TÉCNICO LISBOA

Predicting Real Operating Room Occupation, an Interpretable ML Approach

Maria Teresa de Carvalho Dias Marcelino

Thesis to obtain the Master of Science Degree in

Biomedical Engineering

Supervisor(s): Prof. Cláudia Alexandra Magalhães Soares
Prof. Qiwei Han

Examination Committee

Chairperson: Prof. Maria do Rosário De Oliveira Silva
Supervisor: Prof. Cláudia Alexandra Magalhães Soares
Member of the Committee: Prof. Iolanda Raquel Fernandes Velho

October 2021

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Preface

The work presented in this thesis was performed at CUF (Lisbon, Portugal), during the period March-October 2021, under the supervision of Eng. João Leal and Eng. Daniela Burrinha. The thesis was co-supervised at Instituto Superior Técnico by Prof. Cláudia Soares and also by Prof. Qiwei Han, Professor of Data Science and Business Analytics at Nova SBE.

For this thesis, a non-disclosure agreement was signed with CUF, for the protection of sensitive information shared for the development of the thesis.

Acknowledgments

Em primeiro lugar, gostaria de agradecer aos meus orientadores, Professora Cláudia e Professor Qiwei, por toda a confiança que depositaram em mim ao longo destes meses e por me receberam como sua aluna. A forma como me trasmitiram a sua curiosidade e paixão por data science foi uma motivação ao longo de toda a realização do trabalho, e o seu pensamento crítico e orientação serão um exemplo a seguir. Foi um privilégio poder aprender tanto com eles.

Um agradecimento especial à Daniela e ao João, que me apresentaram este projeto e confiaram em mim para o desenvolver em conjunto com a CUF. Muito obrigada pelo acompanhamento, por me apoiarem e se mostrarem sempre disponíveis para responder às minhas questões.

Numa nota mais pessoal, agradeço às pessoas com quem tive o prazer de partilhar esta longa caminhada. Primeiro, aos amigos de longa data que Coimbra me deu, que mesmo longe acompanharam sempre o meu crescimento e conquistas. Aqueles com quem sem dúvida partilhei os segredos da minha cidade, Coimbra, e que levarei p'rá vida. Por mais que nunca tenha sentido o verdadeiro significado de ser estudante de Coimbra e o peso da capa negra, foi também aqui, nesta cidade, que é minha desde sempre, que vivi dos melhores anos da minha vida, criei amizades e partilhei momentos.

Em segundo lugar, aos amigos que o Técnico me presenteou. Foi há 5 anos que cruzei a Alameda pela primeira vez e esta foi a primeira paragem de uma viagem por alguns dos melhores anos da minha vida. Foi aqui que aprendi a pensar, a olhar para o mundo de outra forma e a superar-me. Viagem esta que não faria sozinha. Obrigada Lu, Mary J, Maria, Marga, Pipa, Cat, Carol, João, Gonçalo e Nuno por terem sido a minha casa durante estes 5 anos e alinharem sempre nas minhas aventuras. Foi com eles que partilhei desde noites de estudo na P10 e almoços contrariados no social a diversos momentos de lazer e viagens. Obrigada por me fazerem querer ser melhor todos os dias e me acompanharem desde então. À Maria e ao Hugo, muito obrigada por todo o apoio e ajuda, e pela companhia durante estes últimos meses de tese.

Por último, à minha família, a base de tudo aquilo que sou hoje e peça chave dos meus sucessos. Aos meus pais, pelo apoio incondicional, por terem sempre as palavras certas e por serem um exemplo daquilo que quero ser. Aos meus irmãos, por partilharem as alegrias comigo e serem incansáveis sempre que preciso. Aos meus avós, pelo amor incondicional. Obrigada por acreditarem em mim e me terem deixado voar e fazer as minhas próprias escolhas, mesmo que isso implicasse estar longe. Foi sem dúvida um desafio que valeu a pena.

Ao Técnico, que me proporcionou tudo isto e que permitiu ir além fronteiras até à Bélgica, não te digo adeus porque te levo sempre comigo.

Resumo

Hoje em dia, o potencial do uso de técnicas de aprendizagem automática (ML) para resolver problemas do mundo real é amplamente explorado, e muitos são os domínios de aplicação, como cibersegurança, aviação e saúde, onde há pesquisas aprofundadas sobre sua aplicabilidade. Com a quantidade de dados recolhidos atualmente no contexto hospitalar, modelos capazes de aprender e melhorar automaticamente sustentados na exploração dos dados podem solucionar problemas que colocam em risco o bom funcionamento dos hospitais. O bloco operatório é um ambiente de alto custo e a sua utilização deve ser eficiente. Assim, o trabalho proposto foca-se no desenvolvimento de modelos de ML interpretáveis de previsão para integração num sistema de suporte à decisão a fim de melhorar a previsão dos tempos cirúrgicos, comparando-os com métodos tradicionais. Implementámos três modelos de ML, XGBoost, RuleFit e uma rede neuronal, e analisamos o seu desempenho, incluindo precisão e interpretabilidade. Para cada um dos algoritmos, implementamos três estratégias diferentes. Posteriormente, uma vez que as durações cirúrgicas mostraram um desequilíbrio significativo e isso pode prejudicar o desempenho de algoritmos de ML, treinamos uma *Gaussian Mixture Model* (GMM) para aprender a distribuição de probabilidade nos valores minoritários da *label*, permitindo superar o desequilíbrio. O desempenho dos modelos em conjuntos de dados balanceados e desequilibrados foram comparados usando o *Utility-based Algorithm* (UBA). Este trabalho é uma evidência de que a implementação adequada de tecnologias de ML interpretáveis podem melhorar significativamente os padrões atuais de estimativa, representando uma redução de custos, mantendo a confiança dos *decision-makers* no sistema.

Palavras-chave: Bloco Operatório, Aprendizagem Automática, Eficiência, Duração da Cirurgia, Modelos Interpretáveis

Abstract

Nowadays, the potential of using Machine Learning (ML) techniques to solve real-world problems is extensively explored, and many are the application domains such as cybersecurity, aviation and healthcare, where there is in-depth research into their applicability. With the amount of data currently gathered in the hospital environment, models capable of learning and improving automatically through the use of data might solve problems that endanger the proper functioning of hospitals. The Operating Room (OR) is a high-cost environment, and its usage must be efficient. Therefore, our presented solution focuses on developing interpretable prediction ML models for an OR decision support system to improve the prediction of surgical times, comparing them with traditional methods to aid the OR scheduling process. We implemented three different ML models, XGBoost, RuleFit and a neural network, and we compared and analyzed their performance, including both accuracy and interpretability. For each of these algorithms, we implemented three different strategies. Then, since surgical durations showed a significant imbalance and this is known to hinder the performance of accuracy-based ML algorithms, we trained a Gaussian Mixture Model (GMM) to learn the probability distribution on the minority values of our label enabling sampling to overcome the imbalance. The performance of the models on balanced and imbalanced datasets was compared using the Utility-Based Algorithm (UBA). This research work is an evidence that the proper implementation of interpretable ML technologies can significantly improve current standards of estimation, representing a cost reduction from an operation's perspective, maintaining the decision-makers' confidence in the system.

Keywords: Operating Room, Machine Learning, Efficiency, Surgery Case Duration, Interpretable Models

Contents

Preface v
Acknowledgments vii
Resumo ix
Abstract xi
List of Tables xvii
List of Figures xix
List of Algorithms xxi
Nomenclature xxiii
List of Acronyms xxvi

1 Introduction **1**
1.1 Motivation 1
1.2 Objectives and Contributions 2
1.3 State of The Art 2
1.3.1 Operating Room Stages 2
1.3.2 Prediction of Case-time Duration 3
1.3.3 Related Work 4
1.4 Thesis Outline 6

2 Theoretical Background **7**
2.1 Supervised Learning 7
2.2 Regression Algorithms 8
2.2.1 Extreme Gradient Boosting 8
2.3 Interpretable Methods 8
2.3.1 RuleFit 8
2.4 Feedforward Neural Network 9
2.5 Categorical Data Encoders 11
2.6 Imbalanced Approaches 11
2.6.1 Synthetic Minority Over-sampling Technique 12
2.6.2 Gaussian Mixture Model 13
2.7 Regression Metrics 15

2.7.1 Mean Squared Error (MSE) . . . 15
2.7.2 Mean Absolute Percentage Error (MAPE) . . . 15
2.7.3 Utility-Based Regression . . . 15
2.8 Interpretability and Model Explanations . . . 19
2.8.1 Shapley Additive Explanations . . . 19
2.8.2 Rashomon Curves . . . 20

3 Data Analysis and Preparation **22**
3.1 Dataset Introduction . . . 22
3.2 Exploratory Data Analysis . . . 23
3.3 Time Series Analysis . . . 28
3.4 Feature Engineering . . . 30
3.5 Data Imbalance . . . 30
3.6 Missing Data . . . 31
3.7 CUF Predictions . . . 32
3.8 Remotion of erroneous data . . . 33
3.9 Data and Features Selection . . . 33
3.10 Encoding . . . 34

4 Modeling **35**
4.1 Proposed Approaches . . . 35
4.1.1 General Model . . . 35
4.1.2 Specialty-specific Models . . . 36
4.1.3 Surgeon-specific Models . . . 36
4.2 Data Split . . . 36
4.3 Machine Learning Algorithms Implementation . . . 36
4.3.1 Extreme Gradient Boosting Implementation . . . 36
4.3.2 RuleFit Implementation . . . 42
4.3.3 Feedforward Neural Network Implementation . . . 48

5 Balanced Approach **58**
5.1 Deal With Imbalanced Data . . . 58
5.2 Interpretability Curve for Imbalanced Data . . . 61
5.3 Balanced Model Results . . . 62
5.3.1 Model Selection . . . 62
5.3.2 Performance Evaluation . . . 65

6 Results from an Operation's Perspective, Generalization Error and Conclusions **70**
6.1 Final Model and Generalization Error . . . 70
6.2 Results from an Operation's Perspective . . . 72
6.3 Conclusions . . . 74

6.3.1 Future Work 75

Bibliography **76**

A Technical Nomenclature **83**

A.1 Data Dictionary 83

A.2 Types of Anesthesia 84

B Code **85**

B.1 Code Organization 85

B.1.1 Python 85

B.1.2 R 86

List of Tables

4.1 Set of parameters used in XGBoost Tuning. 37
4.2 Validation error obtained for each approach with XGBoost algorithm and from CUF model. 38
4.3 Set of parameters used in Ensemble Methods Tuning. 44
4.4 Selection of parameters used by the rule fitting method in each model approach. 45
4.5 Validation error obtained for each approach with RuleFit and from CUF model. 45
4.6 Five of the rules that were generated by general RuleFit model, along with their support and importance. 47
4.7 Set of parameters used in Feedforward Neural Network Tuning. 49
4.8 Range of Batch Size used in Feedforward Neural Network Tuning. 52
4.9 MSE results for surgeon-specific models, specialty-specific models, general model and CUF model. 53
4.10 Validation error obtained for each approach with FNN and from CUF model. 54

5.1 Model selection for each model approach and algorithm type for imbalanced and balanced data based on Interpretability Curve. 65
5.2 Results of utility metrics for general model with imbalanced data and with balanced data from GMM. 68
5.3 Results of utility metrics for ophthalmology specialty models with imbalanced data and with balanced data from GMM. 68
5.4 Results of utility metrics for surgeon ID 96440008 model with imbalanced data and with balanced data from GMM. 68
5.5 Summary of the RMSE in minotiry classes for each model approach and its comparison with CUF predictions. 69
5.6 Summary of the RMSE in all classes for each model approach and its comparison with CUF predictions. 69

6.1 Summary of the generalization error measured by RMSE in minotiry classes for each model approach and its comparison with CUF predictions. 71
6.2 Summary of the RMSE in each such model and its comparison with RMSE of CUF predictions. 73
6.3 Ratio between preventive costs for each model in relation to CUF's baseline cost. 73

List of Figures

2.1 Rectified Linear Activation Function. 10
2.2 Synthetic samples creation process by SMOTE. 12
2.3 Covariance types. 13
2.4 Relevance function using ”extremes” method and its utility surface. 16
2.5 The utility surface obtained with the relevance function ϕ shown in Figure 2.4 a), with p = 0.95. 18
2.6 The utility surface generated by UBL package when set the type of surface that is being interpolated as cost. 19
2.7 The Rashomon Curve. 21

3.1 Number of surgeries per year since 2017 to 2020. 23
3.2 Gender and age distribution over the patients. 24
3.3 Anesthesia and procedures number distribution over surgeries. 25
3.4 Frequency of surgeries per specialty. 26
3.5 Anesthesia and procedures number distribution over surgeries. 26
3.6 Week distribution of specialities. 27
3.7 Week distribution of procedures. 27
3.8 Monthly distribution of surgeries from 2017 to 2020. 28
3.9 Yearly and monthly seasonality. 29
3.10 Trend resulting from the use of *Prophet* tool after the inclusion of an exceptional season. . 29
3.11 Histogram and density plot of target. 31
3.12 Visualization of missing values. 31
3.13 Error distribution since 2017 to 2020. 32
3.14 Summary of the data cleaning process. 34

4.1 Distribution of within cases using XGBoost and CUF predictions for each specialty. 38
4.2 SHAP summary plot of XGBoost general model. 39
4.3 SHAP summary plot of orthopedics specialty. 40
4.4 SHAP summary plot of surgeon ID 132273102. 41
4.5 Feature importance chart for general model. 42
4.6 Feature importance computed in general model with SHAP values. 43

4.7 Distribution of within cases using RuleFit algorithm and CUF predictions for each specialty. 46
4.8 Bar chart listing the explanatory variables based on their significance level for the general model. 47
4.9 Architecture of Feedforward Neural Network. 50
4.10 Loss and validation loss before and after parameters tuning. 51
4.11 Split of data into test, train and validation set. 52
4.12 Validation loss and training loss of final general model. 53
4.13 MSE distribution in Boxplots. Box represents the data that exists between the first and third quartile. 54
4.14 Distribution of within cases using Feedforward Neural Network and CUF predictions for each specialty. 55
4.15 Loss and validation loss before and after parameters tuning. 56
4.16 Features individual contributions in general model. 57

5.1 Plot of true labels with predicted labels. 59
5.2 GMM selection for ophthalmology specialty using AIC and BIC scores. 60
5.3 Selection of best number of components for ophthalmology specialty. 60
5.4 Distribution of surgery durations for the ophthalmology specialty dataset before and after the generation of synthetic data in minority classes by GMM. 60
5.5 The proposed Interpretability Curve. 62
5.6 XGBoost model selection for ophthalmology. 63
5.7 RuleFit model selection for ophthalmology. 64
5.8 Illustration of train and validation error performance as a function of rules number. 64
5.9 The relevance function for the prediction of surgeries times in case of surgeon ID 9644000 66
5.10 An utility surface for the the prediction of surgeries obtained with the relevance function shown in Figure 5.9, with p = 0.90. 67

6.1 Cost comparison of baseline and proposed solutions. Total cost in function of ratio. 74

List of Algorithms

1 Operating room decision support system. 71

Nomenclature

Greek symbols

Δ	Difference.
Γ_B	Bounded-loss function.
Γ_c	Bounded-loss function.
κ	Gaussian distribution.
μ	Mean of a Gaussian distribution.
ϕ	Relevance function.
ϕ^p	Joint relevance function.
Σ	Covariance of a Gaussian distribution.
θ	Rashomon parameter.
K	Number of models generated by Gaussian Mixture Models.

Roman symbols

$\hat{L}$	Loss function.
$\hat{R}_{ratio}$	Rashomon Ratio.
$\hat{R}_{set}$	Rashomon Set.
$\hat{y}$	Estimated target value.
$\hat{z}$	Predicted class.
$\mathscr{F}$	Hypothesis space.
$\mathscr{V}$	Volume space.
D	Number of dimensions.
f	Function.
k	Number of adjustable parameters.

L Likelihood.

N Number of samples.

n Sample size.

U Utility function.

Y Original domain of the target variable.

y Target value.

z True class.

t_E Threshold.

p Weight parameter.

Subscripts

∞ Free-stream condition.

i, j Computational indexes.

x,y Cartesian components.

List of Acronyms

ADAM Adaptive Moment Estimation

AI Artificial Intelligence

AIC Akaike Information Criterion

AUC-ROC Area Under the Receiver Operating Characteristic Curve

BIC Bayesian Information Criterion

EDA Exploratory Data Analysis

EHR Electronic Health Record

FN False Negatives

FNN Feedforward Neural Network

FP False Positives

GMM Gaussian Mixture Model

ICU Intensive Care Unit

ID Identity Document

KDE Kernel Density Estimation

L1 Lasso

L2 Ridge

LIN Least-squares Linear Regression

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

ML Machine Learning

MSE Mean Squared Error

NaN Not a Number

NN Neural Network

OR Operating Room

ORN Operative Registry Number

OWL Ordered Weighted ℓ_1

PACU Post-Anesthesia Care Unit

ReLU Rectified Linear Unit

RMSE Root Mean Squared Error

RMSprop Root Mean Squared Propagation

SGD Stochastic Gradient Descendent

SHAP Shapley Additive Explanations

SMOTE Synthetic Minority Over-sampling Technique

STEP Stepwise regression

TOM Tabela Ordem dos Médicos

TP True Positives

UBA Utility-Based Algorithm

UBL Utility-Based Learning

XGBoost Extreme Gradient Boosting

Chapter 1

Introduction

1.1 Motivation

For public or private hospitals, efficiency is a common goal. Both need to manage the limited resources available to provide high quality care and care for a larger number of patients. However, this management is hard due to the unpredictability of various events that escape the foresight of the best managers and medical doctors.

More precisely, the schedule planning of Operating Rooms (ORs) is one of the biggest challenges in the health sector since this service is a hospital key element, responsible for around 42% [1] of income but, simultaneously, due to high cost of use, most hospital expenses are related to the OR, around 35% to 40% of hospital's costs [2]. Nowadays, operating rooms are costly, ranging from $30 to more than $100 per minute. Therefore, it is crucial to maximize this critical financial bottleneck's efficiency [3].

Besides financial criticality, this service is one of the biggest headaches in the hospital due to its extremely high complexity. The interactions between different healthcare stakeholders (such as surgeons, patients, nurses, and anesthesiologists), the difficulty of predicting the time in certain types of procedures due to unpredictable patient circumstances, the need for sterile material that depends on third parties, and the availability of beds in Post-Anesthesia Care Unit (PACU) are just a few reasons that help us understand the difficulty in managing this service [1].

Regarding the hospital group under study, CUF is a Portuguese private healthcare provider with 18 clinics and hospitals spread across the country. CUF is one of the most important private groups in Portugal that between 2017 and 2020 treated 190 thousand patients on operating rooms in its 15 units equipped with an OR. To highlight the importance of CUF in Portugal, last year 2020 CUF won the "Trusted Brand" award promoted by Seleções dos Reader's Digest magazine, in the "Private Clinics and Hospitals" category, a choice that has been made for the sixth consecutive year. At the CUF operating room management level recently, in 2019, a global management consulting firm delivered a report that stated inefficiencies at the OR organization system, which came to support the fact that these problems exist.

There is a cascade of negative consequences due to inadequate estimation of surgery times with a

high impact on health professionals that can be highly affected since if time is wasted unnecessarily, the higher is the probability of working over hours, which generates discontent in the workplace. Moreover, it may also impact patients since surgery waiting lines can be long and ineffective scheduling quickly leads to very high waiting times during patient flow.

To conclude, in this thesis, we seek to solve this problem and explore methods for minimizing the incorrect prediction of surgical times, minimizing blockings between two consecutive surgeries to provide good care to all patients and increase health professionals' satisfaction.

1.2 Objectives and Contributions

The goal of the thesis is to develop a Machine Learning (ML) model to improve the operating room planning through a case-time estimation using CUF historical data from 2017 to 2020.

The purpose of the work is to find an accurate method to estimate the surgery times to optimize the operating room process flow, improving surgery allocation in the overall scheduling and care delivery from hospitals. For this approach, several factors were considered, such as type of surgery, operating room time, pseudo-anonymized patient information, and type of procedures.

