the resampled data shows an overfitting behavior, with significant differences between training and test results.

Table 4.3: Resampling Results

	Train				**Test**			
	Acc.	**Pre.**	**Rec.**	**AUC**	**Acc.**	**Pre.**	**Rec.**	**AUC**
Original Model	0.959	0.968	0.586	0.792	0.952	0.925	0.543	0.769
Resampled Model	0.935	0.964	0.889	0.931	0.924	0.587	0.672	0.811

The inefficiency of the model trained over resampled data can be explained by looking to the structure of the original data and the methods that are used to implement SMOTE techniques. First of all, SMOTE implementations use k nearest neighbors (KNN) algorithm that relies on Euclidean distance to create synthetic samples, which may be more suitable for continuous variables. Since the dataset used in this study is mostly made up of categorical and discrete variables, this could be one of the main reasons affecting the SMOTE results. Another factor that could be influencing the results has to do with the ineffectiveness of SMOTE techniques on datasets that contain overlapping classes. This means that if classes of the target variable overlap in feature space (if there are no clear

differences between the classes), the data created using the euclidean distance could be of very poor quality, due to the uncertainty in its class, introducing noise into the data and significantly reducing the model's performance.

5

Conclusions, Limitations and Future Work

5.1 Conclusions

The different negative effects related to booking cancellations, not only in terms of revenue management but also in terms of demand forecasting, have made this an emerging and important subject in the hospitality industry. To overcome this behavior, hotels have been forced to take strict actions, such as creating cancellation policies and overbooking strategies that essentially harm customer satisfaction, making the search for solutions that can reduce the negative impacts associated with these strategies an increasing priority for hotels. Aligning this need with the recent explosion of solutions based on data analysis through the use of machine learning, the creation of predictive models capable of helping to identify possible cancellations began to be an investigated solution that shown to have good potential. With this in mind, the aim of this study was to create and analyze some predictive models, using a dataset of reservations from a real hotel chain provided by the Portuguese company Host Hotel Systems, to understand their viability in identifying booking cancellations.

To create these predictive models, an adaptation of the most popular methodology for predictive analysis, CRISP-DM, was used, and a process of the dataset was made, in order to extract the most possible value from it allow the use of various ML algorithms and DL architectures, comparing and analyzing the results of both methodologies. Although DL is at the heart of most of the solutions currently being created in the world of artificial intelligence, its application to tabular data remains questionable and understudied, especially in predictive analysis, not only in terms of predictive capacity but also in complexity costs. The results obtained in chapter 4 confirmed the better performance of ML models, aligning these results with the conclusions obtained in the studies covered in the related work chapter. Despite these results, the literature review and documentation around these DL architectures is still underdeveloped, due to their recent appearance, showing that they still have plenty of room to grow and their use in this study was mainly aimed

at making them known and contributing to their testing. Among all the models tested, the model created using the XGBoost algorithm showed the greatest potential to be taken into a production environment, not only because it was able to achieve good prediction values but also because it was one of the fastest to be trained and to make its predictions.

The observed results show that in all the models, the inability to reach higher values is not due to the quality of the predictions that are identified as canceled by them, but rather to their inability to identify an higher number of cancellations, which means that around 40%/50% of canceled bookings are not identified. The analysis of the models' learning curves showed that this limitation is not with the models themselves but with the nature of the problem and the data. In other words, the imbalance between the number of not canceled and canceled samples in the dataset, making it harder to identify the patterns of the not canceled bookings, and their own non-predictive component may be the main reasons why it is not possible to reach higher values.

Training these models also revealed some indicators that can be favorable for the cancellation prediction, such as the room code, the week of the year in which the reservation is checked in, and the lead time of the reservation. These variables proved to have a significant impact on the creation of the models and should be taken into account in future studies.

Besides the prediction of booking cancellations with ML/DL techniques still being a not widely studied topic, the results of the research works described in chapter 2 and the results obtained during this study, suggest a very favorable path for the use of these models as a solution and encourage further study of them. With this specific study, it was shown that models can be created using a completely different dataset to the one used in other studies, increasing confidence in the use of these models to solve the problem presented.

5.2 Limitations

Since the author of this study is not specialized in the hotel industry, his lack of business knowledge was felt to be the main limitation throughout the process. Since this is a fundamental key in creating predictive models, especially in understanding the data, feedback from people with knowledge of the area proved to be fundamental at various stages of this dissertation.