1.3 State of The Art

This Section presents an overview of what already exists in the literature regarding the implementation of machine learning in the scope of operating rooms efficiency improvement efforts. Important concepts are introduced to better understand the operating room's organization and its dependence on other healthcare services.

1.3.1 Operating Room Stages

Operating room scheduling does not just depend only on the use of the operating room itself, there is a whole path that the patient has to go through, and all stages are directly correlated. As described by Abedini et al. [4] we can divide the surgical process into three stages:

- The peri-operative process
- The intra-operative process
- The post-operative process

The peri-operative stage is related to administrative and clinical admissions, and the interaction between the patient and the anesthesiologist. Thus, in this first stage, some important information is gathered after the patient's arrival at the hospital to be admitted and the patient is prepared for surgery that occurs in the next phase. The surgery is performed during the intra-operative stage and in the last stage the patient is moved to a PACU where it waits for recovery after surgery and anesthesia.

Therefore, there is a complex environment to manage which depends on a lot of factors. In each stage, some problems can represent a bottleneck with a significant impact on all processes, such as delayed patient registration, staff unavailability, case duration accuracy, and the lack of beds in PACU. These factors create congestion throughout the operating room organization because patients cannot be moved to the next stage and health professionals have to keep them in the state they are in [3, 4].

Regarding CUF, we had the opportunity to visit CUF Sintra in order to better understand the process that the patient undergoes since it arrives at the hospital until it leaves. This visit gave us an overview of the patient flow that was extremely important in identifying the steps that are wasting unnecessary time, realizing CUF's expectations and confirming the existence of the same problems that were found in the bibliographic review.

1.3.2 Prediction of Case-time Duration

The accurate prediction of surgical procedure times is essential to maintain efficiency and avoid a cascade of delays in OR. Thus the use of inaccurate strategies may have a significant impact on the entire service and waiting lists.

Nowadays, the historical information on the OR operation is well annotated and there is a lot of information available, such as the surgical service performing the procedure, the duration of surgery, and the patient's information, which has a vast potential to optimize the OR pathway. However, these data are still not fully explored in most hospitals and forecasts of the surgery duration are made based on the experience and opinion of surgeons, that estimate the operating times that they consider necessary, or by using simple statistics on the conventional Electronic Health Records (EHRs), the electronic collection of a patient's medical history where the historical average for each case duration can be performed.

The study conducted by Laskin et al. [5] with oral and maxillofacial surgeons showed that only 26% of surgeon estimates were accurate and there is an overestimation in 42% of the analysed cases. Overestimation occurs because various factors can influence the doctor's prediction, simply because complications arise during the procedures or sometimes the doctor may overestimate or underestimate the surgery depending on the number of appointments they have scheduled on that specific day.

Regarding the EHR sample means method also used by CUF, the healthcare provider whose data was analysed in this study, it allows predicting surgical time based on the average of historical data from a specific procedure or surgeon. However, this type of approach does not take into consideration other factors, such as patient and procedure-specific information, which can influence up to 30% of the total surgery duration [6]. Tuwatananurak et al. [6] used Leap Rail engine to show how can a machine learning algorithm improves the EHR predictions, getting a significant reduction of around 70% in the total scheduling inaccuracy, improving the estimation in approximately about 7 minutes per case regarding actual case duration. Moreover, in Rozario and Rozario [7] work the baseline time prediction was the surgeon's average procedure time of the last 10 cases. However, with the current method, case times follow a Gaussian distribution with an underestimation in 50% of the cases.

As these modest results evidence the challenging nature of the problem, they also encourage a ma-

chine learning approach, given the excellent results that machine learning methods have provided in natural language understanding [8], computer vision [9] or games [10]. For these reasons, the methods used today are seen by healthcare management as not effective, not allowing the most efficient use of surgery rooms. Thus, machine learning optimization methods that handle the information already available and recorded in hospitals have the potential to accurately predict future outcomes.

1.3.3 Related Work

The need for efficiency in planning and scheduling procedures has led to an increase in research in OR related problems since 2000, with a significant increase in publications since then [11]. In addition, since 2015, there has been an exponential growth in research in terms of the application of ML in the scope of medicine, since the availability of big data and the growth of data science have allowed a positive contribution to the decision-making processes [2].

Firstly, statistical analysis of the variability of surgical durations has been studied for years [12], and techniques such as Lognormal Estimation and Bayesian statistical techniques were intensively explored. These approaches find the best fit in a family of distributions to predict surgical durations and characterize relationships between variables. Stepaniak et al. [13] fitted a 3-parameter lognormal model that improved the OR scheduling and reduced the mean over reserved OR time per case by up to 11.9 minutes. Strum et al. in two studies [14, 15] compared the modeling of surgical procedure times with normal and lognormal distributions and concluded that lognormal models provide accurate predictions and fit better procedure times.

Moreover, models based on Gaussian Mixture Model (GMM) are also widely applied as a prediction model, even in the surgical area i.e. support patient flow models [16]. The Bayesian method obtained by Dexter and Ledolter [17] allowed improving predictions for cases where few or no historical data exist and concluded that GMM can be a reasonable choice when surgical times do not follow a lognormal distribution. Taaffe et al. [12] also studied the application of Kernel Density Estimation (KDE) to model surgical durations. The results outperformed traditional methods such as lognormal and GMM when there is limited historical data.

Other studies also investigate the potential of using mathematical models to improve durations, showing an OR efficiency improvement by combining advanced mathematical and financial techniques [18] [19]. However, these approaches postulate a simplified model for the data distribution and this thesis takes a data-driven, machine learning approach, while keeping interpretability as a requirement. Although machine learning and statistics are closely related fields in terms of methods, their main goal is different. Lee and Yoon [20] summarized the differences between classical statistical analysis and big data medical analysis. While ML models are designed to make the most accurate predictions possible and find patterns in the data that can be generalized, statistical models are designed for inference about the relationships between variables and reach conclusions about populations or derive scientific insights from data. Thus, in ML, the algorithm learns from a considerable amount of data and generates the hypothesis from the data, while in statistical models, we need to commit on *a priori* assumptions based

on various underlying probability distribution functions [2].

Even in the machine learning field, the high complexity of the OR environment allows and leads to different approaches to the problem and the use of different metrics by authors and researchers. Fairley et al. [21] defined as objective the minimization of maximum PACU occupancy, using constraints to control and maintain OR utilization. Thereby, to predict PACU recovery times for each patient, a gradient boosting tree model was used, which is used as input in a program that formulates the schedule of procedures in the operating room. Abedini et al. [4] developed a blocking minimization model to reduce the number of blockings between OR and PACU, allowing the hospital to define the OR schedule for the next day, considering the current stage occupancy of the OR, in order to to ensure the availability of downstream resources, such as beds in PACU and Intensive Care Unit (ICU).

The case duration accuracy is one of the most common approaches since to allocate the staff and maximize the use of OR accurately, it is important to predict the time required for each surgery with the smallest possible error. Bartek et al. [1] developed a linear regression and two ML models to predict OR case-time duration, with the XGBoost [22] attaining the best performance. Besides these, service-specific and surgeon-specific models were considered, where each speciality and doctor were modeled individually. Tuwatananurak et al. [6] compared the duration of the predicted cases from the conventional method based on averaged historical means for case duration with cases duration predicted by the Leap Rail engine, a proprietary algorithm that combines different supervised learning algorithms. Rozario and Rozario [7] resorted to the Operations Research Tools from Google Artificial Intelligence (AI), an open-source software suite for optimization, and developed an algorithm to optimize efficiency in OR in the era of COVID-19 with the objective of minimizing overtime and undertime cases in an OR that has shown to be beneficial to reduce the long waiting lists generated during this period.

Regarding machine learning-based solutions proposed to accurately predict surgical durations, Martinez et al. [23] compared Linear Regression, Support Vector Machines, Regression Trees, and Bagged Trees. In general, the methods considered are beneficial for operating room scheduling, but Bagged Trees was the one that achieved the best overall performance to predict the surgical time duration. Furthermore, Hosseini et al. [24] developed a classical Least-squares Linear Regression (LIN) and a Stepwise regression (STEP), showing both improvements compared to traditional methods. Lastly, Edelman et al. [25] performed linear regression models with data from six academic hospitals. Even with few variables, all are highly significant predictors and models presented a low error.

Researchers frequently use the approaches described above, however, other metrics can also be used with the goal of optimizing the operating room management. Lee et al. [3] performed an OR's efficiency review and mentioned methods such as identifying surgeries with high risk of cancellation and optimizing the turnover time between surgeries as frequent metrics used to evaluate and improve efficiency. Furthermore, Bellini et al. [2] presented a systematic review about the AI implementation in ORs where the majority of the studies use supervised learning techniques, being more frequently used random forest and decision trees algorithms. Decision trees are powerful and intuitive data structures, and because they are easily interpretable, they are widely used in the context of medicine, where it is essential to explain the predictions of the model, something difficult in ML because most predictive

models are complex and challenging to interpret.

Moreover, several researchers address the features used as inputs in their optimization models. Bartek et al. [1] took greater account of procedures and personal data to the detriment of the patient's health status and describes the primary surgeon as the most important feature to create variability. Fairley et al. [21] used a set of 10 features chosen based on discussions with health professionals, such as surgical service, patient information and the hospital unit the patient will go to after PACU recovery, where the most important feature was the procedure type with 0.41 of weight within the total of features. Tuwatananurak et al. [6] took into consideration more than 1,500 features, factors related to patients, providers, facility/room, procedures and prior events. Lastly, Rozario and Rozario [7] addressed that the machine learning algorithm held features such as frequency and distribution of procedure types, average case times and case times variability, highlighting the importance of the development of surgeon-specific models due to the variability that this feature can generate.

Unlike low-stakes applications, in decision-making and particularly in healthcare, black-box methods that output pure predictions without any verifiable explanation are not acceptable. Thus, the focus of this thesis is on interpretable machine learning models.

1.4 Thesis Outline

The thesis comprises a total of six chapters, and in detail it has the following structure.

Chapter 1 includes an overview of related work, OR organization and current standards of estimation. In Chapter 2 we introduce relevant topics for the full comprehension of the work. In Chapter 3, we present one of the most important steps for choosing the statistical model. The Exploratory Data Analysis (EDA) of available datasets allows us to analyze a massive dataset, correct errors and maximize insight into the data to extract important data characteristics. In addition, we describe the feature engineering and feature selection process, transforming raw data into features suitable for modeling and choosing non-redundant and relevant features to use in model construction.

In Chapter 4 we present and discuss the results of model approaches after applying three different ML algorithms to compare the applications of white box decision systems with black-box systems and current standards. Chapter 5 describes the application of GMM strategy used to deal with imbalanced data and presents a novel curve developed to have the model selection function.

To conclude, Chapter 6 introduces the proposed model and analyzes the results from an operational perspective comparing costs with standard performance. The chapter also presents the main conclusions of the research work and explores some possibilities regarding future work.

Chapter 2

Theoretical Background

In this Chapter, several theoretical concepts applied throughout the thesis are covered in order to demonstrate an understanding of the theories relevant to the topic and facilitate full third-party compression.

2.1 Supervised Learning

In the machine learning world, the two common machine learning tasks are supervised and unsupervised learning. What will lead us to use one or the other is the type of data we have to develop and train the model. In general, data can be described as labeled or unlabeled, in other words, it may or may not contain the solution we intend to reach. Our work, since we know the target, will be focused on supervised learning. To deal with imbalance, we will use unsupervised techniques like the GMM to learn the distribution of the scarce regions of our data and sample from it to rebalance the learning data.

Thereby, through the work we will design algorithms to learn by example and training with labeled data that will map the inputs in order to predict outcomes for unseen data and solve our machine learning-based problem.

Regarding labels, these can be categorical or continuous, leading us to perform a classification task or a regression task, respectively. In classification, algorithms work with discrete values and models are trained to categorize data into different classes. In the case of regression, we sought to find the relationship between the features in order to predict continuous output variables such as predicting house prices based on relevant information e.g., location, area and number of rooms. As the point of our work will be to predict surgical times and these are continuous, we will be working with supervised learning methods using regression.

2.2 Regression Algorithms

2.2.1 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is an ensemble learning method that combines the outputs from individual trees called ”weak learners”, modest models that performed slightly better than random chance. XGBoost is a gradient boosting framework developed by Chen and Guestrin [22] and a supervised learning technique that can be used for regression or classification tasks.

First, we introduce some notions of ensemble learning such as bagging and boosting to better understand gradient boosting. Bagging uses parallel training in multiple independent models to combine them and takes the average of the models' responses for regression tasks. In contrast, the boosting technique allows sequential training and the generation of ”weak learners” sequentially to correct the error of the previous one until no further improvements can be made. This method combines several models into one giving more weight to the models that perform better.

Finally, gradient boosting is a re-definition of boosting where the objective is to minimize a loss function. This function measures how close the predicted value is to the actual values, using a gradient descendent algorithm and finding the direction in which the loss decreases the fastest. The loss is the combination between the target value and the predicted value in order to figure out patterns on residuals. In this method, ”weak learners” have the same weight to the final prediction given by the learning rate, parameters that range between 0 and 1.

Therefore, XGBoost is a gradient boosting algorithm that uses decision trees, that combines simple decision rules, as its ”weak” prediction to predict a target variable accurately. XGBoost minimizes the objective function with Lasso (L1) and Ridge (L2) regularization to prevent overfitting penalizing model complexity. Thus, during training, the algorithm will iteratively generate decision trees to predict the residual errors of previous trees, and then combine the result with the generated trees in order to get the final prediction.

XGBoost is described as an algorithm that can model very complex relationships and it is popular by its performance and speed. This computational performance is enhanced by the algorithm design since it is able to use hardware resources efficiently and due to the possibility of the user specifying the maximum depth parameter, *max_depth*, when using the method, allowing to avoid unnecessary time pruning trees backward. An advantage of this machine learning model is its effective ability to achieve great results even with imbalanced datasets with skewed distributions because the algorithm is able to adjust the training to give more attention to minority class misclassification.

2.3 Interpretable Methods

2.3.1 RuleFit

RuleFit [26] is an algorithm that combines tree ensembles and linear models to take advantage of tree ensemble's accuracy and linear models interpretability. This algorithm allows us to generate rules from

a decision tree that create a set of new ”features” from interactions between the original features.

A tree-based model such as Random Forest or Gradient Boosting machine model can feed the RuleFit model and train the model using the dataset. The difference between the two lies in the way the trees are built. Gradient Boosting, as described in the XGBoost Section, builds trees one at a time, where each new tree helps correct mistakes made by previously trained trees, while Random Forest trains each tree independently with random sets in order to build more robust models and less likely to overfit on the training data.

From the generated decision tree, hundreds of rule combinations are generated, where each path can be converted into a decision rule through the combination of splits and therefore, depending on the depth and number of leaves, many rules can be generated that make it difficult to interpret and explain the model.

To circumvent the increase in dimensionality, Lasso, the L1 regularization technique, is called to assign weights to each decision rule since the current implementation of RuleFit can produce redundant features. By assigning a coefficient of 1 or 0 to the rules, Lasso will shrink the less important feature's coefficient and transform the input feature space into a smaller subset and easier to explain.

RuleFit is an interesting algorithm to apply in nonlinear problems since the generation of candidate rules from a combination of a tree model and Lasso regressor may help us better interpret predictions. Furthermore, it is a white box algorithm which is crucial from the point of view of the user and end consumer. Through relevant explanations directly taken from the model, we can increase the user's confidence in the model, and at the same time, understand if the model may be making illogical decisions or if it is unintentionally biased [27]. The set of rules generated should meet the insights returned from the exploratory analysis of the data.

2.4 Feedforward Neural Network

The Feedforward Neural Network (FNN) is a set of structured neurons in a series of layers, with each neuron in a layer containing weights to all neurons in the previous layer. The FNN goal is approximate some function f^* with succesive compositions of linear and nonlinear operators on x. The name ”Feedforward” is derived from the assumption that inputs and outputs are independent of each other and the corresponding decision that there are no feedback connections in which outputs of the model are feedback into itself [28].

The model is associated with a directed acyclic graph and represented by a combination of many layers of perceptrons. The first layer is the input layer and the rightmost is the output layer. Between them, there are a set of hidden layers with hidden units associated with often a nonlinear activation function to preserve many of the properties that make linear models generalize well. In FNN the piecewise linear function, Rectified Linear Unit (ReLU), is the recommended activation function represented with the formula $f(x) = \max\{0, x\}$ and shown in Figure 2.1.

The ability of the ReLU function set to zero values lower than zero, ensuring that the function is linear for values greater than zero, brings many advantages to the backpropagation process and the use of

gradient-based methods. Additionally, some hidden units have some points that are not differentiable, but with ReLU the derivative becomes 0 on the left side of x=0 and 1 on the right side. A drawback is that ReLU is not differentiable at zero. Nevertheless, it is differentiable almost everywhere, as the set of non-differentiable points has measure zero. Therefore, in practice, it is relatively rare to have a zero as the input to the ReLU.

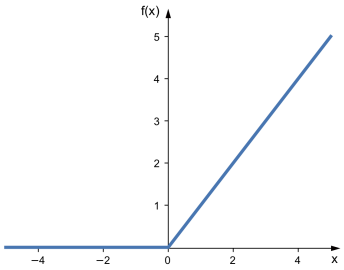


Figure 2.1: Line plot of Rectified Linear Activation Function for negative and positive inputs.

Backpropagation of the gradients allows the network to efficiently compute the cost function gradient to be used in the optimization algorithm, e.g., Stochastic Gradient Descendent (SGD).

To optimize the Neural Network (NN) cost or loss, SGD and Adaptive Moment Estimation (ADAM) [29] are gradient-based optimization algorithms commonly used. SGD is generally a little noisier because it takes small steps in a noisy direction of a minimum and is influenced by every set of samples. The optimizer updates weights after seeing a small subset of data or mini batch, instead of computing the gradient of the cost function for the whole training set.

ADAM has the advantage of being an algorithm that computes adaptive learning rates for each parameter and also adds the expected value of past gradients. The speed and faster convergence make ADAM a very interesting optimizer to use, being robust and suitable for a wide range of non-convex optimization problems in field machine learning as described by Kingma and Ba [29].

In FNN, it is common to normalize inputs to concentrate the spread of the data for the features in a smaller region to facilitate learning in the backpropagation phase. A significant difference between input features would generate large weights and, respectively, large updates, which would create greater instability and cause greater difficulties during training. Another issue relates with the saturation of dead zones of activation functions when gradients are zero. For these reasons, it is essential to normalize the features before introducing them into the model and after each activation layer. Thus, to keep all activation values on the same scale, we use a batch normalization layer to help us to get a faster convergence of the learning algorithm.

Lastly, dropout layers can be used to reduce model overfit and generalization error. Dropout is a regularization technique that allows us to train the network with random configurations, where we can drop some nodes at random during each training stage and learn redundant information pathways.

2.5 Categorical Data Encoders

Most machine learning algorithms have trouble handling categorical variables as inputs and require encoding them as real continuous variables. It is common to convert categorical features into numerical ones before fitting the data. Thereby, the three most popular encoding techniques are ordinal encoding, one-hot encoding and dummy encoding.

Ordinal encoding converts variables in ordinal ones, retaining order and for that reason end up to rearrange variables based on ranks. Suppose a ”Nucleotides” column with the four nucleotides types (Adenine (A), thymine (T), cytosine (C) and guanine (G)) found in DNA, after the implementation of the ordinal technique, the nucleotides will be converted into 0, 1, 2 and 3 respectively. Thereby, the encoding enforce ordered output and ”A” will be considered lower than a ”T”, which is lower than a ”C”, which is lower than a ”G”. Thus, afterwards it is relevant to comprehend if the ordinal relationship between the inputs that ordinal encoding will preserve is interesting for the dataset.

On the other hand, one-hot encoding will encode nominal features and generate a feature column per each variable. Each category value of the feature will be mapped into a binary column with 1's and 0's, where 1 represents the presence of that specific category. Thus, if we have a feature with five possible categorical values, one-hot encoding will generate five new columns and drop out the original one. However, this encoding will significantly increase the cardinality of the problem. If we applied one-hot encoding to columns like the first procedure, specialties or doctors, because these are feature variables with multiple categories, the encoding would create very high dimensionality that become problematic.

Regarding dummy encoding, this categorical encoding method is very similar to one-hot encoding and transforms variables into a set of binary variables. However, while in one hot encoding N variables are created to represent N values of one categorical variable, with dummy encoding, there is a slight upgrade and it can represent the same N labels in N-1 variables.

Lastly, one method that can help us with the high dimensionality is target encoding [30]. This encoder replaces categorical values with the mean of the target variable, so it picks up values that can explain the target. This encoding is a Bayesian encoding technique since it replaces each category with the posterior probability of the target and should be used with great care in order to minimize leakage.

2.6 Imbalanced Approaches

Imbalanced data is a common issue in learning problems mainly in classification problems where the ratios of each class are unbalanced and may lead the model to ignore minority classes. However, this problem is inherent in the real world as it is rare to have uniform distributions across several categories and we always end up observing skewed distributions in data labels.

As an example, we can think of a classification problem in which we want to predict whether a given person has cancer or not. In this problem, the dataset that we will have should contain much fewer instances of people classified with cancer than people without cancer. Therefore, we will have a disproportionate ratio between classes. Usually, the cost function aims to minimize overall error and maximize

classification accuracy, so with this disproportion of classes, our model will learn much better for majority classes. Consequently, if we do not correct this imbalance issue, we could get a model with high accuracy by being correct for most non-cancer instances without correctly predicting a single minority class instance.

To deal with imbalanced datasets, three data level methods are commonly used: Undersample majority class, oversample minority class, and generate synthetic samples.

Firstly, undersampling can be implemented by removing some instances from the majority class, however, it should be only used when we have a considerable amount of data because we do not want to remove valuable data.

Secondly, oversampling is related to increasing the number of samples in the minority class by adding random copies of the minority class in our dataset as if we increased the weight of these copied instances in the cost function. This method implies that some precautions have to be taken since the generation of copies can lead to overfitting and we need to ensure that copies are not from the test set in order to guarantee that there is no memorization by the model.

On the other hand, we can deal with imbalanced datasets creating synthetic samples through Gaussian Mixture Model (GMM) or Synthetic Minority Over-sampling Technique (SMOTE) [31]. These imbalanced solutions are going to be explored and therefore explained in a more concise way below.

Lastly, it is essential to note that we just should implement the imbalanced method after splitting the data into training and test to maintain the test set intact and ensure an adequate generalization error model in unseen data.

2.6.1 Synthetic Minority Over-sampling Technique

SMOTE uses the Euclidean distance between neighbors in the nearest neighbors algorithm to generate artificial minority class instances that will be available to train our model. First, SMOTE takes samples of feature space from the minority class and then the k-neighbors closest to the data are found. Afterwards, new instances will be randomly generated in space between target cases and their neighbors.

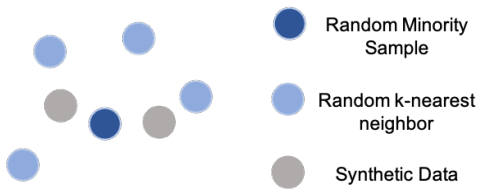


Figure 2.2: Synthetic samples creation process by SMOTE using k-nearest neighbor algorithm.

The algorithm helps to overcome the overfitting problem posed by oversampling however it has some drawbacks. As described by Chokwitthaya et al. [32], SMOTE cannot distinguish outliers from minority samples and it is limited in a line segment which is unreasonable for high dimensional data. Additionally, SMOTE does not check if neighboring examples are from other classes, so it may be introducing some noise in the dataset.

2.6.2 Gaussian Mixture Model

GMM is a powerful clustering and unsupervised classification method, defined as a convex combination of multiple Gaussian normal distributions, which has been proven to perform better than many other clustering methods as k-means or k-nearest neighbor [32].

The Gaussian Mixture is a function comprised of K Gaussians, where K represents the number of models. Each Gaussian identified by a $\kappa \in 1,..,K$ is represented by a mean, μ_κ, and a covariance, Σ_κ, the first one defining the center and the second one the spread and orientation of the cluster.

Although GMM are also applied in clustering tasks, this generative method learns complex data distributions from which we can sample synthetic data points in the high-dimensional feature space, instead of a linear sampling space [33]. In addition, GMM is also able to distinguish outliers from minority class instances and thereby it was the proposed framework implemented by Zhang and Yang [33] instead of SMOTE.

Type of covariances for the GMM

The covariance measures how much two random variables vary together and it is an important parameter that can be different along with GMM models. The covariance matrices vary between spherical, diagonal, full or tied, getting different performances according to how the data is adjusted. In Figure 2.3 is possible to observe the behavior of each covariance type in a generic example from *Scikit-Learn* [34]. The goal is to understand which covariance type in GMM best represents the three classes available.

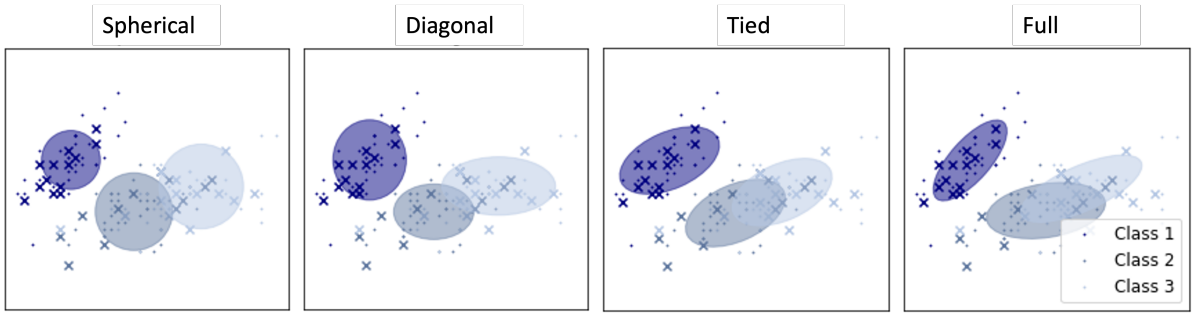


Figure 2.3: Different covariance structures for Gaussian Mixture Models. The example is a two-dimensional case with three different classes [34].

Firstly, diagonal covariance implies that each component has its own diagonal covariance matrix meaning different variances along the diagonal, therefore each component adopts an elliptical shape. This covariance implies D parameters per Gaussian, so a total of DK to be learned, where D is the number of dimensions. In contrast, in spherical covariance, a type of diagonal covariance, each element from the covariance matrix has its own single variance adopting a spherical shape. This covariance shape only require one parameter per Gaussian, so is less flexible. Regarding tied covariance, all components share the same general covariance matrix, thus each component shares the same shape and $\frac{D(D-1)}{2}$ parameters are needed to represent the model. Lastly, the more flexible covariance is the full because each component can adopt any shape or position in space. In terms of the covariance

matrix, each element has its own general matrix and is necessary $\frac{D(D-1)}{2}$ parameters for each Gaussian. Although this shape is more expressive, the more parameters, the more data is required for training.

Although it can be expected to achieve better results using full covariance, sometimes it tends to overfit with small datasets, and therefore it will be important to test always with all types before choosing a covariance matrix.

Number of Components

The number of components is related to the number of Gaussian models needed to fit our data. In order to get good synthetic samples, this parameter will have to be adjusted since too many components can generate overfitting but few may not represent the data structure well.

The optimal number of components will be chosen based on the Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC).

Probabilistic Model Selection

As mentioned previously, there are information criteria methods for determining the number of clusters that maximize efficiency while minimizing error, being these BIC and AIC. These information criteria allow controlling overfitting cases and attempt to correct the maximum likelihood bias, penalizing models with many components and ending up finding a reasonable optimal value for the number of components [35]. The lower the score value, the better the prediction of the GMM model. The formulas for BIC and AIC are shown in Equations (2.1) and (2.2) respectively.

Looking to formulas presented, where L is the likelihood, k is the number of adjustable parameters in the model and n is the samples size, we are able to understand that both scores add a penalty for additional parameters to maintain the balance between model performance and model complexity.

$$BIC = -2\log(L) + 2\kappa\log(n) \tag{2.1}$$

$$AIC = -2\log(L) + 2\kappa \tag{2.2}$$

Despite some subtle theoretical differences, their only difference in practice is the size of the penalty, in which BIC penalizes more heavily complex models. AIC has a higher probability of overfitting and selecting many parameters because it emphasizes model performance, but in contrast, BIC may choose a underfitted model and not being able to capture relevant variations.

Technically, the calculation of score curve gradient is also important to find the optimal model number of components since at a given point, the gradient will be practically constant and there is no advantage of increasing components number and computational time. Therefore, when it will be necessary to identify the optimal number of clusters for a given dataset, we will analyze both BIC and AIC metrics but also their gradients.

2.7 Regression Metrics

The evaluation of a machine learning algorithm is a crucial step during the machine learning process. After getting the predictions, we have to understand how close they are to the expected value and therefore, different metrics can be used. However, we will have to keep in mind that different metrics will lead to different results depending on our goal and data distribution, and that our model can get outstanding results on the training set, but behave poorly with the test set.

So, next, we will succinctly mention some regression metrics that will be used throughout the development of the models, either in the tuning step or in the final evaluation step.

2.7.1 Mean Squared Error (MSE)

MSE is a popular metric used to evaluate regression tasks, characterized by taking the mean of the square of the difference between the original values and the predicted values to obtain the final error. This metric is presented in the following Equation (2.3), where $\hat{y}_i$ represents the predicted value and y_i is the actual value. The MSE can be further decomposed in variance and squared bias of $\hat{y}$

Looking to Equation (2.3), more significant errors will be very expressive in the final calculation because the square has the effect of magnifying these errors, and therefore this metric will have a great focus on large errors.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.3}$$

2.7.2 Mean Absolute Percentage Error (MAPE)

MAPE represents the error normalized by the true observation value. This performance metric is asymmetric, being biased to under-predicted models over over-predicted ones, which may be interesting for our study. An under-forecast will never contribute more than 100%, as for example the limit case where $\hat{y}_i$ is 0 and y_i is 2, however the contribution of an over-forecast is unbounded below, as the case of $\hat{y}_i$= 6 and y_i= 2. Thus, the error imposes a higher penalty for negative errors and when the predicted value is higher than the actual.

Finally, the formula to this scale-independent metric is presented in Equation (2.4).

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{(y_i - \hat{y}_i)}{y_i}\right| \tag{2.4}$$

2.7.3 Utility-Based Regression

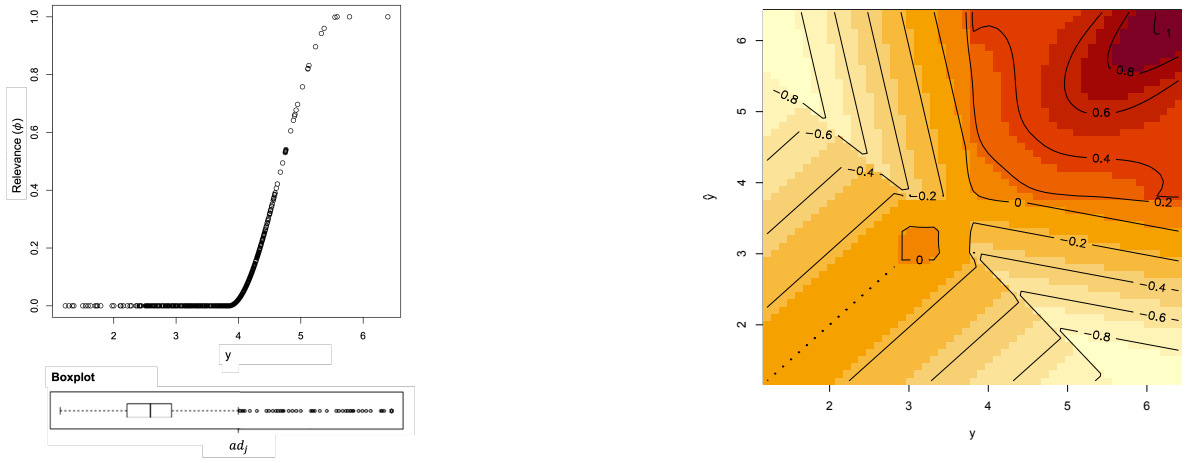
The performance evaluation sometimes may require the use of special metrics as the most popular metrics are based on averages and are not prepared for unbalanced domains [36]. To address regression problems where extreme values are also important to predict accurately and where we can focus on key application cases, Torgo and Ribeiro [37] developed a regression algorithm in the non-uniform costs

domain, which allows user to specify domain preferences and it also includes utility-based performance metrics, precision and recall metrics, often used in classification, but to be applied in regression tasks.

The package provides various pre-processing functions to deal with classification and regression problems and involves evaluating the utility (cost/benefit) of predictions. Nowadays, traditional formulas assume that all errors are of equal importance, however, this is not generally the case. So, to answer cost-sensitive problems, this metric assigns an utility score to any prediction based on the relevance of both the true and predicted values and on the loss of the prediction [38].

Before delving into both metrics, it is important to address the concept of Utility-Based Learning (UBL), the package that will be also used in this section and available in the R programming language [39]. Firstly, there is a relevance function, $\phi(.)$, which expresses the target variable in original domain into a continuous scale of relevance ( ]-∞,∞[$\rightarrow$ [0,1]) [37]. This function allows a specification of different degrees of relevance where 1 identifies the most important value.

The responsibility of defining the relevance function is on the user. One can use a method named "range" where the user manually defines the most important regions, or an automatic method named "extremes" based on the box plot statistic of the target for extreme values. This last method is the one that will be used throughout the thesis since the method assigns larger importance to the least represented target values of the dataset and so we do not have to worry about interpolating the points. In the "extreme" method, the score distribution of the final target values assigns more importance to the most extreme values, which depending on the type of extreme that we choose (low, high or both types of extremes) will generate a different $\phi(.)$ [40]. As represented in Figure 2.4 a), only developed for the explanatory purpose, we selected the "extreme" method and "high" type, thus samples with target values higher than 4, above the adjacent value (adj_H), will be more relevant than lower values. A single sigmoid defines the relevance function.



(a) Relevance function ϕ.

(b) Utility Surface.

Figure 2.4: Relevance function using "extremes" method and its utility surface. The relevance is associated with extreme and rare values. This function allows the specification of the target variable into a continuous scale of relevance, advantageous in terms of sensibility concerning the different values of the target variable. The utility surface is a function that maps the prediction value, $\hat{y}$, for the true value y into a utility score.

After defining the relevance function, we will be able to develop the utility function, $U(.)$, through methods of spatial interpolation of points. This function generates a surface nominated utility surface which maps the predicted value, $\hat{y}_i$, for the actual value, y_i, into a utility score. The score ranges from -1 to 1 and will be meaningful for the metric evaluation. If a data point has a utility score of 1, the point is of great importance and its predicted value is very close to the real one. In contrast, a utility score of -1 means that the point has a larger error and low relevance. In the following Figure 2.4 b) is possible to observe the utility surface that represents the previous relevance function.

As we can see in Figure 2.4 b), the utility surface generates a uniform-cost domain like we had a symmetry along the y = x plane. However, Ribeiro [38] developed a set of tools for regression algorithms in non-uniform cost domains with the inclusion of utility-based performance metrics.

Based on the relevance function $\phi(Y)$, a joint relevance function was developed (Equation (2.5)), which depends not on the relevance values from the pair $(\hat{y}, y)$ but also on a weight parameter ($p \in$ [0,1]).

$$\phi^p(\hat{y}, y) = (1-p)\phi(\hat{y}) + (p)\phi(y) \tag{2.5}$$

The weight parameter defines the importance given false alarms and missed values, where false alarms are events that are predicted in left upper corner of utility surface and missed values events predicted in lower right corner. Thus a $p = 0.5$ is equivalent to the previous situation where no cost-sensitivity exists. Additionally, as more weight is given to the real value relevance component in costs, false alarms, are even less punished and have less associated cost.

Moreover, the utility function is dependent on $\phi^p(\hat{y}, y)$ where Γ_B and Γ_c represent two bounded-loss functions with domain [0, ∞]$\rightarrow$[0,1]. As presented in Equation (2.6), the function in its limits is bounded by $\phi(y)$ when $\hat{y} = y$ and below by $U^p(\hat{y}, y) = p.(1-\phi(y)) - 1$ [40, 41].

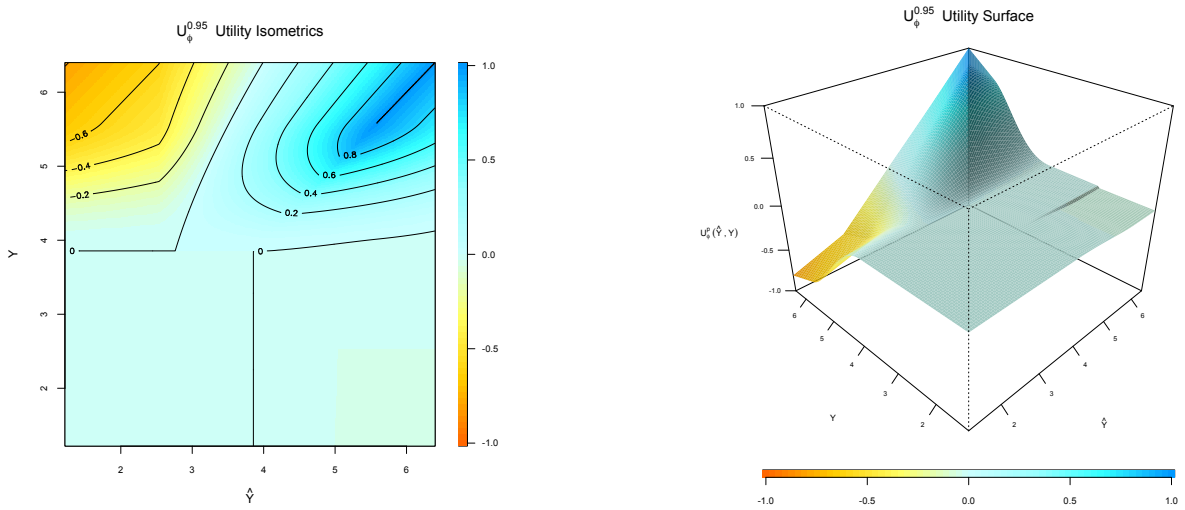
$$U^p(\hat{y}, y) = \phi(y).(1 - \Gamma_B(\hat{y}, y)) - \phi^p(\hat{y}, y).\Gamma_c \tag{2.6}$$

Figure 2.5 represents the utility surface and its utility isometrics defined for a $p = 0.95$. This value of p encodes that opportunity costs are considered more serious than false alarms. Thus, comparing Figure 2.5 a) with Figure 2.4 b), it is understandable than a p higher than 0.5 will exhibit higher costs associated with large errors about the relevant actual values, and the false alarms are not so relevant.

Precision and recall metrics are two of the most commonly used metrics in model evaluation in classification tasks and were originally defined by Kent et al. [42] in 1955. Precision expresses the proportion of data points that our model says are relevant and actually are relevant. At the same time, recall is the ability of a model to find all relevant cases within a dataset. The following equations (2.7) and (2.8) are the formulas to calculate these measures, where TP, FP and FN refer to true positives, false positives and false negatives respectively.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.7}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.8}$$



(a) Utility isometrics ϕ.