From a technical point of view, the method used to cross-validate the data may not have been the right one. The ideal would be to use nested cross-validation, where the first split between train/test sets is also dynamic. In this study, cross-validation was only used within the training set, to create a dynamic validation set for the hyperparameters to be found but it would also be important to create this dynamic between the training and the test set split, especially to evaluate different test folds. Although not critical, this detail can be important in model evaluation, giving greater confidence in its results. Allied to this factor, the use of only one static test set also weakens the evaluation of the models. Finally,

the limited documentation surrounding the DL models also made their implementation difficult, especially in terms of hyperparameters tuning, and they may not be used to their full potential.

A final limitation has to do with the application of the model in a more generic context. This model is trained on the specific data of a hotel chain, with specifications and categories that are particular to the hotel, so its use will have to be restricted to the hotel chain in question. Despite this, the creation of these models revealed some insights into booking cancellations and could serve as an example for models created using data from other hotel chains.

5.3 Future Work

The development of predictive models using machine learning or deep learning is, in most cases, a long and continuous process that involves constant learning, not only at a technical level, but also in terms data understanding. The conclusions drawn from this study can and should be used to create new models in order to constantly improve the quality of the models. As already mentioned, it would be very important to find new features that influence booking cancellations, so that they can be used and tested in the models, such as the cancellation policy associated to the booking or the type of deposit, to see if they add any value to it.

Another crucial part of data processing has to do with the method used to encode the categorical variables. In this study, the techniques used were the ones that made the most sense to the author, but many others could be tested to try to improve the results or simply to understand their impact on the models.

Aligned with what has already been said about the DL architectures that have been used, it would be interesting to try out a few more, such as SAINT [50] or an ensemble between a DL architecture and an ML model, as suggested in [48]. As the documentation around these architectures is developed, other options may be considered and tested.

In order to follow the structure of the CRISP-DM methodology, it would be interesting to deploy one of these models and take it into a production environment in order to study its behavior when exposed to real time bookings. Deploying a machine learning model implies a complete study of the requirements it has to maintain itself in a production environment, not only in terms of prediction but also in terms of building continuous evaluation and training plans. Another very important aspect is the creation of a pre-processing pipeline for the inputs that arrive in the system, ensuring that they are fed into the model in the structure it requires.

Bibliography

[1] *Using data science to predict hotel booking cancellations*. IGI Global, 2016, pp. 141–167. DOI: 10.4018/978-1-5225-1054-3.CH006 (cit. on pp. 1, 3, 4, 21).

[2] N Shirisha, K Anusha, A. Kiran, and Y. T. S. Buavani, "Prediction of hotel booking & cancellation using machine learning algorithms," in *2023 International Conference on Computer Communication and Informatics (ICCCI)*, IEEE, 2023, pp. 1–4 (cit. on p. 3).

[3] N. Antonio, A. De Almeida, and L. Nunes, "Predicting hotel booking cancellations to decrease uncertainty and increase revenue," *Tourism & Management Studies*, vol. 13, no. 2, pp. 25–39, 2017 (cit. on p. 3).

[4] P. Chapman, J. Clinton, R. Kerber, *et al.*, "The crisp-dm user guide," in *4th CRISP-DM SIG Workshop in Brussels in March*, sn, vol. 1999, 1999 (cit. on p. 4).

[5] R. Wirth and J. Hipp, "Crisp-dm: Towards a standard process model for data mining," in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, Manchester, vol. 1, 2000, pp. 29–39 (cit. on pp. 4, 5).

[6] J. Saltz, "Crisp-dm is still the most popular framework for executing data science projects," *Data Science PM*, 2022. [Online]. Available: https://www.datascience-pm.com/ (cit. on p. 4).

[7] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, no. 3, pp. 685–695, 2021 (cit. on p. 5).

[8] E. G. Learned-Miller, "Introduction to supervised learning," *I: Department of Computer Science, University of Massachusetts*, vol. 3, 2014 (cit. on p. 6).

[9] Q. Liu and Y. Wu, "Supervised learning," 2012-01. DOI: 10.1007/978-1-4419-1428-6_451 (cit. on p. 6).