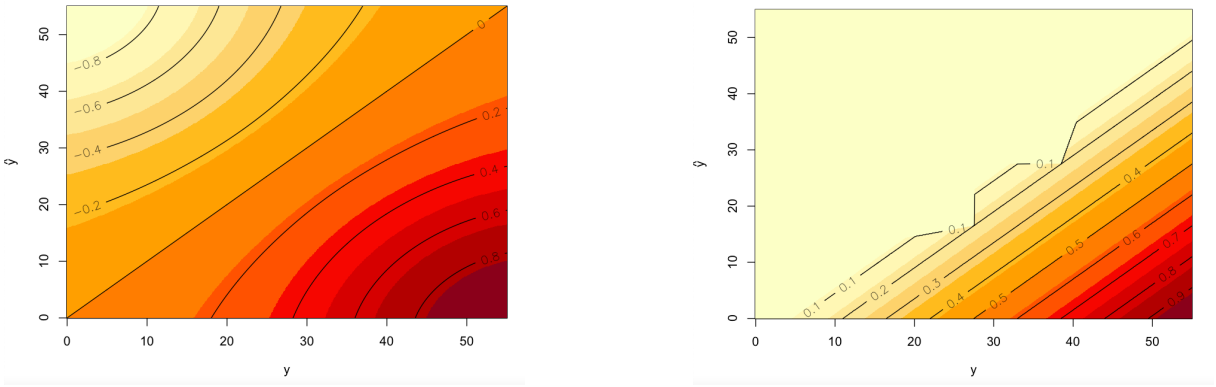
(b) Utility surface.

Figure 2.5: The utility surface obtained with the relevance function ϕ shown in Figure 2.4 a), with p = 0.95.

For obtaining precision and recall metrics for imbalanced regression tasks, two equivalent metrics were proposed by Torgo and Ribeiro [37] and Ribeiro [38] in order to adequately assess the performance of models in applications with non-uniform distributions of the target variable. Thus, for regression tasks, precision and recall will be expressed in terms of the utility obtained by the model rather than a hit/miss ratio as in the classification. Now, $precision^\phi$ and $recall^\phi$ metrics will be calculated considering the relevance function and its utility surface to respond to the problem of imbalanced data. The metrics will be used without the slight alteration in the formulation made later by Branco [43] since the package does not allow to control the p and it is more difficult to define the utility surface regions of interest. In Figure 2.6 we can see the utility surface generated by UBL when the type of surface is set to "cost" with splines and krige interpolation methods where the only difference is how the separation between the different utility levels is done. The "cost" parameter assumes that the diagonal of the surface where $\hat{y} = y$ is zero however it is also relevant that the diagonal has a high utility value, mainly for the region where there are fewer points.

First, a threshold, t_E, will be defined so that only points with a score above the threshold will be accepted for the metric evaluation. Thus, if we set the limit, for example, to 0.8, only will be evaluated data points with scores equal to or greater than 0.8.

The $recall^\phi$ (Equation (2.9)) defined as the proportion of relevant events that a model retrieves, in which relevant events are the ones $\phi(Y) \geq t_E$ and Y is the original domain of the target variable [37]. Basically, this metric is responsible for evaluating how well the y points that have high relevance are being estimated. In Equation (2.9) the value z can be 1 or 0 depending on $\phi(Y) \geq t_E$, so if the relevance function is greater than the domain-dependent threshold on relevance, the argument will be 1, if not is 0. The z_i is the true class and $\hat{z}_i$ is the predicted class.



(a) Utility Surface with splines interpolation method.

(b) Utility Surface with krige interpolation method.

Figure 2.6: The utility surface generated by UBL package when set the type of surface that is being interpolated as cost.

$$recall^{\phi} = \frac{\sum\limits_{i:\hat{z}_i=1,z_i=1}(1+U(\hat{y}_i,y_i))}{\sum\limits_{i:z_i=1}(1+\phi(y_i))} \tag{2.9}$$

Regarding $precision^{\phi}$ (Equation (2.10)), this is the proportion of the events retrieved by a model that are effective events, thus only takes into account the relevance of the predicted values, $\hat{y}_i$ [37].

$$precision^{\phi} = \frac{\sum\limits_{i:\hat{z}_i=1,z_i=1}(1+U(\hat{y}_i,y_i))}{\sum\limits_{i:\hat{z}_i=1,z_i=1}(1+\phi(y_i)) + \sum\limits_{i:\hat{z}_i=1,z_i=0}(2-p(1-\phi(y_i)))} \tag{2.10}$$

2.8 Interpretability and Model Explanations

2.8.1 Shapley Additive Explanations

Due to the complexity of machine learning models, sometimes it is not easy to interpret these models, which compromise their own application. Areas such as the health sector are linked to a high-risk industry, where it is imperative to understand and trust the decision-making process carried out by machine learning [44]. Keeping in mind this issue, the Shapley Additive Explanations (SHAP) [45] will be used in the context of tree ensembles like XGBoost, but also in cases of great complexity like Feedforward Neural Network to debug black-box models.

The framework proposed by Lundberg and Lee in 2017 allows us to train a ML model whose goal is to explain the contribution of features to individual predictions and understand which ones push more the output for a longer or shorter surgery duration. In the case of trees and ensembles of trees, the SHAP framework provides the implementation of Tree SHAP. For deep learning models provides an implementation of Deep SHAP to calculate SHAP values.

The calculation of SHAP values from the SHAP library is based on Shapley Values, a concept coming from cooperative game theory in Economics. Through this concept is possible to explain the difference between the actual prediction of output and the average prediction and comprehend the effect of each

feature taking into account the weight of all features.

2.8.2 Rashomon Curves

"Why would we trust the prediction of a machine learning algorithm over our own prediction?". When presented with a machine learning solution to a specific problem, this is a question frequently raised by stakeholders. From their point of view, it must be accessible to understand the model decisions so that we can also verify traits such as reliability and fairness [46]. Therefore, interpretability is important, mainly in areas where accuracy is crucial and the model will significantly impact.

Based on the need to obtain models that explain the output, but at the same time with sufficiently high accuracy and low generalization error on the test set, a diagnostic tool called the Rashomon Curve was proposed by Semenova et al. [47] to help answer this challenge.

Before introducing the Rashomon Curve concept, it is necessary to define some important terms.

- **Rashomon Effect**: As defined by Breiman [48] in 2001 this term describes problems where many accurate but different models exist to describe the same data.
- **Rashomon Set**: This is a subset of the entire hypothesis space of possible models in which the performance of the training set is close to the best model in the class. This is a set of almost equally accurate models for a given problem, so if the Rashmon Effect is large the Rashomon Set will contain a large number of models.
- **Rashomon Ratio**: The measure that will allow us to trade-off between simplicity and accuracy. This is calculated as the ratio between the volume of the set of accurate models and the volume of the hypothesis space.

Finally, the Rashomon Curve is a Γ-shaped curve formed as we increase the size of the hypothesis space. The curve represents the log of the Rashomon Ratio as a function of the empirical risk, where with the increase in the size of the hypothesis space, we will have a lower empirical risk for the training set.

The empirical risk is easily obtained because it corresponds directly to the loss, however the formulation of the Rashomon Ratio is more complex. The ratio varies always between 0 and 1 and, given a hypothesis space, represents the fraction of models that perform equally well when fit the data. The following Equation (2.11) shows the ratio calculation, which depends on the Rashomon parameter, θ, hypothesis space, $\mathscr{F}$ and the subspace of the hypothesis space, $\hat{R}_{set}$. The Rashomon ratio calculates the ratio of the volume ($\mathscr{V}$) of models inside the Rashomon set to the volume of models in the hypothesis space.

$$\hat{R}_{ratio}(\mathscr{F}, \theta) = \frac{\mathscr{V}(\hat{R}_{set}(\mathscr{F}, \theta)}{\mathscr{V}(\mathscr{F}, \theta)} \tag{2.11}$$

Semenova et al. [47] set the Rashomon parameter to 5%. Technically, all the models in the Rashomon set that have an empirical risk not more than $\hat{L}(\hat{f})$+θ, where $\hat{L}(\hat{f})$ is the lowest possible empirical risk across all algorithms, will be considered by the Rashomon volume. Thus, the Rashomon ratio is a ratio

of the Rashomon volume to the total number of models (hypothesis space).

Considering the different complexities of the models that can solve the ML problem under development, it is expected to obtain a Rashomon Curve similar to the one shown in Figure 2.7. Each hypothesis space is represented with a colored dot and the generalization error by an arrow.

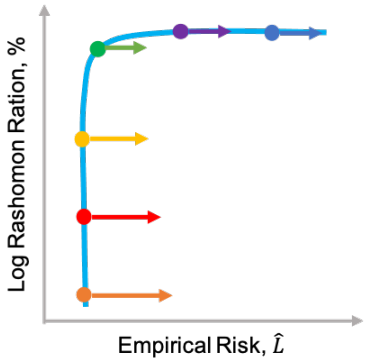


Figure 2.7: The Rashomon Curve illustrating the generalization ability of the Rashomon elbow and the empirical risk effect of increasing Rashomon Ratio.

Initially, for overly simple models, we will get a high error, however, with increasing model complexity, we will gradually reduce the associated error as it is possible to observe along the horizontal region with the decrease of empirical risk. It is important to mention that the Rashomon Ratio is almost constant in this zone since the Rashomon volume grows at about the same rate as the volume of all possible models.

Moving along the vertical part of the curve, models start to be too complex and the error is practically constant despite the increase in complexity.

Therefore, for the graph illustrated in Figure 2.7, among the hierarchy of model classes, the turning point named Rashomon Elbow is a good choice for model selection. This will be the sweet spot that allows us to have a balance between low empirical risk and a low complexity hypothesis space with desired properties such as generalization and interpretability [47].

Chapter 3

Data Analysis and Preparation

In this Chapter, firstly we introduce the datasets that are used to develop the model. Afterwards, these datasets are prepared and analyzed in order to summarize their main characteristics, a crucial initial step in data science.

3.1 Dataset Introduction

For the development of the time prediction model, historical data from CUF was studied. Four anonymized datasets corresponding to the years 2017, 2018, 2019 and 2020 were made available, even as a dataset with the description of all types of hospital procedures described in medical association, the official Portuguese Order of Physicians table (”Tabela Ordem dos Médicos (TOM)”).

Regarding the procedures dataset, this is a small dataset in terms of volume when compared with the historical data, which contains each procedure name and corresponding TOM code. This dataset does not require analysis since it is only used to verify that the procedures used in each surgery match with existing procedures codes in the TOM document, since there may be an error in data entry.

Historical datasets provide the surgeries that have been performed at CUF in the past four years, so each row represents an episode. For each surgery, relevant data related to unit, patient, doctor and surgery performed were made available. Concerning patient information, it is provided age, gender and encrypted CUF Identity Document (ID) (common to all units). About the surgery, data such as the surgical specialty, type of anesthesia, procedure types, the predicted and real used time inside OR and the recovery room time are given.

All datasets were made available in two formats (*.csv* and *.xls*), with the historical datasets together containing 191,046 rows, a value that represents the number of surgeries performed over the four years. To access and understand the information provided in historical datasets, a data dictionary is displayed in Appendix A. Latsly, the dataset of procedures contain a total of 2,832 procedure codes.

3.2 Exploratory Data Analysis

The EDA provides us with a fundamental insight into the dataset before starting to make assumptions. In this process, the main characteristics of the datasets are summarized, patterns are studied and clues are found to help formulate the assumptions and hypotheses for our model. In general, we are investigating the data, asking questions and looking for answers so relationships between features will become clearer. Based on a survey presented by Forbes, data scientists spend 80% of their time cleaning and organizing data, which highlights the importance of this step in the machine learning process [49].

To create the EDA, the two most commonly used data science tools are the language Python and R. For this analysis, we utilized Python and libraries such as Pandas, Matplotlib and NumPy. Furthermore, we concatenated the historical datasets of the last four years to avoid an extensive analysis. Only the most relevant information for interpreting the dataset is shown throughout this Chapter due to the order of magnitude of number of features that each dataset contains. The entire analysis is publicly available in [50].

CUF datasets come from an electronic medical records database, where patient information and important data from the operating room is recorded. Nurses register some records at the end of the surgery e.g., type of anesthesia or patient information, however, time records are automatically registered when certain buttons present in ORs are pressed by the doctor or nurse at certain times, such as when the patient enters or leaves the OR.

The dataset contains a total of 191,046 rows and 31 features, where each row is a data record of a surgery performed at CUF. These surgeries correspond to the activity of CUF at fifteen CUF units spread across Portugal with the distribution over the years present in Figure 3.1.

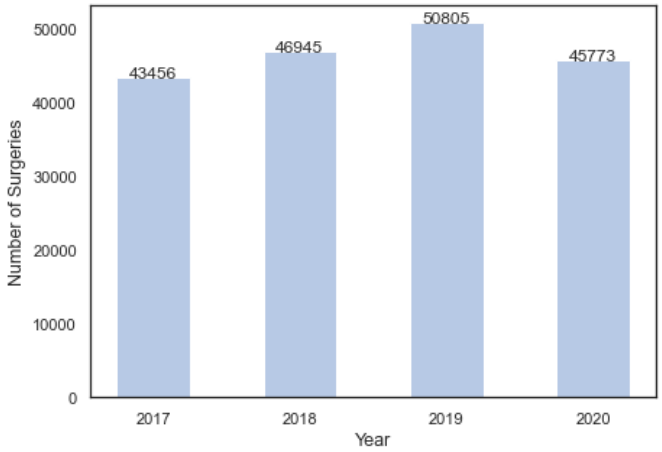


Figure 3.1: Number of surgeries per year since 2017 to 2020.

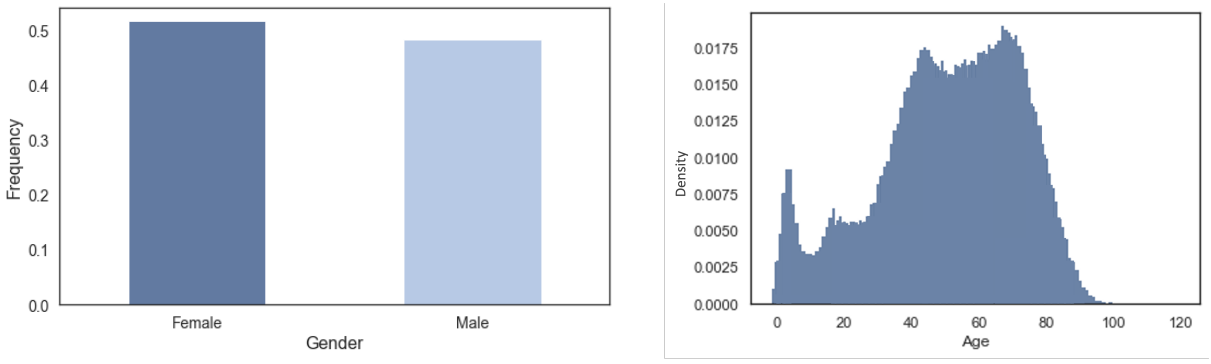
The units present in our dataset are categorical features represented by integer numbers. Each number is linked to a CUF unit where there are surgery services, however, we do not have access to the hospital name and its location for confidentiality reasons. The majority of surgeries, around 68%, are performed in CUF units 13, 14 and 15, that have the highest number of operating rooms, 21, 24 and 15 respectively.

Regarding patients, our dataset contains specific columns from patient data, such as their birthday date, gender, encrypted Local ID and Unique ID, Operative Registry Number (ORN) and the status of the patient (Outpatient or Inpatient). Outpatient care defines a service that does not require any type of hospitalization. In contrast, inpatient care is related to patients who remain in the hospital for one or more nights.

First, the Local ID is a unique key to encode each patient in each unit, where for example within the same unit patient ”1234” will always have the same number, but patient ”1234” from another unit will be different. The Operative Registry Number is an internal unit code, however the same number can be present in different units, so to identify unmistakably we should use as a unique key for a patient's surgery a composite key between ORN and Unit. In addition, the Unique ID makes it possible to identify the number of surgeries that each patient underwent at all CUF units, and it was possible to observe that in the last four years, almost 80 % of patients performed only one surgery at CUF.

In Figure 3.2 is possible to observe the distribution of age and gender among patients, where the calculation of the patient's age at the time of surgery was possible using his date of birth and the date of surgery. For the case of gender, the proportion of male and female patients is almost 50/50. This observed relationship takes into account all surgeries, however, in some specialties such as gynecology, there is a predominant gender.

For the distribution of age is possible to visualize an interesting shape. Although specialties are only studied further in Figure 3.4, we analyzed that up to the age of 10, we observed a considerable number of surgeries with a particular incidence in otorhinolaryngology (ear, nose and throat) and obviously pediatric surgery. From the age group of 35 years old, we have once again a substantial increase in the number of surgeries performed, with orthopedics, general surgery and ophthalmology being the predominant specialties. Besides, it is interesting to note that the distribution shape of Figure 3.2 b) is consistent in each of the four years.



(a) Gender Distribution.

(b) Age Distribution.

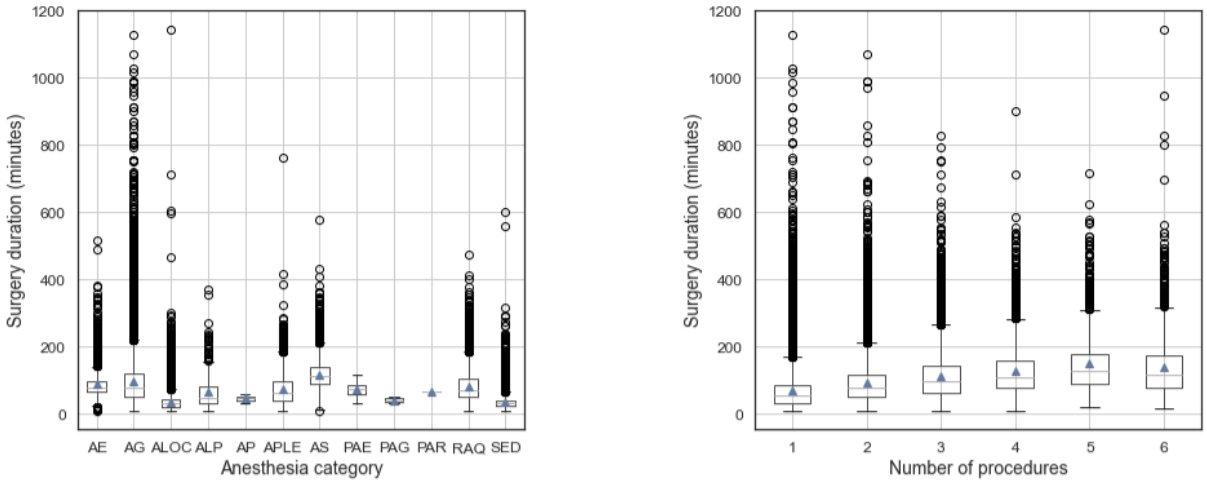
Figure 3.2: Gender and age distribution over the patients. The proportion of males and females is approximately the same. Underage patients represent 9.98% of the total surgeries.

Moreover, for each surgery itself, there is a variety of information provided by the CUF records, such as type of anesthesia, surgeon, procedures performed, surgery date and CUF unit. Each of these features may or may not be relevant to predicting surgical time, as will be determined by the EDA.

Regarding the type of anesthesia, it is noticeable by observing Figure 3.3 a) that specific categories

of anesthesia are associated with longer times of OR usage. Therefore, due to the apparent time differences between the anesthesia categories, we expect the anesthesia feature to be important for the final prediction.

For the same reason, the number of procedures is essential to estimate the final surgical time, since with the increase in the number of procedures, the average time within the OR is increasing. The results in Figure 3.3 b) are consistent with what we were expecting since for surgeries with several procedures some idle time may also be associated with procedures transition.



(a) Categories of anesthesia distribution. Nomenclature of anesthesia categories presented in Appendix A

(b) Number of Procedures distribution.

Figure 3.3: Anesthesia and procedures number distribution over surgeries. Each anesthesia type has its surgery duration distribution pointing to the importance of this feature. Surgeries with different numbers of procedures also present different spread out of data.

Concerning specialties, CUF's dataset covers 26 specialties, of which 25 are valid surgical specialties for further analysis. Administration request was excluded as it was incorrectly recorded as a specialty and therefore should not be considered. Thereby, in Table 3.4 is possible to look at the variety and number of surgeries performed over the four years in each specialty. Orthopedics, general surgery and ophthalmology are the specialties with more surgeries covering almost 50% of the total number of surgeries in ORs. However, although they are at the top in terms of the number of surgeries, obstetrics and gynecology is the one that contains more surgeons.

Furthermore, exploring the distribution of surgeries over the week and throughout the day is also important to confirm that the data meets our expectations. Thus, in Figure 3.5 is presented the distributions of elective and urgent surgeries for the situations mentioned above. Programmed or elective are surgeries scheduled in advance and urgent surgeries are considered emergent and must be performed as soon as possible.