[10] K. El Bouchefry and R. S. de Souza, "Chapter 12 - learning in big data: Introduction to machine learning," in *Knowledge Discovery in Big Data from Astronomy and Earth Observation*, P. Škoda and F. Adam, Eds., Elsevier, 2020, pp. 225–249, ISBN: 978-0-12-819154-5. DOI: https://doi.org/10.1016/B978-0-12-819154-5.00023-0.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128191545000230 (cit. on p. 6).

[11] N. Grira, M. Crucianu, and N. Boujemaa, "Unsupervised and semi-supervised clustering: A brief survey," *A review of machine learning techniques for processing multimedia content*, vol. 1, no. 2004, pp. 9–16, 2004 (cit. on p. 6).

[12] L. Van Der Maaten, E. Postma, J. Van den Herik, *et al.*, "Dimensionality reduction: A comparative," *J Mach Learn Res*, vol. 10, no. 66-71, 2009 (cit. on p. 7).

[13] K. J. Cios, R. W. Swiniarski, W. Pedrycz, *et al.*, "Unsupervised learning: Association rules," *Data Mining: A Knowledge Discovery Approach*, pp. 289–306, 2007 (cit. on p. 7).

[14] Y. Li, *Deep reinforcement learning: An overview*, 2018. arXiv: 1701.07274 [cs.LG] (cit. on p. 7).

[15] H. Belyadi and A. Haghighat, "Chapter 5 - supervised learning," in *Machine Learning Guide for Oil and Gas Using Python*, H. Belyadi and A. Haghighat, Eds., Gulf Professional Publishing, 2021, pp. 169–295, ISBN: 978-0-12-821929-4. DOI: https://doi.org/10.1016/B978-0-12-821929-4.00004-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128219294000044 (cit. on p. 7).

[16] V. Nasteski, "An overview of the supervised machine learning methods," *HORIZONS.B*, vol. 4, pp. 51–62, 2017-12. DOI: 10.20544/HORIZONS.B.04.1.17.P05 (cit. on p. 7).

[17] Umar. "Sigmoid function." (), [Online]. Available: https://artificialintelligencestechnology.com/ml/sigmoid-function/ (visited on 2023-10-19) (cit. on p. 8).

[18] D. Berrar, "Bayes' theorem and naive bayes classifier," *Encyclopedia of bioinformatics and computational biology: ABC of bioinformatics*, vol. 403, p. 412, 2018 (cit. on p. 8).

[19] S. Joelsson, J. Benediktsson, and J. Sveinsson, "Random forest classifiers for hyperspectral data," in *Proceedings. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS '05.*, vol. 1, 2005, 4 pp.–. DOI: 10.1109/IGARSS.2005.1526129 (cit. on p. 9).

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015 (cit. on p. 10).

[21] D. Graupe, *Deep learning neural networks: design and case studies*. World Scientific Publishing Company, 2016 (cit. on p. 10).

[22] K. Suzuki, *Artificial neural networks: methodological advances and biomedical applications*. BoD–Books on Demand, 2011 (cit. on p. 10).

[23] A. D. Rasamoelina, F. Adjailia, and P. Sinčák, "A review of activation function for artificial neural network," in *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2020, pp. 281–286. DOI: 10.1109/SAMI48414.2020.9108717 (cit. on p. 11).

[24] S. Lee, S. Jung, and J. Lee, "Prediction model based on an artificial neural network for user-based building energy consumption in south korea," *Energies*, vol. 12, p. 608, 2019-02. DOI: 10.3390/en12040608 (cit. on p. 11).

[25] S. Popov, S. Morozov, and A. Babenko, "Neural oblivious decision ensembles for deep learning on tabular data," *arXiv preprint arXiv:1909.06312*, 2019 (cit. on p. 12).

[26] M. Joseph, "Pytorch tabular: A framework for deep learning with tabular data," *arXiv preprint arXiv:2104.13638*, 2021 (cit. on p. 12).

[27] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 6679–6687 (cit. on p. 13).

[28] H.-T. Cheng, L. Koc, J. Harmsen, *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10 (cit. on p. 13).

[29] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for ctr prediction," *arXiv preprint arXiv:1703.04247*, 2017 (cit. on p. 14).

[30] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*. Cambridge University Press, 2023 (cit. on p. 14).

[31] D Ramyachitra and P. Manikandan, "Imbalanced dataset classification and solutions: A review," *International Journal of Computing and Business Research (IJCBR)*, vol. 5, no. 4, pp. 1–29, 2014 (cit. on p. 15).