Regarding distributions, during the week we have a greater occurrence of surgeries and they mostly occur between 9 am and 8 pm. Compared to elective surgeries, the proportion of urgent surgeries increases significantly on weekends and between 9 pm and 8 am, even representing more than 50% of all surgeries on Sundays and between 1 am and 4 am. These values meet our expectations since one of the main reasons for operating at night is the emergence of unavoidable cases. Therefore, surgeries

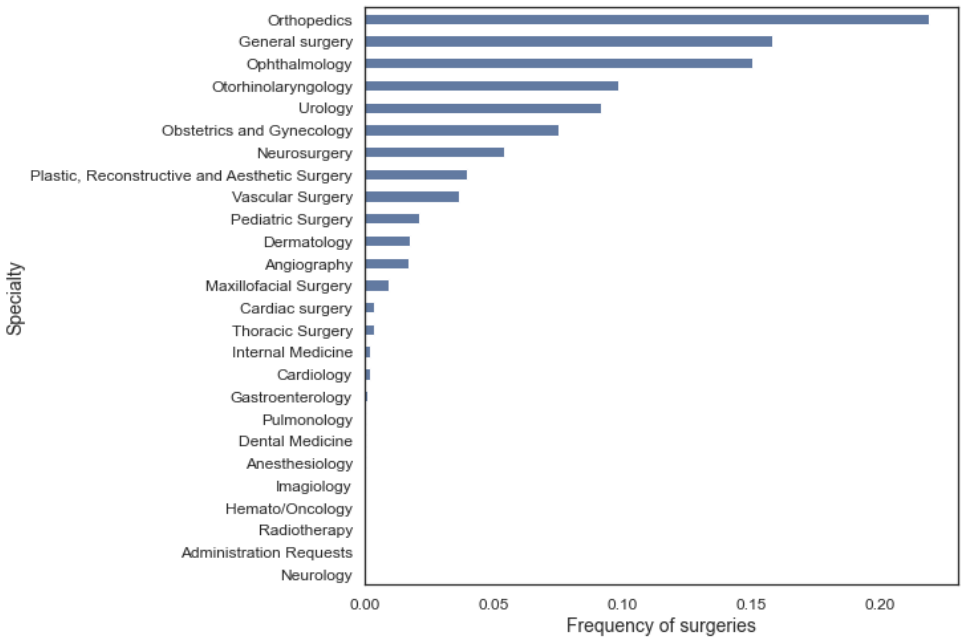
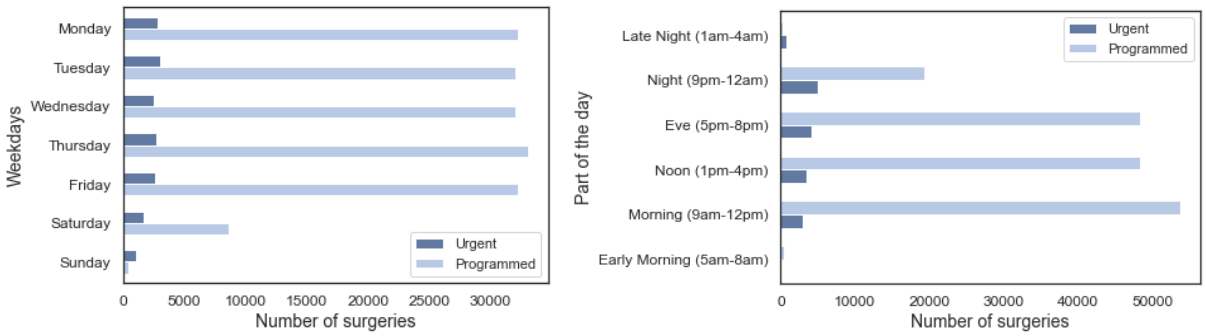


Figure 3.4: Frequency of surgeries per specialty. A small number of specialties accounts for most surgeries.



(a) Surgeries distribution over the week

(b) Surgeries distribution throughout the day

Figure 3.5: Anesthesia and procedures number distribution over surgeries. More urgent than elective surgeries on Sunday and between 1 am and 4 am.

at these hours are performed infrequently and predominantly in urgent cases.

Then, we explored the distribution of procedures and specialties throughout the week. Again, we performed a similar investigation over the different months and throughout the different parts of the day, but these results are not described here because of their similar patterns.

The objective with these distributions is discover patterns and relationships, and in the analysis of surgeries by weekday in Figure 3.6, understand the most performed surgeries throughout the week. In Figure 3.6, we can see, for example, that orthopedics is the most performed surgery type over the week days except for Thursday and general surgery is consistently in the batch of 3 specialties with more surgeries every day.

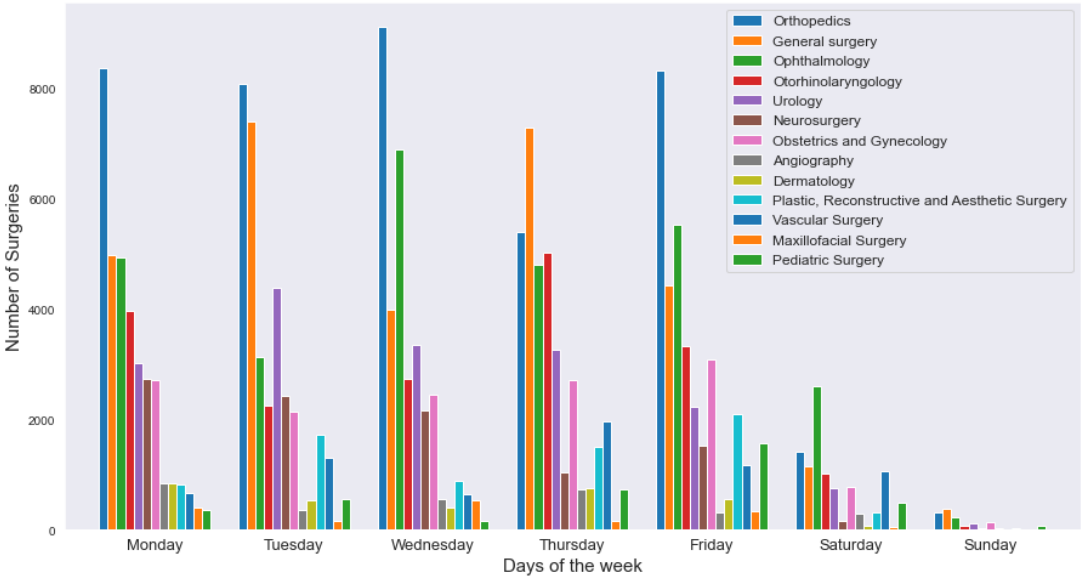


Figure 3.6: Week distribution of specialities. Orthopedics, general surgery and ophthalmology are the main specialties performed throughout the week.

Regarding Figure 3.7 is possible to verify a relationship between the procedures and the type of specialty. In this figure, procedure 33, described by the nomenclature as ”skeletal muscle system”, is the most frequently performed procedure throughout the week, presenting the same orthopedic pattern and clearly associated by name to orthopedic surgery. We observed the same on Wednesday, where we have ophthalmology as the second most performed type of surgery and associated with procedure 46, meaning ”eyes and ocular attachments”.

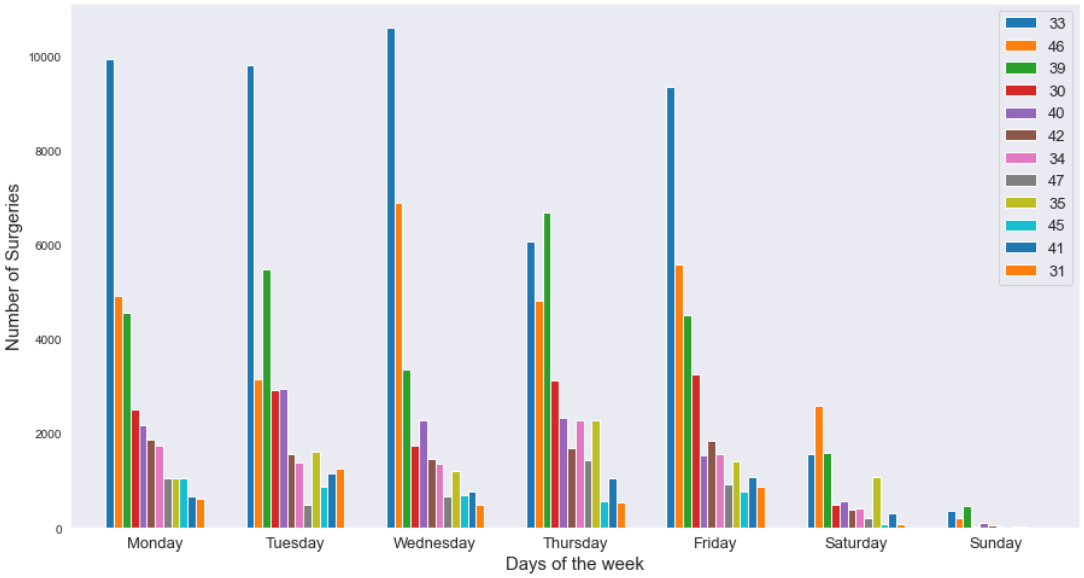


Figure 3.7: Week distribution of procedures. 33 (skeletal muscle system), 39 (digestive system) and 46 (eyes and ocular attachments) are the main procedures performed throughout the week.

3.3 Time Series Analysis

Time series is critical to understand how the data is distributed over time and look for patterns, such as trends and seasonality. Thereby, based on past data and its underlying structure, it is possible to predict future events' behavior.

Firstly, Figure 3.8 represents the monthly number of surgeries over the four years. As can be seen, August and December are the months in which we have a considerable reduction in the demand for surgeries, a fact consistent over the years and potentially related to summer, hospital staff vacation, and the end of the year. In addition, the reduction in demand for surgeries during the early phase of the COVID-19 pandemic is visible from April 2020, however, there is also a greater demand after the summer of 2020, probably related to the reduction of fear and demand for scheduling surgeries previously postponed, and therefore, in the annual total, there is no significant reduction in surgeries in 2020.

Moreover, an open-source software from Facebook, *Prophet*, was explored. The library was designed for forecasting time series data and, in addition to adapting to yearly, weekly and daily seasonality, it can also handle specific events, such as COVID-19 as a sporadic event, allowing users to adjust forecasts.

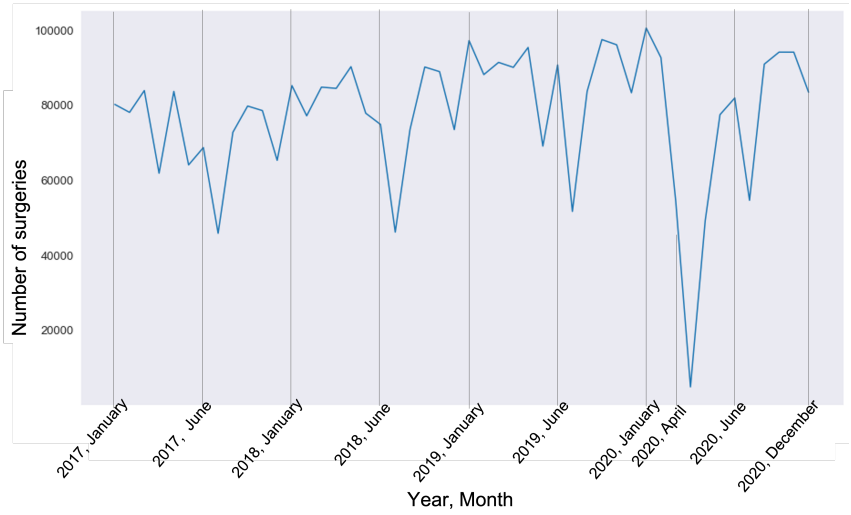


Figure 3.8: Monthly distribution of surgeries from 2017 to 2020.

Therefore, the following Figures 3.9 and 3.10 represent the yearly and monthly seasonality, and the trend, respectively. On these figures, the y-axis of each component represents the incremental effect on y, so the value related to a particular weekday means how much y is added to the final value due to weekly seasonality. Observing Equation (3.1) is noticeable that the tool uses a decomposed time series model with three main components: models trend ($g(t)$), models seasonality i.e yearly, weekly and daily ($s(t)$) and models the effects of holidays or large events ($h(t)$). Lastly, there is also an error term related to unusual changes. Hence, if Friday has the value of 40 in weekly seasonality, this indicates that for every Friday, 40 is added to the sum of all components.

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{3.1}$$

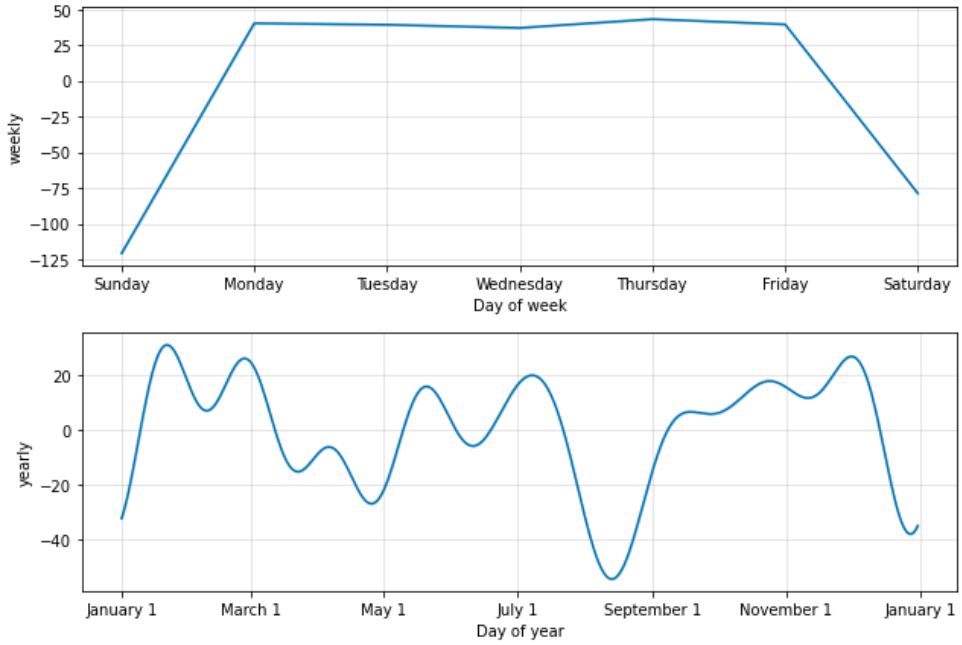


Figure 3.9: Yearly and monthly seasonality obtained through *Prophet* Library

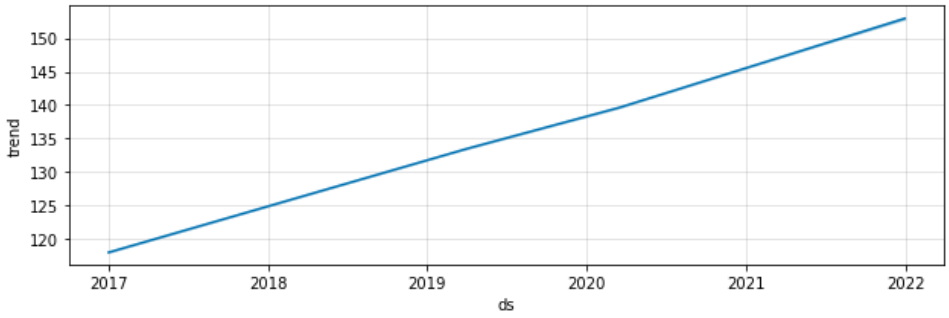


Figure 3.10: Trend resulting from the use of *Prophet* tool after the inclusion of an exceptional season (COVID-19).

These plots were obtained considering the COVID-19 pandemic between April and August as a sporadic event. As it is possible to observe in Figure 3.8, we chose this interval because it is the period where is noticeable a significant change about the same period in previous years.

From the weekly component, it is possible to observe the behavior of the number of surgeries throughout the week, where, as expected, it noticed a reduction in surgeries at weekends with a greater impact on Sunday. Regarding the monthly component, as seen earlier, in summer, between June and September, there is a reduction in the number of surgeries performed, and at the end of December and beginning of January, low demand is also evident.

Concluding, intending to develop machine learning methods and predict future events, the analysis of the data and the sequence in time are significantly relevant to be afterward possible to explain the obtained interpretable models. Thus, using the time series analysis, it is possible to understand the behavior of surgery throughout the year and from year to year.

3.4 Feature Engineering

Feature engineering is related to the good utilization of domain knowledge in order to ably transform raw data into new additional features that improve the performance of ML models. Throughout feature engineering, we can generate additional information that has the potential to boost model performance and develop a helpful predictor.

Some important features already mentioned before, such as age, month, weekday and part of the day, had been generated through these processes, but directly from date and surgery time columns. However, with the insights gained after investigating and exploring data, we find it interesting to develop some specific features.

Firstly, due to the potential difficulty using procedures columns (I1, I2, I3, I4, I5, I6) since most of the columns have a considerable number of missing values, a column with the total number of procedures is created. Each surgery can have 6 procedures involved during surgery, so we can have up to 6 TOM codes. The first procedure(I1) is usually associated with the main and general procedure, and the following ones are secondary and more specific procedures. Thus, if a surgery has I1, I2 and I3 values not null but I4, I5 and I6 with Not a Number (NaN) values, 3 will be the value present in the additional column.

Additionally, the doctors' daily capacity and the total number of surgeries performed by the doctor in CUF may have an impact on its performance and, therefore, in surgery duration. Thus, a column was created to reflect the surgery order on a specific day and for a given doctor, and another column to reflect the doctor's experience level. So, if a doctor has already performed five surgeries in a day, the subsequent surgery performed on that day will have the number six associated with this column.

Finally, a column with the total number of surgeries performed by the doctor in CUF so far is created. The goal is to attempt to identify if the doctor's experience level can cause an impact on surgery duration.

3.5 Data Imbalance

Throughout the EDA we noticed an unequal distribution of surgery times in the dataset, which could be more challenging to model and require specialized techniques. As we can see in the following histogram in Figure 3.11, the actual values are very funneled close to the mode (mode=30), presenting a left-skewed histogram.

By looking at the histogram in Figure 3.11, we notice that our model probably will predict longer times very poorly due to the lack of data in this region and therefore as machine learning models minimize the general error of the problem, the model could be focused on reducing the error for surgeries with short times because the majority of the labels are in the first region of the domain. Consequently, a data imbalance problem may arise. In this scenario, we can use synthetic data generation techniques to address data imbalance.

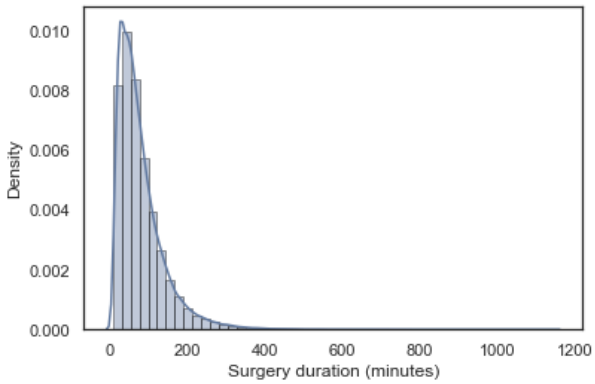


Figure 3.11: Histogram and density plot of target. Evidencing the data imbalance.

3.6 Missing Data

About missing data, the majority of machine learning algorithms do not support NaN values in the input data and, therefore, realizing how they are distributed in features and handling them is very relevant. So, with this analysis, we intended to understand if there may be a negative impact on removing surgeries with missing values since we do not want to lose a large amount of valuable data.

To visualize the distribution of missing values, understand the structure of the dataset and figure out correlations, Figure 3.12 below was plotted using the *Missingno* library. Here, all surgeries in the dataset are represented and each white line indicates an absent value in each column.

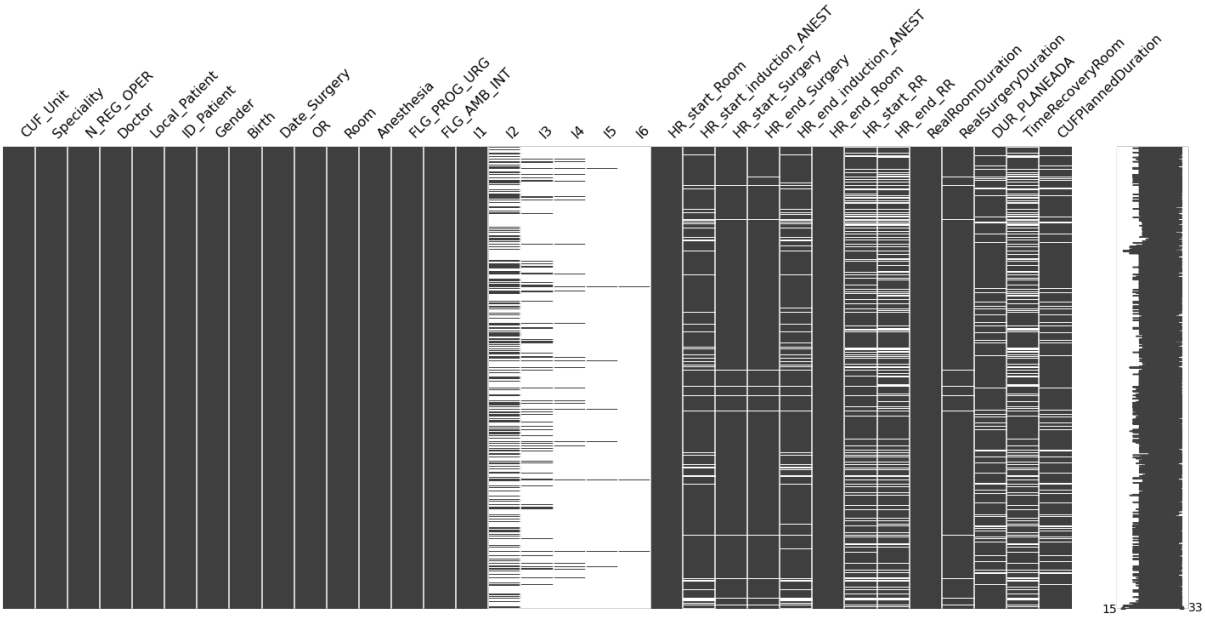


Figure 3.12: Visualization of missing values. The white lines represent the missing values in each column. Procedures columns present a high number of missing values except I1 column since each episode needs to have at least one procedure.

In the CUF dataset, most features have less than 2% of missing data, a substantially low percentage compared to the total number of surgeries presented. However, regarding procedures, the dataset

has six columns representing up to a maximum of six distinct procedures that can be performed during surgery, and most of them have a considerable number of NaN values. The high number of missing values happens because most surgeries have a unique procedure and only 1% of surgeries have six procedures.

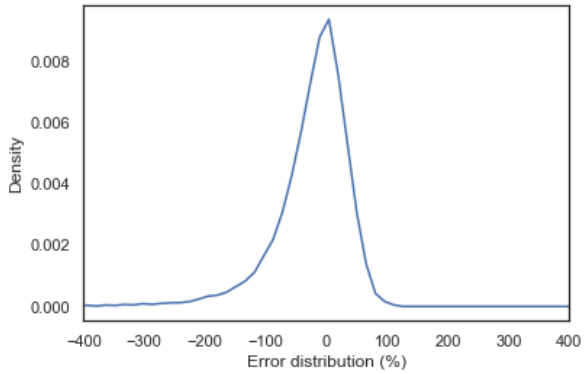
The remaining features have missing data ranging from 10% to 24%, and it is important to point out that for 11% of the data, we do not have the duration planned by CUF and 25% of patients have no time in the recovery room recorded.

3.7 CUF Predictions

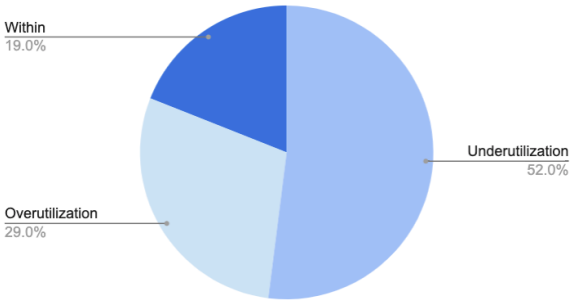
After extensive data analysis, CUF forecasts were compared with the actual surgery durations. So we can understand the distribution of the error and how close the predictions are to the recorded real time. This will be the benchmark for our work, where the focus will be on reducing this error and improving precision in the prediction of surgical times.

In Figure 3.13, it is possible to observe the error distribution of CUF predictions. This error was calculated considering the actual case-time duration and considering a tolerance threshold of 10%. Overutilization cases are those where the real duration exceeds the predicted time by greater than 10%, representing a positive error. In contrast, underutilization cases are those estimated 10% below real time, associated with an OR overestimation and a negative error. Lastly, within cases are considered when the prediction of surgery is within ±10% threshold.

Analysing plot shape, the error follows a Gaussian distribution with a mean of -27% and a standard deviation of 90%. Therefore, most surgeries take less time than scheduled and operating rooms end up not being used to their full potential. Furthermore, considering as well-planned surgeries those with a module error of less than or equal to 10%, 29% of surgeries are overestimated and 52% underestimated.



(a) Error distribution between planned and realized surgery durations.



(b) Overutilization and underutilization of operating rooms.

Figure 3.13: Error distribution since 2017 to 2020. A negative error is directly related to the underutilization of the room, meaning that CUF prediction is greater than the real time used. Positive errors are related to OR overutilization. Both Figures highlight the importance of a more effective method for predicting OR surgeries, as CUF tends to overestimate the case duration, correctly estimating only 19% of surgeries.

3.8 Remotion of erroneous data

Given the dataset from January 2017 to December 2020 with a total of 191,046 surgeries, data cleaning was performed to facilitate the use of subsequent machine learning models.

Firstly, duplicate surgeries were removed from the dataset, as well 64 instances associated with surgeon ID 0009, doctor number used by the CUF surgery scheduling team to test the platform. In addition, all procedures present in columns I1 to I6 were checked with the Order of Physicians table to remove any mistakenly introduced procedure that was not listed. In this case, the procedure was set to NaN.

During the data exploration, it was also possible to verify that certain patients contained duplicate registered surgeries that had not been removed. This happened because there were patients with the same surgery at the same time but associated with different units or different operative registry numbers. Thus, to remove the duplicated rows we checked the ones that contained the same patient ID, doctor, surgery date and surgery start time, since this set of variables must be unique throughout the dataset. Thereby, 4,067 surgeries were removed and the variable that was different within the duplicates was defined as NaN.

Moreover, there is a low relative frequency ($<$0.01%) of male patients associated with data entry mistakes in the gynecology specialty, so we removed these patients' surgeries. Lastly, within the 26 specialties present in our dataset, administration requests is not recognized as a specialty within surgeries. This may be an error introduced by the healthcare professional and therefore the associated lines have been deleted.

Thus, after data cleaning we have a total of 186,979 surgeries in our dataset.

3.9 Data and Features Selection

After the feature engineering process in which we generated more columns from the original data and the knowledge acquired in the exploration data analysis, our dataset contains a total of 41 columns. At the moment, it includes a lot of information, some of it redundant, so it is crucial to make the right choice on the data that we intend to use.

Hence, the columns that we consider relevant to train the models were selected. These include: specialty, CUF unit, anesthesia category, the total number of procedures performed, first procedure, surgeon, patient's gender and age, number of surgeries that a given doctor has performed so far, number of surgeries that the doctor has performed on that day, temporal data such as month, weekday and part of the day, actual duration of surgery, planned time by CUF, type of hospitalization (outpatient or inpatient surgery) and planned or urgent surgery. From these 17 columns, we removed all surgeries that contained surgeries with missing information. The column with the time planned by CUF is not an input to the models but is kept to compare the current methods used. The case was not taken to not allow for information leakage for this benchmark.

With this selection and removal of surgeries with missing values in at least one of these 17 columns, we obtained a total of 169,772 surgeries in our final dataset, which we consider a good starting point

for the development of the thesis. In Figure 3.14 we can note the cleaning process performed to ensure that the dataset is free of inaccurate or corrupt information and ready for use.

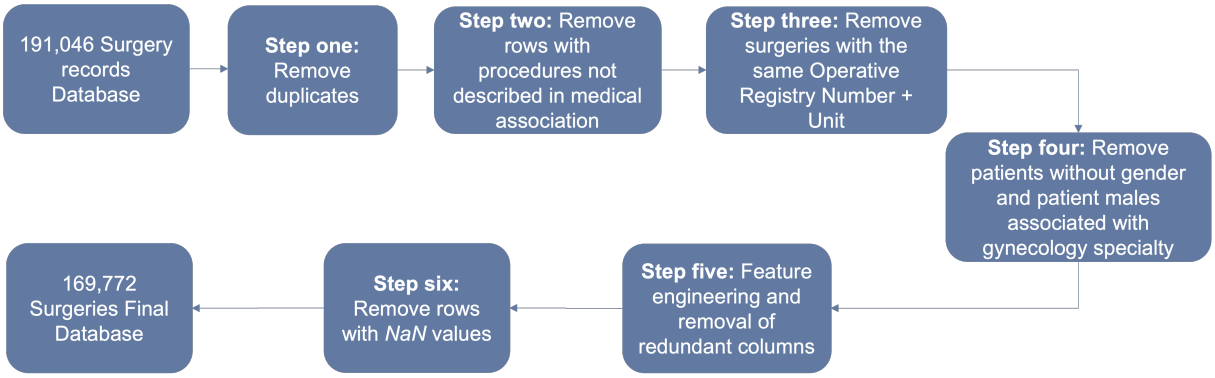


Figure 3.14: Summary of the data cleaning process. Flow diagram illustrating the data cleaning process to create the final dataset.

3.10 Encoding

The majority of the models work with numeric data as input, so it is important to understand if our data needs to be transformed to be compatible with a specific model type. As explained in Section 2.5, there is a wide variety of encoding methods and have into consideration the advantages and disadvantages of each method we want to convert categorical data into suitable numeric values.

Consequently, for the case of features with two values, which happen in 3 of the 17 features, we performed dummy encoding, an encoding similar to one-hot encoding in which only one column is kept to represent features such as gender, hospitalization and if the surgery is programmed. Thereby, one column with 1's and 0's is generated, meaning female or male respectively in case of gender feature.

Lastly, in remaining categorical features with more than two values, seven features in total, target encoding on features was performed. As discussed, although one-hot encoding may have better encoding than order encoding since sorting variables is not in our interest, it produces countless columns hindering the good performance of machine learning models.

With target encoding some considerations must be taken into account because we do not want target variable leakage in the new encoded feature. To prevent this problem, encoding should be applied after the split into training and test data because otherwise, our data could overfit and the results may not be reliable. Thus, to prevent this problem, target encoding utilizes training data to fit the encoder and transform the new categorical data in both training and test sets.

Chapter 4

Modeling

In this Chapter three different strategies have been implemented to our dataset. We applied the interpretable and explanatory model, RuleFit, and two opaque models, the gradient boosted decision trees, XGBoost, and a feedforward neural network to comprehend the behavior of CUF data with black-boxes and the trade-off between predictive power and interpretability. Since our objective will go through an explanatory model, RuleFit will be the main algorithm used as a starting point for the final development of the model.

4.1 Proposed Approaches

With all the data processed and ready to be used, we had the opportunity to understand with CUF our main focus and how we would approach and solve the problem. After analyzing similar works in the OR topic and considering the indication of CUF's stakeholders, the strategy chosen to move forward was the study of three types of approaches: a general, specialty-specific and surgeon-specific models. The last two specific models were based on the work developed by Bartek et al. [1], in which the authors generated specific models for surgeons and each specialty.

The models of each approach will contain a different structure, its specific encoding and different observations in the target column.

4.1.1 General Model

The general model is a model that receives as input all 17 variables defined previously. We intend to realize if a single model without being focused on a specialty or doctor can be good enough and even accurately assess the operating room durations or if it turns out not to be great because the problem is very complex and there is great data dispersion.

4.1.2 Specialty-specific Models

The specialty models are generated for each specialty present in the dataset using all available features except the specialty column. The specialty-specific models are developed for each surgical specialty with more than 100 surgeries in the training dataset to achieve a reasonable performance value in the test dataset. Accordingly, of the 25 specialties present in the dataset, 18 have specific models.

4.1.3 Surgeon-specific Models

Regarding surgeon models, these models use 16 features because the doctor column is removed. Surgeon-specific models are modeled individually and only those surgeons with more than 100 surgeries in the training dataset are considered. Therefore, 381 surgeon-specific models are trained and evaluated.

4.2 Data Split

Concerning the split, as throughout the thesis we are going to apply different methods to the data and in the end, we desire to obtain the algorithm generalization error, we divided the data into: training, validation and test datasets.

Technically, we require a training set to fit the model and a validation set to provide an unbiased final model evaluation. However, as these training and validation datasets are used across the different algorithms, once we have all final models defined, we will evaluate them all at once with the test dataset, which allows us to compute the generalization error. Therefore, we divided the model's data into 80% for training, 10% for validation and 10% for testing dataset.

Lastly, it is important to mention that intermediate validation sets that help us have an unbiased evaluation while tuning model hyperparameters are not defined at this stage because it is used differently depending on the algorithms. Most algorithms receive a validation parameter that corresponds to the percentage of the training we want to use as validation, so it is not necessary to define *a priori*.

4.3 Machine Learning Algorithms Implementation

4.3.1 Extreme Gradient Boosting Implementation

As a starting point of our work, the approach taken by Bartek et al. [1] was reproduced to understand what results we would obtain and whether our data would be able to achieve an improvement concerning the current CUF predictions. The paper used three different algorithms: linear regression, random forest and XGBoost, thus the one with the best results, XGBoost, was used to generate the first model. A decision-tree-based ensemble ML algorithm that uses a gradient boosting framework, one of the most powerful methods to produce predictive models. XGBoost framework was developed based on the idea of successive improvements of weak learners, where second-order gradients of the loss function are

computed to get more information about the direction of the minimum.

We performed a 5-fold cross-validation for each model to optimize the hyperparameters, so the training set is split into five groups and each fold is used as a testing set at some point. These parameters were set before the algorithm was trained because it affects the reliability of the model.

In Table 4.1 is possible to observe the tuned parameters and the tested values. All sets of combinations were tested for each model, and the set of parameters that minimized the evaluation metric was chosen. Each adjusted parameter is essential for the construction of the model. *Max_depth* control the maximum depth of a tree, *min_child_weigth* defines the minimum weight necessary to create a new node or child, and *gamma* is a regularisation parameter which determines, in order to make a split, the minimum necessary loss reduction. The higher *gamma* is, the higher the regularization and the more conservative the algorithm will be. *Subsample* and *colsample_bytree* are used to help avoid the overfitting of a single sample or feature, in which the first corresponds to the percentage of observations/rows used in each step and the second one corresponds to the portion of features utilized for each tree. Lastly, *eta* has the same behavior as the leaning rate and shrinks the weights associated with features after each round, so we have to use a small value enough to avoid overfitting but not minimal because it makes the computation substantially more difficult.

Table 4.1: Set of parameters used in XGBoost Tuning.

Parameters	Set
Maximum Depth	$\{3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$
Minimum Child Weight	$\{1, 2, 3, 4, 5, 6, 7, 8\}$
Gamma	$\{0, 0.1, 0.2, 0.3, 0.4\}$
Subsample	$\{0.7, 0.8, 0.9, 1.0\}$
Colsample_bytree	$\{0.7, 0.8, 0.9, 1.0\}$
Eta	$[0.01, 0.35]$ with a step of 0.005

The customized evaluation metric combines both the MAPE (2.4) but also the percentage of within cases, where surgeries considered within present an error of less than 10%, concerning the difference between the actual duration of the surgery and the one predicted by the XGBoost model. Moreover, if the evaluated metric does not improve the performance on our intermediate validation after ten rounds during tuning, we enforce an early stop mechanism to halt learning. In that case, early stopping will occur before reaching the total number of rounds defined by the number of boosting rounds, corresponding to the number of trees to build.

Finally, the results shown in Table 4.2, were consistent with the results presented in Bartek et al. paper. All models learned better results than the current CUF estimates and the surgeon-specific models still have a 6% improvement in within cases in respect to the specific model for each specialty. The models' performance was measured based on within, overutilization, and underutilization cases. Within cases are those in which the forecast has a maximum error of 10%, which is the threshold chosen, and therefore the higher this percentage, the better the model's performance. Overutilization and underutilization are cases estimated with an error greater than 10%, with a time shorter than the real one and with a time exceeding the actual case-time duration, respectively.

Lastly, it is important to mention that the cases of within, overutilization and underutilization achieved for the CUF model vary between the different approaches since the split is different. However, the difference (Δ) between the approaches is minimal and less than Δ 0.008. Therefore, only the values for general case CUF model are shown in Table 4.2 because they are practically the same for specific models.

Table 4.2: Validation error obtained for each approach with XGBoost algorithm and from CUF model.

Model	Within	Overutilization	Underutilization
CUF	0.20	0.31	0.49
General Model	0.26	0.39	0.35
Specialty-specific Models	0.27	0.41	0.32
Surgeon-specific Models	0.33	0.41	0.26

Regarding specialties, the performance distribution within the models is not always constant. As shown in Figure 4.1, dermatology is the specialty in which XGBoost can obtain a more outstanding performance, which may be associated with a low standard deviation compared to other specialties. In addition, XGBoost's prediction does not improve in plastic and reconstructive surgery, although the precision is not significantly worse, perhaps because this is one of the surgeries where we have the most remarkable data dispersion, being the second largest standard deviation and variance among specialties.

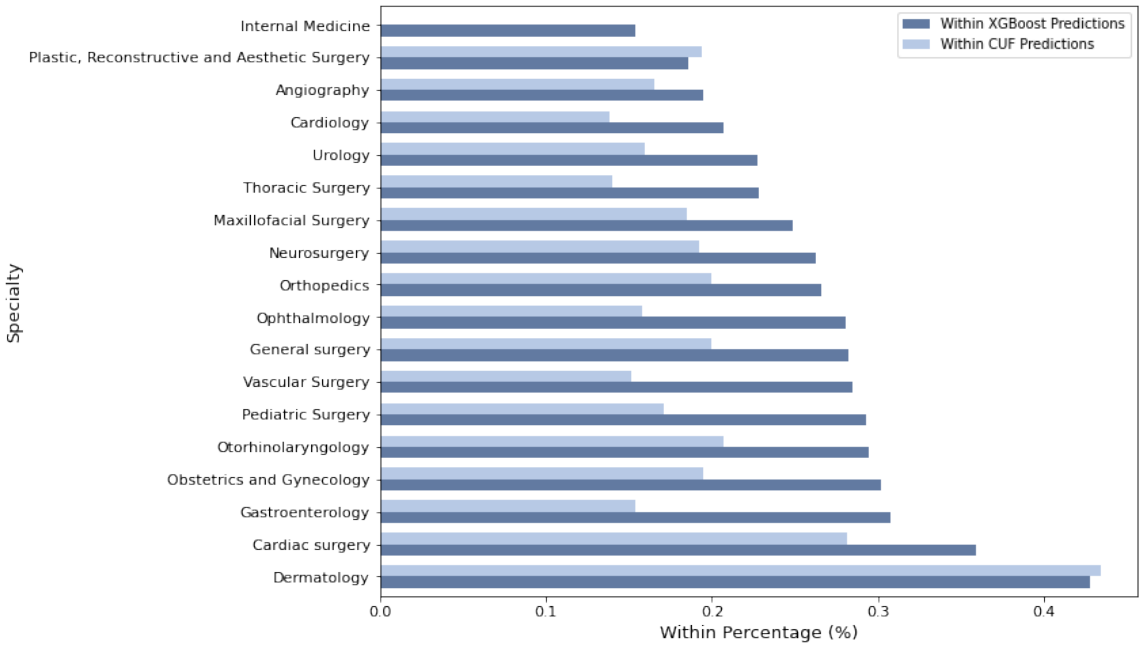


Figure 4.1: Distribution of within cases using XGBoost and CUF predictions for each specialty. The 18 specialty-specific XGBoost models varied in their within cases. 88% of all models have an accuracy greater than or equal to that of CUF schedulers.