[32] V. Ganganwar, "An overview of classification algorithms for imbalanced datasets," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 4, pp. 42–47, 2012 (cit. on p. 15).

[33] P. Gulati, "Hybrid resampling technique to tackle the imbalanced classification problem," 2020 (cit. on p. 15).

[34] R. Blagus and L. Lusa, "Smote for high-dimensional class-imbalanced data," *BMC bioinformatics*, vol. 14, pp. 1–16, 2013 (cit. on p. 15).

[35] E. A. Team. "How to explain the roc curve and roc auc score?" (), [Online]. Available: https://www.evidentlyai.com/classification-metrics/explain-roc-curve (visited on 2023-10-18) (cit. on pp. 18, 19).

[36] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, IOP Publishing, vol. 1168, 2019, p. 022 022 (cit. on pp. 19, 20).

[37] Y. Dar, V. Muthukumar, and R. G. Baraniuk, "A farewell to the bias-variance tradeoff? an overview of the theory of overparameterized machine learning," *arXiv preprint arXiv:2109.02355*, 2021 (cit. on p. 19).

[38] S. Mallick. "Bias-variance tradeoff in machine learning." (2017-02-28), [Online]. Available: `https://learnopencv.com/bias-variance-tradeoff-in-machine-learning` (visited on 2023-10-19) (cit. on p. 19).

[39] S. Ivanov and V. Zhechev, "Hotel revenue management–a critical literature review," *Tourism: an international interdisciplinary journal*, vol. 60, no. 2, pp. 175–197, 2012 (cit. on p. 21).

[40] P. H. Saputro and H. Nanang, "Exploratory data analysis & booking cancelation prediction on hotel booking demands datasets," *Journal of Applied Data Sciences*, vol. 2, no. 1, pp. 40–56, 2021 (cit. on p. 21).

[41] M. Adil, M. F. Ansari, A. Alahmadi, J.-Z. Wu, and R. K. Chakrabortty, "Solving the problem of class imbalance in the prediction of hotel cancelations: A hybridized machine learning approach," *Processes*, vol. 9, no. 10, p. 1713, 2021 (cit. on pp. 21, 41).

[42] H.-C. Huang, A. Y. Chang, C.-C. Ho, *et al.*, "Using artificial neural networks to establish a customer-cancellation prediction model," *Przeglad Elektrotechniczny*, vol. 89, no. 1b, pp. 178–180, 2013 (cit. on p. 21).

[43] M. Falk and M. Vieru, "Modelling the cancellation behaviour of hotel guests," *International Journal of Contemporary Hospitality Management*, vol. 30, no. 10, pp. 3100–3116, 2018 (cit. on pp. 21, 33).

[44] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" *Advances in Neural Information Processing Systems*, vol. 35, pp. 507–520, 2022 (cit. on p. 22).

[45] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022 (cit. on p. 22).

[46] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kégl, and D. Rousseau, "The higgs boson machine learning challenge," in *NIPS 2014 workshop on high-energy physics and machine learning*, PMLR, 2015, pp. 19–55 (cit. on p. 22).

[47] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 932–18 943, 2021 (cit. on p. 22).

[48] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022 (cit. on pp. 23, 57).

[49] H. J. Weerts, A. C. Mueller, and J. Vanschoren, "Importance of tuning hyperparameters of machine learning algorithms," *arXiv preprint arXiv:2007.07588*, 2020 (cit. on p. 42).

[50] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, and T. Goldstein, "Saint: Improved neural networks for tabular data via row attention and contrastive pre-training," *arXiv preprint arXiv:2106.01342*, 2021 (cit. on p. 57).

A

Hyperparameters Tuning

Model	Hyperparameter	Values
Random Forest	n_estimators	[150, 200]
	max_depth	[10, 15, 20, 25]
	min_samples_split	[2, 5, 10]
	bootstrap	[True, False]
	criterion	['gini', 'entropy', 'log_loss']
	min_samples_leaf	[1, 2, 4]
LightGBM	n_estimators	[50, 100, 150, 200, 250]
	max_depth	[2, 3, 4, 5]
	learning_rate	[0.001, 0.01, 0.1, 0.2, 0.3}
AdaBoost	n_estimators	[25, 50, 100, 150]
	learning_rate	[0.01, 0.1, 0.2, 0.3, 1.0]
XGBoost	n_estimators	[100, 200, 150, 200, 250]
	max_depth	[2, 3, 4, 5]
	learning_rate	[0.01, 0.1, 0.2, 0.3]
	alpha	[0, 0.1, 0.2, 0.3]
	lambda	[[1, 5, 10, 20]
	min_child_weight	
CatBoost	iterations	[100, 200, 250, 500, 750]
	learning_rate	[0.01, 0.1, 0.2]
	depth	[2, 4, 6, 8]
	l2_leaf_reg	[1, 3, 5]

Table A.1: Hyperparameters and values for each model.

B

Confusion Matrixes of all models

B.1 Machine Learning Models

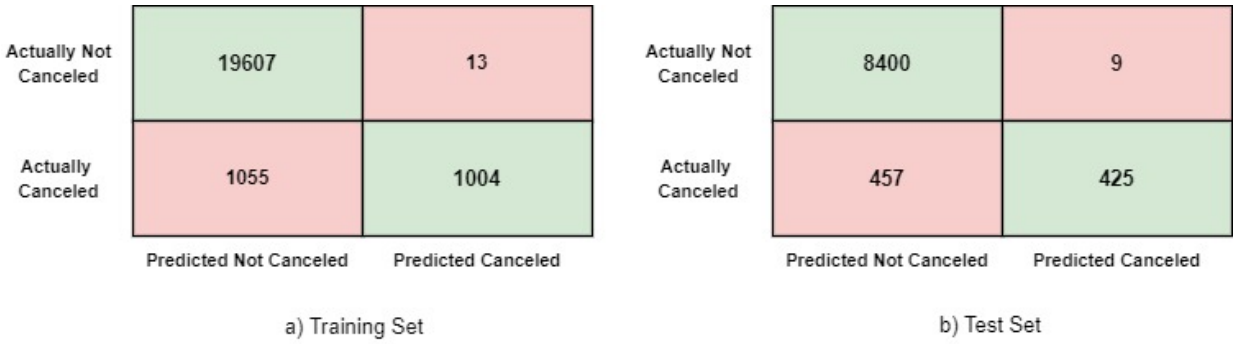

Figure B.1: Confusion Matrixes of Random Forest model.

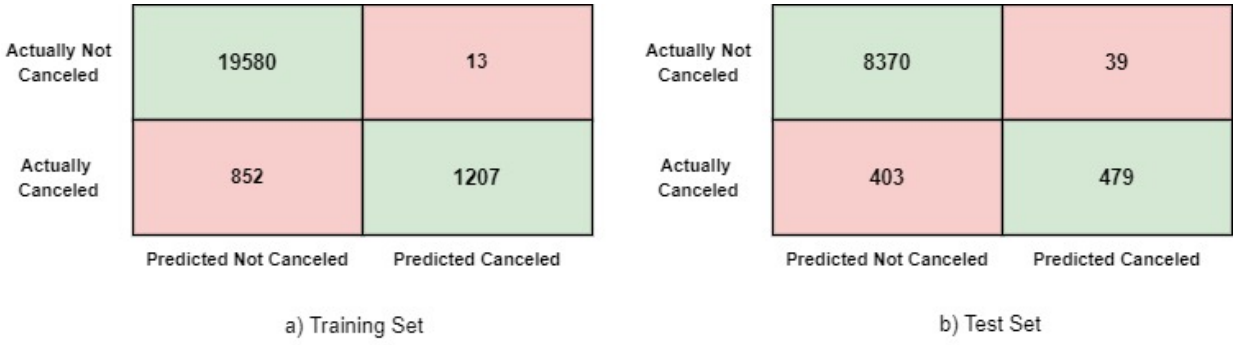

Figure B.2: Confusion Matrixes of XGBoost model.