Moreover, it is important to understand how these predictions are performed and interpret both specialty-specific and surgeon-specific models since the explainability of models in high-risk decision-making is crucial. Hence, Shapley Additive Explanations [45] was used to estimate SHAP values and

explain the output of XGBoost models.

Thus, the SHAP summary plot was performed to graphically represent each feature's importance and effect in the model given the data. The SHAP summary presents, for each feature, SHAP values of each observation as a data point. The features are also arranged in descending order according to their importance for the model.

The assigned color is related to each point's value in each feature, where low values correspond to more bluish colors and high values to red colors. In addition, we have wider regions than others in each feature, which correspond directly to the SHAP values distribution since overlapping equal points generate an increase along y.

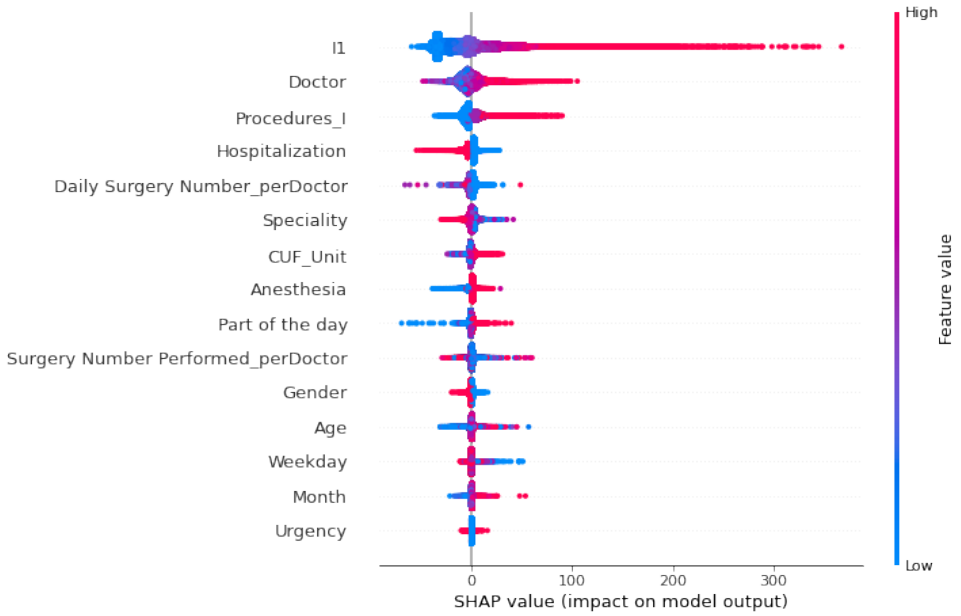


Figure 4.2: SHAP summary plot of XGBoost general model. The y-axis indicates the variable name in order of importance from top to bottom. The top one, the first procedure, is the most contributor to the predictions. Shap values are provided in the x-axis. The color of each dot, which represents a single data point, denotes the value of that on corresponding feature.

The SHAP summary plot for the XGBoost's general model, the one that receives all data without any particular alteration or specification, is presented in Figure 4.2. The four most important features are the first procedure, the doctor, the number of procedures performed during the surgery, and the type of hospitalization. The way the model identifies the relevant features is very interesting, emphasizing the importance of a specific model for each doctor due to the relevance given to the doctor column.

In Figure 4.3 is possible to observe a SHAP summary plot for the orthopedics model, one of the 18 specialty-specific models. The four most important features are the first procedure, the number of procedures performed during the surgery, the doctor and if the patient is an outpatient or inpatient. This set of features are also highlighted as the most important across all specialty models.

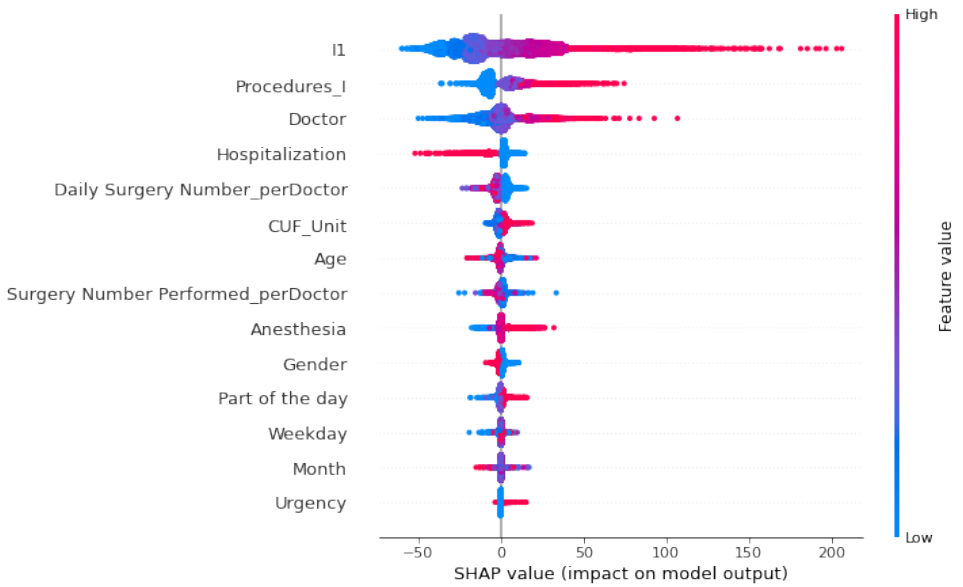


Figure 4.3: SHAP summary plot of orthopedics specialty. The y-axis indicates the variable name in order of importance from top to bottom. The top one, the first procedure, is the most contributor to the predictions. Shap values are provided in the x-axis. The color of each dot, which represents a single data point, denotes the value of that on corresponding feature.

Regarding the first procedure, this is the most important feature and the SHAP values are widely distributed between -50 to 150. This feature has been encoded with target encoding, so positive SHAP values represent procedures with high variable values. Thereby, as this feature considerably impacts the model's prediction, samples with high first-procedure values are likely to have a longer estimated surgical time.

The second feature with greater importance is the number of procedures associated with surgery. Surgeries with more procedures are correlated with positive SHAP values, which is expected since surgeons with more procedures tend to take longer to perform the surgery. These longer times occur due to the time required in each procedure and because there may be an interval of time necessary between each procedure, e.g. material change. Most surgeries are generally related to a negative procedure's SHAP value, as there are usually 1 or 2 procedures per surgery.

The surgeon also has a relevant impact on the final value predicted by the XGBoost model according to the SHAP analysis, which can justify the significant enhancement in predictions when models are trained according to each doctor. The doctor was encoded using target encoding by the same logic as the first procedure, so negative SHAP values are related to doctors with lower coding values. However, in most doctors for the orthopedics specialty, the SHAP value is around zero between -10 and 10.

Concerning hospitalization, this feature can only take two values: 0 or 1 for inpatients or outpatients, respectively. As expected, inpatients are related to higher SHAP values since the need for a bed at least one night before surgery may imply that surgery is more complicated or requires patient monitoring care.

Lastly, it is interesting to note that the fifth most crucial feature, the number of doctor's surgeries in a day, presents a curious behavior. Surgeries with lower values in this column, such as the first or second

surgery of the day for a given doctor, tend to be longer than the last surgeries of the day. This behavior could be related to scheduling longer surgeries at the beginning of the day or time tightening throughout the day due to overestimation issues.

Conversely, Figure 4.4 presents the SHAP summary plot for one of the 322 models treated for each doctor, in this case for the doctor with the number 132273102 coded as 65.9142 through target encoding. Contrary to what happened for specialty models, the importance order of features is no longer as consistent for all surgeon-specific models, which confirms the importance of the surgeon to create variability between different models. The first procedure is transversely the most important across the models. However, the second and third most important features vary between age, gender, hospitalization, type of anesthesia and number of procedures performed depending on the surgeon. However, we can already remark similarities in the three most important features by analyzing doctors within the same specialty.

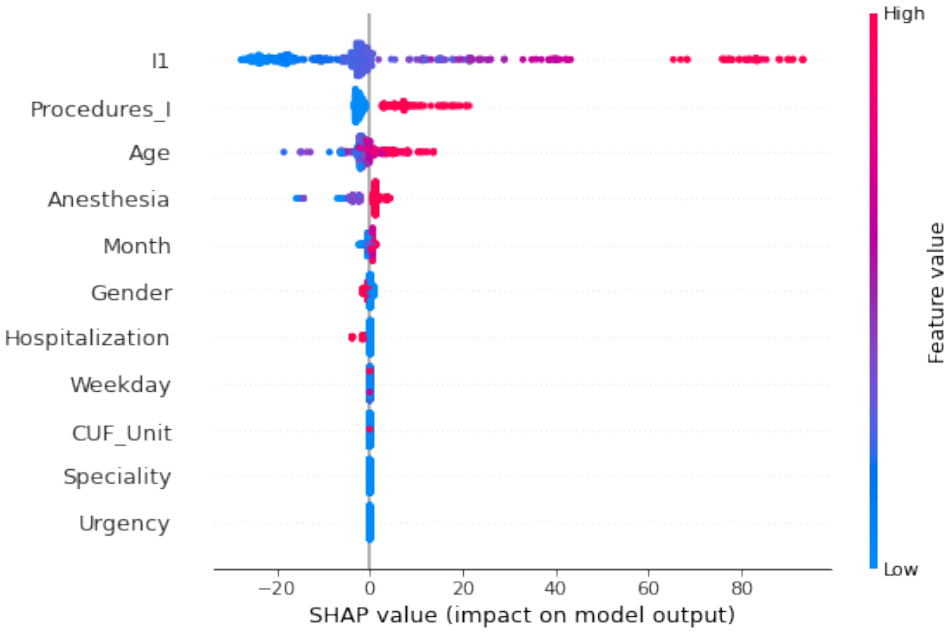
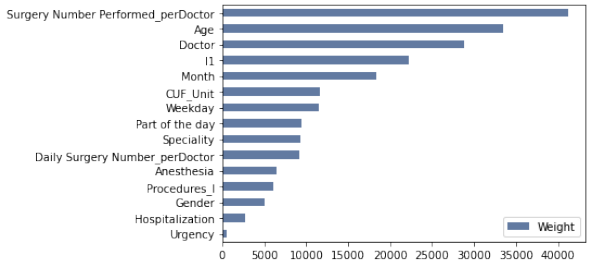


Figure 4.4: SHAP summary plot of surgeon ID 132273102. The y-axis indicates the variable name in order of importance from top to bottom. The top one, the first procedure, is the most contributor to the predictions. Shap values are provided in the x-axis. The color of each dot, which represents a single data point, denotes the value of that on corresponding feature.

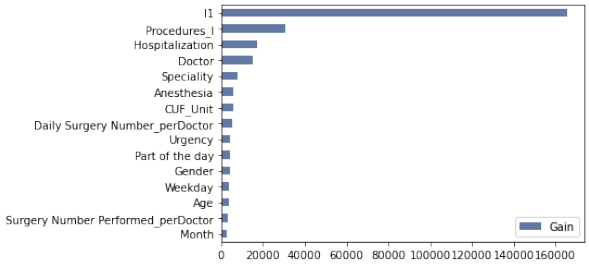
Even when faced with XGBoost's robust accuracy, we must not forget that explaining the predictions of this ensemble tree is difficult. Furthermore, identifying the most relevant features for the model is not easily extracted because different measures report different feature importance orderings, making the model even less reliable. In Figure 4.5, we ran three options for measuring feature importance in the general XGBoost model, where weight states for the number of times a feature is used across trees, gain represents the average gain of a feature when used to split data and cover means the relative number of observations related to a feature. As we can notice, the ordering of features varies considerably with the chosen approach, and even the feature reported as the more influential of income is different.

As plausible, the use of XGBoost in the health context must be done very carefully, as we cannot report conclusions unequivocally without complete confidence in the model's behavior. Analyzing feature importance derived from SHAP, Figure 4.6, we observe a different feature importance ordering, highlighting the inconsistency of previous methods. The charts from all methods can produce different results, leading us to raise questions about their application. For a given problem, it is not easy to conclude which is the most appropriate feature importance method for the situation and which we should consider.

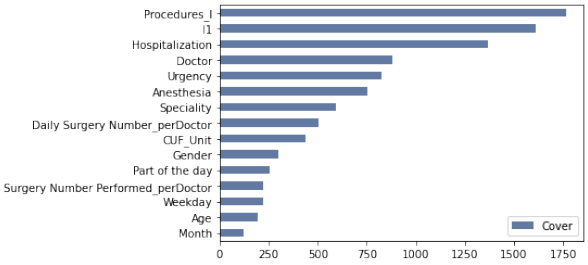
To conclude, it is important to highlight that XGBoost behaves very well under imbalanced data, in line with results found in similar research works [51, 52]. However, we have to point out that as a tree ensemble and therefore as a combination of the predictions of different decision trees, XGBoost ends up approaching a black-box model where we would have very difficult to interpret without SHAP demystifying the models [53]. Although SHAP had been developed to address the problem that current feature attribution methods are inconsistent [54], we have difficulties explaining the reasoning process behind their predictions.



(a) Relative importance of features for weight metric.



(b) Relative importance of features for gain metric.



(c) Relative importance of features for cover metric.

Figure 4.5: Feature importance chart for general model.

4.3.2 RuleFit Implementation

After analyzing CUF data performance with XGBoost, we developed a ML model based on rule-based methods through RuleFit [26] implementation. This interpretable framework was chosen once interpretability is fundamental in the health context.

Regarding inputs, features were used without additional normalization or scaling because decision trees are not sensitive to the variance of the data and are not impacted by outliers.

During the development of the RuleFit model, we came to understand that we should vary the type of ensemble tree that generates the decision tree and the type of regularizer applied to reduce the number

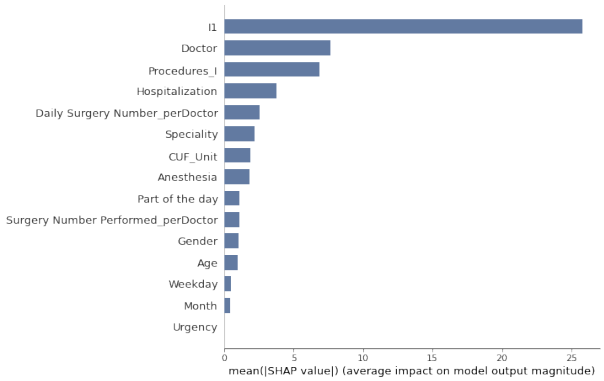


Figure 4.6: Feature importance computed in general model with SHAP values.

of decision rules. Hence, the RuleFit algorithm itself had to be customized to meet what we wanted to perform.

In a first instance, the RuleFit framework only allowed the use of Random Forest and Gradient Boosting as a decision tree at the regression level. Thus, considering the ensemble methods available in the *Scikit-Learn* library, we added Bagging Regressor since it has a lower tendency to overfit than Gradient Boosting. Additionally, Extremely Randomized Tree was also added as an option of the decision tree since Random Forest in a first analysis worked interestingly. Thus, we would analyze if Extremely Randomized Trees could behave similarly or even better since it is usually related to a reduction in variance from the bias-variance point of view [55].

Finding the best parameters of ensemble trees is also important, mainly the maximum tree depth, to ensure interpretability. Thus, as also advised by Molnar [56], we should keep the depth at a reasonable level, so the rule does not have more than 3 or 4 conditions.

The number of estimators was also varied. In the case of Gradient Boosting, Random Forest and Extra Trees, the decision trees ensembles fit several trees and then average them. Thus, if the number of estimators is 10, we have ten trees and the final result is performed through the average of these ten trees. In Bagging Regressor, the number of estimators is related to the base estimator utilized, which in our case is a Decision Tree Regressor.

Lastly, the learning rate was also tuned, but this one is only present in Gradient Boosting. This parameter will shrink the contribution of each tree so that it will be directly related to the number of estimators. Thereby, in Table 4.3 is possible to observe the range of values tested for each parameter.

The total number of combinations is not so expressive as in XGBoost, so we do not need to limit the trails number. Each tree was optimized by cross-validation grid-search, with a 5-fold and the negative MSE as a scoring strategy. A negated version of the score needs to be utilized because, in *GridSearchCV*, a high score is better. Thus, we need to return the negative to handle losses.

Furthermore, RuleFit can control a large amount of generated rules through Lasso regularizer, where some rules get non-zero weight. However, to understand which type of regularizer would work better, we also customized the RuleFit regularization to receive any of the following regularizers: Lasso, Elastic Net and Pyowl [57].

The motivation for the regularizer's choice was the following. Firstly, we kept Lasso because it is

Table 4.3: Set of parameters used in Ensemble Methods Tuning.

Parameters	Set
Learning Rate	$\{0.150.1, 0.05, 0.01, 0.005, 0.001\}$
Number of Estimators	$\{100, 250, 500, 750, 1000, 1250, 1500, 1750\}$
Maximum Depth	$\{3, 4\}$

the default regularizer in RuleFit and has an essential role in variable selection. Elastic Net was also used as a regularizer since it usually tends to be more accurate and does not remove features so aggressively as Lasso. Elastic Net is a combination between Lasso and Ridge and takes advantage of both. Lastly, we added Pyowl [57], a recent algorithm that robustly handles highly correlated features, puts the coefficients in order and then discards the lowest ones. This algorithm uses the Ordered Weighted ℓ_1 (OWL) family of regularizers for sparse linear regression. It is known to behave better than previous regularizes when it is desirable to identify all of the covariates relevant for modeling the data in high-dimensional problems since, in a situation where there is near-linear dependence among a few covariables, the Lasso and Elastic Net are unsatisfactory and tend to select one variable from the group of correlated variables [58, 59].

Regarding Ridge, the instability of L2 regularization led us not to use this regularizer. The ridge penalty can not shrink the coefficient precisely to zero, so it will not reduce the number of rules generated by decision trees [60].

In our rule fitting method, the set of regularizers run in two distinct ways. Firstly, by cross-validation in order to find the most suitable parameter alpha, and afterwards without cross-validation and setting a specific value to alpha instead of a range. The alpha assigns how much weight is given to each penalty, so a higher alpha implies a higher penalty and more rules will be set to 0 after using the regularizer. To choose the suitable alpha without cross-validation, the developed algorithm only accepts to generate a RuleFit model when the total number of rules after application of the regularizer is less than 50, so now it is possible to control the rules without reducing them in a more ”raw” way.

We use the rule fitting method with and without cross-validation because of the number of rules we want to get. When we first developed a model that looked for the best alpha value by cross-validation, we got a very high number of rules, over 500 rules. So, to force the model to reduce the number of rules, we had to define the maximum total number of rules that we wanted to get at most before applying the regularization technique and, in this way, we were able to work around the problem.

However, sometimes setting the maximum number of rules at the beginning can make the model worse than using a specific alpha without cross-validation and large enough to obtain enough rules for what we are looking for. So, this was the reason that led us to test RuleFit with both Lasso without cross-validation and Lasso with cross-validation but with the maximum number of defined rules. Regarding regularizers, only the Pyowl implementation does not allow running the cross-validation method because the algorithm requires that we set parameters alpha and beta at the beginning.

We started by analyzing all the possible combinations for each model approach. As it is plausible, we can not look for the best combination for each model within specialties and surgeons due to the high

number of models. Thus, since we are also interested in defining the same combination for all models within the same approach, a set of five specialties and five surgeons were analyzed to understand which best combination fit the data better. The selected parameters chosen for each approach to generate the RuleFit models are present in Table 4.4.

Table 4.4: Selection of parameters used by the rule fitting method in each model approach.

Model	Decision Tree Ensemble	Regularizer	Cross-Validation
General Model	Gradient Boosting Regressor	Lasso	No
Specialty-specific Models	Gradient Boosting Regressor	Lasso	No
Surgeon-specific Models	Bagging Regressor	Lasso	No

The Lasso regularizer without cross-validation consistently obtained the best results for the different approaches from the set of combinations. With cross-validation and respectively limiting the maximum number of rules, we obtained a higher MSE error.