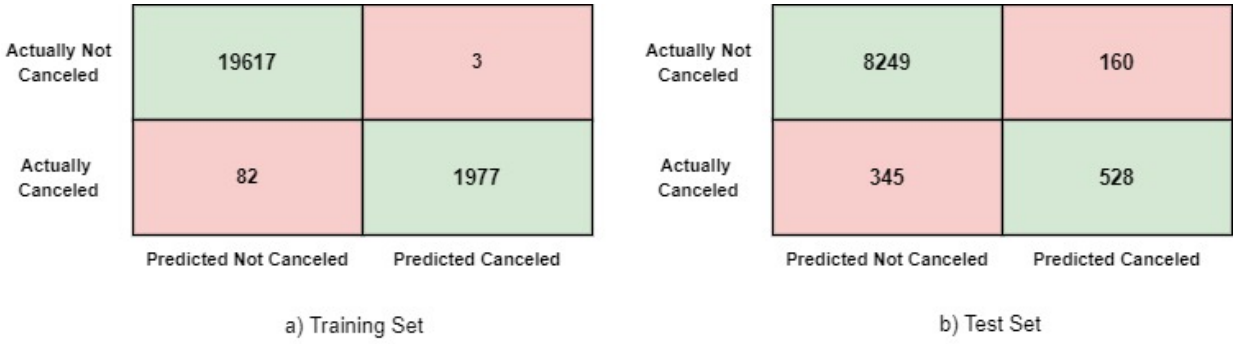

Figure B.3: Confusion Matrixes of AdaBoost model.

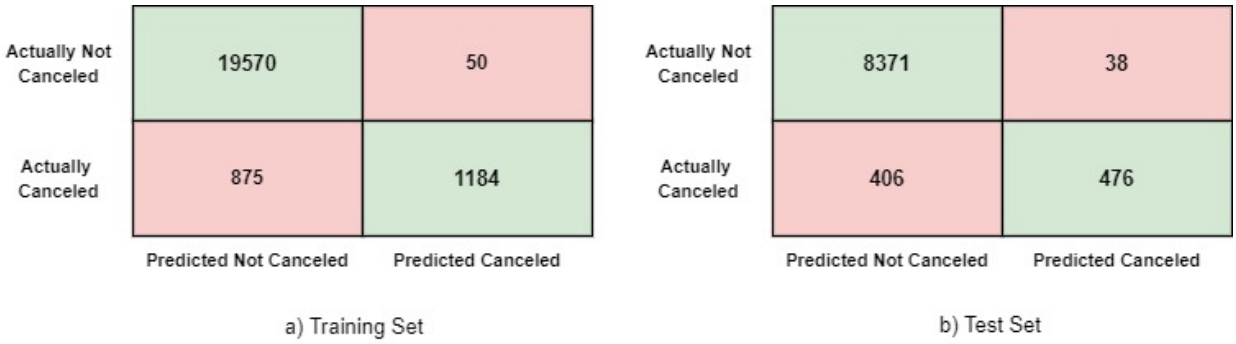

Figure B.4: Confusion Matrixes of LightGBM model.

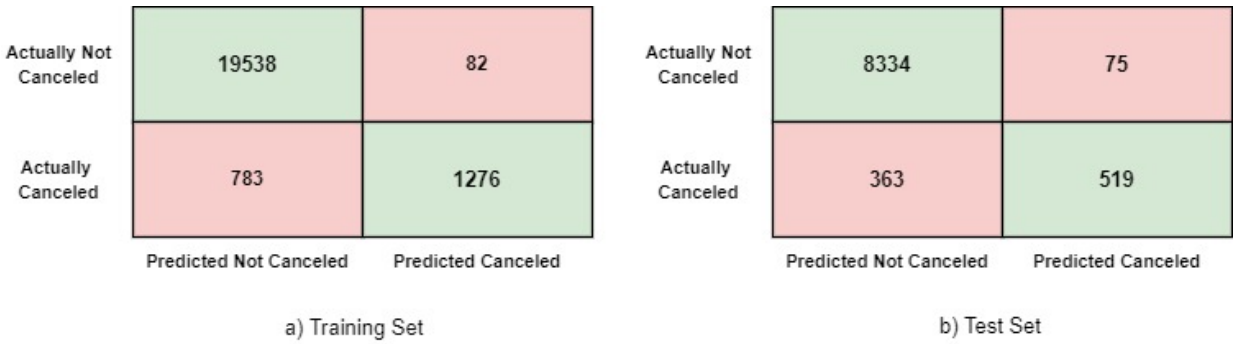

Figure B.5: Confusion Matrixes of CatBoost model.

B.2 Deep Learning Models

Figure B.6: Confusion Matrixes of Wide&Deep model.

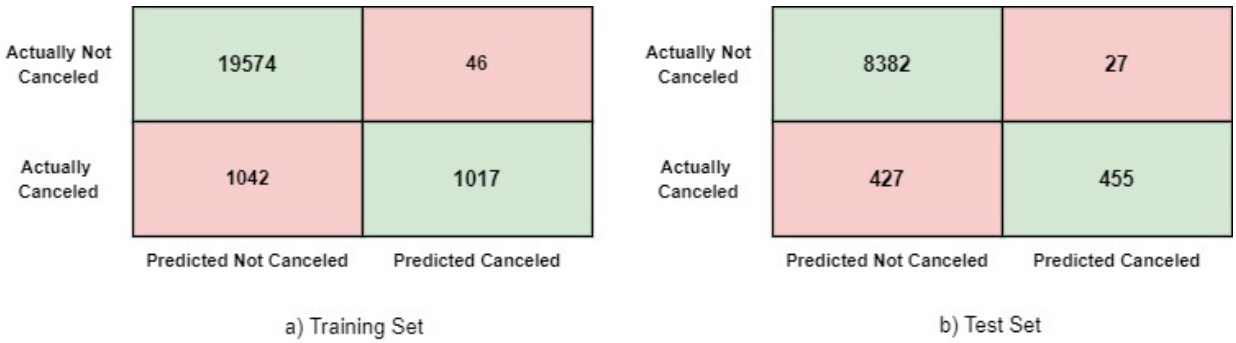

Figure B.7: Confusion Matrixes of DeepFM model.

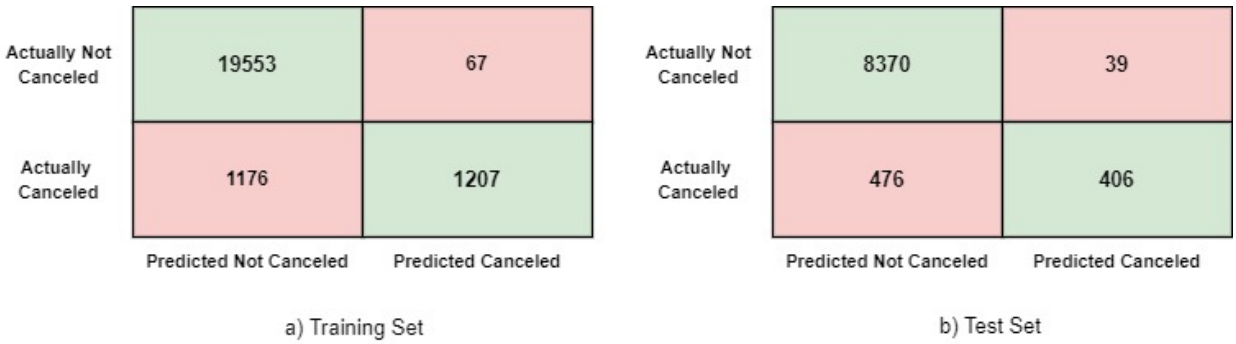

Figure B.8: Confusion Matrixes of AutoInt model.

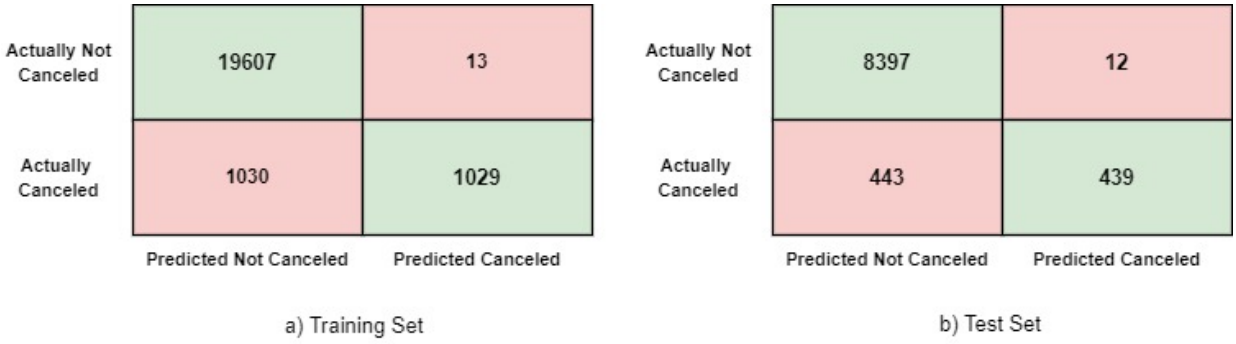

Figure B.9: Confusion Matrixes of TabNet model.