The difference between the model approaches lies only in the type of decision tree. For the general case and specialty models, Gradient Boosting Regressor is the chosen ensemble method, whereas Bagging Regressor was the decision tree that showed the best results for the case of surgeon-specific models. The comparison method used across combinations was the MSE error value in the validation dataset and the percentage of within cases, surgeries with an error of less than 10%, resulted from each decision tree ensemble.

The results obtained for each model approach are presented in Table 4.5. As expected, CUF estimates are less accurate than each approach and it is possible to verify that the less generic the dataset is, thus moving from the general model to the surgeon's models, the percentage of correct predictions increases.

Table 4.5: Validation error obtained for each approach with RuleFit and from CUF model.

Model	Within	Overutilization	Underutilization
CUF	0.20	0.31	0.49
General Model	0.22	0.30	0.48
Specialty-specific Models	0.24	0.29	0.47
Surgeon-specific Models	0.26	0.30	0.44

Concerning each specialty, like in previous algorithms, it is possible to observe the distribution of within cases in Figure 4.7. The RuleFit algorithm cannot overcome CUF's predictions in dermatology by 4% and plastic, reconstructive and aesthetic surgery by 1.5%. However, in general, there is a significant improvement of 4% in relation to CUF estimates. Moreover, contrary to what had happened in the current estimation standards, RuleFit improved the estimates by 7% in internal medicine, one of the most challenging specialties due to the reduced number of data and the great dispersion and variation of surgical times.

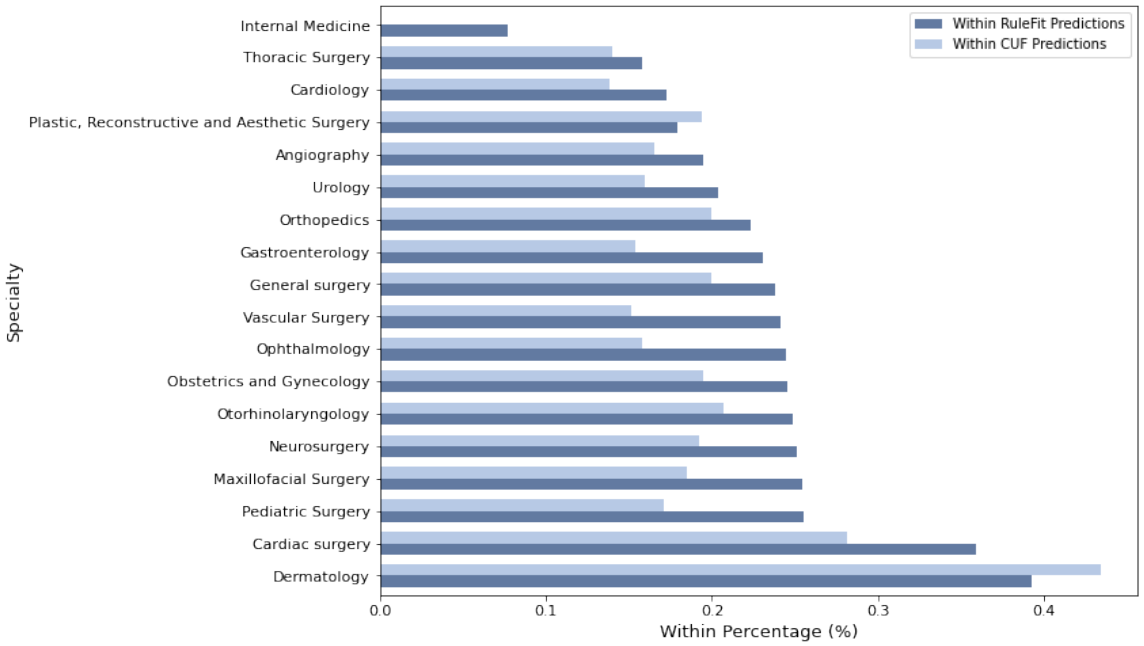


Figure 4.7: Distribution of within cases using RuleFit algorithm and CUF predictions for each specialty. The 18 specialty-specific RuleFit models varied in their within cases. 88% of all models have an accuracy greater than or equal to that of CUF schedulers.

Before analyzing the rules generated by the RuleFit model, we use the model-agnostic tool available in RuleFit to visualize the feature importance given to the explanatory variables after training a tree-based model and identify the most significant features affecting the target. The measurement includes the importance of the raw feature term and all the decision rules in which the feature appears [56]. This visualization is not feasible for surgeon-specific models since the function *feature_importances_()* only works with Random Forest and Gradient Boosting Tree.

The tree-based model optimized creates several decision rules and the RuleFit model is posteriorly fed by a tree model and a combination with Lasso regressor. Therefore, the feature importance graph for the general model is present in Figure 4.8.

Like the previous algorithm, the first procedure and doctor are the most significant features affecting the target in the general case. Regarding specialties, there are some differences in the top 3 most relevant features. For example, the orthopedics model presents the same order of importance as the general model. However, the main one for dermatology is the doctor, then the first procedure and age as the third most important. In general, the first procedure and surgeon are always very relevant and from the third main feature to the last, the order varies a lot from specialty to specialty. However, these percentages of individual relevance are minimal, each one less than 0.05.

After this analysis, it would be expected that the most important rule combinations also mostly contain these features. Thereby, now it is crucial to analyze the most informative rules from the set of decision rules built by each RuleFit model to understand how the model is learning and which aspects it emphasizes. As we know, trees are a form of rules in which the paths to each node form one rule. In addition to the set of rules generated, RuleFit also informs us of the support and importance of each

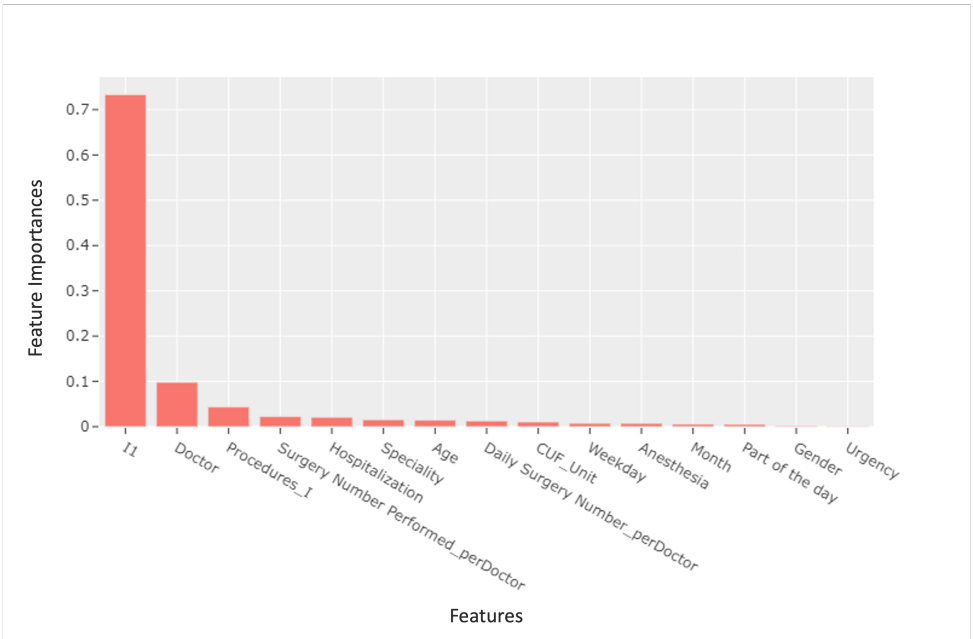


Figure 4.8: Bar chart listing the explanatory variables based on their significance level for the general model. The most important features for the predictions were the first procedure and the surgeon.

rule.

Table 4.6 displays the five globally most important rules resulting from the general RuleFit model based on their estimated importance. The coefficient column represents the slope parameter in the case of linear terms and, for rules, represents the change in predicted value if the rule is satisfied. The support column represents the proportion of the dataset that the rule applies to. The importance column is calculated from the regression model's weights, and it is measured as the overall impact on predictions which is given as the posterior mean weighted with its standard deviation [61].

Table 4.6: Five of the rules that were generated by general RuleFit model, along with their support and importance.

Description	Coefficient	Support	Importance
First Procedure	0.68	1	28.07
Doctor	0.18	1	5.81
First Procedure $\leq$ 181 and Number of procedures $\leq$ 2.5	-10.35	0.84	3.78
First Procedure $\leq$ 115 and Number of procedures $\leq$ 3.5	-8.25	0.80	3.25
Hospitalization $\leq$ 0.5 and Daily Surgery Number (individual) $\leq$ 1.5	5.9	0.30	2.70

In Table 4.6 the first and second terms reflect the linear dependence on the first procedure and doctor, congruent with previous Figure 4.8. The third and fourth rules indicate smaller target values when the number of procedures and the first procedure value are small. The fifth rule produces larger target values when there is no hospitalization and supports 30% of the data.

Regarding specialty-specific models, the first procedure remains with a linear relationship and as a rule of greater importance in the case of orthopedics. Then the number of procedures supporting the dataset reflects a linear dependence. As a third more important rule, surgeries without hospitalization

where the doctor has a high value and the surgery is the first of the day indicates a higher target function value.

For ophthalmology, the first procedure is the most relevant rule. Then, the patient's hospitalization and small doctor values indicate a smaller target value. Moreover, as a third important rule, surgeries with a high doctor number and high anesthesia target encoding value show a higher target value.

In general, within specialties it is possible to verify that the first procedure is consistently the rule of the most significant importance and rule combinations involving doctor, the number of procedures and the daily number of surgery for a particular doctor are also important features and support most of the dataset of each specialty.

Finally, rules generated by surgeons-specific models are not so similar to the general case. The surgeon ID 5050005, for example, has a rule combined with a low number of procedures, low value of the first procedure, and high experience representing the most important rule and having a smaller impact on the target function value. On the other hand, surgeon ID 776636291's main combination is an aggregate between the first surgery of the day in patients with age between 50 and 80 years old, reflecting a lower target value. Between different surgeons, there is a significant variation between the main rules. However, as happened in XGBoost, it is possible to observe more similarities between surgeons within the same specialty.

Contrary to XGBoost, the RuleFit algorithm generates a set of binary decision rules, not all informative but possible to filter, turning out to be more transparent and explanatory from the point of view of functionality since new features are known in the form of decision rules.

4.3.3 Feedforward Neural Network Implementation

In this Section we focus on a deep neural network modeled in *Tensor Flow* to understand the behavior of another black-box model on our problem. The complexity of a black-box model makes it difficult for a human to interpret the output and understand how the algorithm behaves. Even those who design them cannot understand how variables are being combined to make predictions, which can be a problem from an ethical point of view [62].

In several areas, including healthcare management, the models' explainability and interpretability are fundamental for the use of ML in everyday life. For a nurse to be able to commit to signing the time that a given machine estimates for a specific surgery, the latter needs to understand how the models work to provide confidence to the health professional. The doctor is relying more on the machine than on his own estimate, so it is essential to have this explanatory dimension of the generated models.

The neural network's approach allows us to understand whether we have significant differences when comparing interpretable and non-interpretable models since this is widely discussed topic in the ML field, and questions are often raised as to whether precision is sacrificed for interpretability. The use of black-box models for high-stakes decisions, as in predicting medical outcomes, only should be considered if no interpretable model can be constructed that achieves a good enough accuracy level [62].

Again, we developed three types of approaches: a general FNN with all data, surgeon-specific FNNs

and specialty-specific FNNs. Regarding the data split, the fraction used for training was 80% and 10% for testing. From our training set, 20% was considered as a validation set, an essential set to update hyperparameters during tuning. Moreover, the target on the training set was standardized by *StandardScaler* from *Scikit-learn*, so each instance was normalized by subtracting the mean and posteriorly dividing by the standard deviation as represented in Equation (4.1). The scaling process is critical to keep the continuous features identical in terms of the range.

$$z = \frac{x - \mu}{\sigma} \tag{4.1}$$

Using a *Keras Tuner* class, *RandomSearch*, we performed the model tuning to define a searchable space for hyperparameters that needed optimization. The tuner selects random combinations to train the model and evaluates them considering the objective function. Table 4.7 presents the range of values used to tune hyperparameters, however, it is important to note that before the selection of range parameters shown, some experiments were performed to narrow the searchable space.

These four hyperparameters can significantly affect the accuracy of the model and, for that reason, are chosen. First, the tune of the learning rate is crucial to control how much we are adjusting the weights of our network concerning the loss gradient. The tuning of layer units is necessary once a small number of units can lead to underfitting. A higher number of layers can increase the time it takes to train the network and reach a point where it is impossible to train the neural network adequately. Lastly, dropout is essential because it helps the neuronal network avoid overfitting, forces to learn more robust features, and not develop codependency.

Table 4.7: Set of parameters used in Feedforward Neural Network Tuning.

Parameters	Set
Dropout	$[0.1, 0.5]$ with a step of 0.05
Learning Rate	$\{0.1, 0.01, 0.001\}$
Number of layers	$[2, 20]$ with a step of 2
Number of units	$[16, 64]$ with a step of 4

As it is possible to verify, the number of combinations is extremely large, and running all combinations is computationally inefficient, so the maximum number of trails in *RandomSearch* was set at 20 and the MSE of the validation set was used as the objective function. After 20 trials, the best model architecture is chosen based on the minimum MSE obtained from the validation set.

The model's architecture is fundamental in neural networks since there are different layers available, different numbers of units in each layer, and other ways of combining these layers. Our first layer is a normalized layer in all models, which compute the mean and variance, storing them as the layer's weights. Then comes a set of dense layers, whose the optimum number of hidden layers and the number of units for each layer are optimized during tuning. Each layer is also associated with a rectified linear activation function, widely used in deep learning and recommended for most feedforward neural networks [28]. ReLU is linear for half of the input domain and can retain many of the linear properties of the models to facilitate the use of optimization methods.

Moreover, a batch normalization layer is introduced between each dense layer to prevent vanishing gradients [63] following the formula of standardization shown in Equation (4.1). This layer induces a faster and better convergence through normalization using mini-batches instead of the complete dataset. Lastly, a dropout layer is used immediately after batch normalization, so some units are randomly disregarded during the training phase. The last layer of models is always a dense linear layer, with one neuron to return an output per surgery.

To better comprehend the architecture of the models, Figure 4.9 illustrates the structure previously described. For example, a model with five as an optimal number of layers presents the following structure: a normalization layer; five sets of dense layers, batch normalization and dropout; and finalize with a dense layer with one unit.

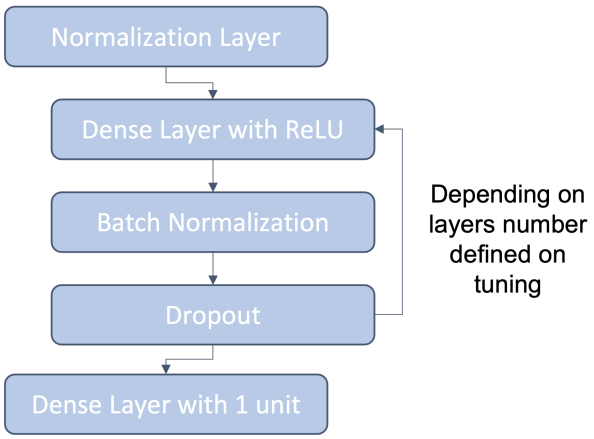


Figure 4.9: Architecture of Feedforward Neural Network.

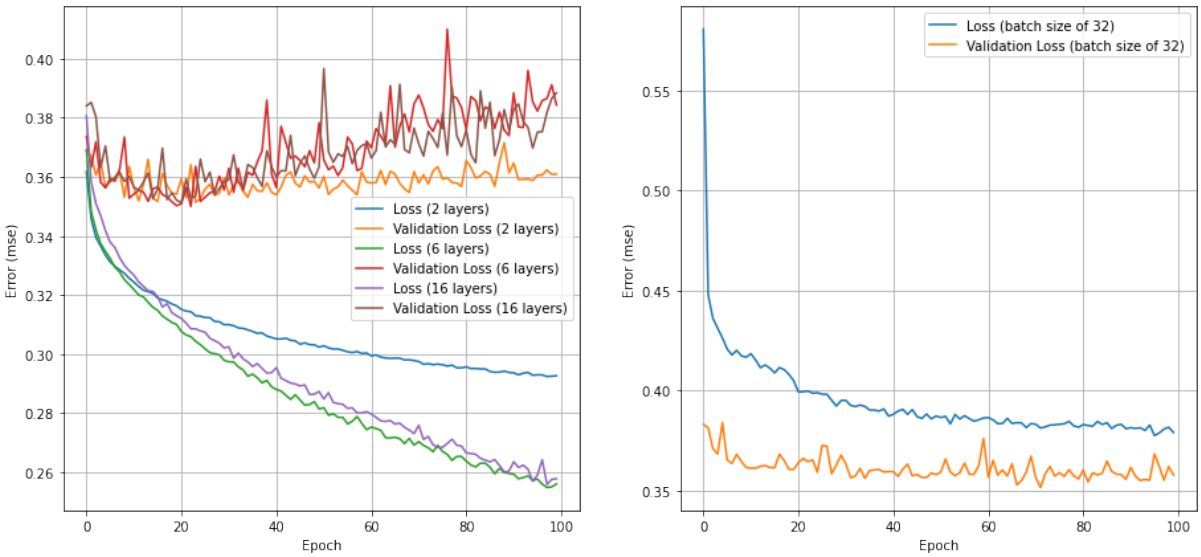
Afterwards, the architecture is fully defined when the loss function and the optimizer are specified. The loss function measures how close the model is towards the goal and since there are various types of regression losses, MSE and Mean Absolute Error (MAE) were studied. The best results were achieved with MSE after verifying how each function would be minimized when the model training algorithm iteratively updated the model parameters.

For the optimizer, the behavior of ADAM and SGD was studied. However, with SGD, a problem arises during the tuning for some models, where the loss takes the value NaN. This may occur because in neural networks regression problems, the output is unbounded, so minimization becomes difficult and issues such as exploding gradients can occur. This problem is related to the significant increase in the norm of gradient [64] during the tuning, which explodes weight terms with long components and the loss goes to NaN. Thus, we used ADAM in all models to avoid this complication, an adaptive learning rate that combines Momentum and Root Mean Squared Propagation (RMSprop) [28]. This optimization algorithm is an extension of SGD, where adjustments to the learning rate are performed during the training phase.

Regarding tuning, firstly, the optimum dense layers number for each model was found. The tuning is done separately from the remaining parameters to ensure that the model is not underfitting the training data. The set of layers number tested to find the best set that performed the minimum MSE in the validation set are shown in Table 4.7 above.

For each set of layers, the training step of the neural network was performed without any dropout or batch normalization layers. The number of epochs for which we need to train the neural network was set to 100, with no callback in order to visualize the U-shape of the validation set and the overfitting of the model. In Figure 4.10 a), it is possible to observe the number of layers tuning for the general model. Here for a better plot visualization, only three of the ten tested sets are shown. The model with six layers obtained the minimum MSE in the validation set and therefore, this value was selected as an ideal parameter to the general model.

In Figure 4.10 b) we are able to analyze the performance of the training after tuning the remaining parameters mentioned in Table 4.7. As expected, the addition of batch normalization and dropout layers significantly reduces the overfitting issues.



(a) Loss and validation loss during tuning of layers number with 2, 6 and 16 layers.

(b) Loss and validation loss with 6 layers and after implementing Batch Normalization and Dropout in model.

Figure 4.10: Loss and validation loss before and after parameters tuning. A neural network with six layers for the general model was chosen since present the lowest MSE. The validation loss behavior was not expected because it should be higher than training loss for the first epochs.

An important point to note is that both graphs are very noisy, indicating that the batch size is too small or the optimizer is not optimal. This noisy result is explained by using the default value of 32 as batch size, once the batch size was the last parameter to optimize and is not yet tuned. Moreover, the validation set presents a lower loss concerning the training loss, which occurs during the model's training after the parameters tune previously mentioned (Figure 4.10 b)).

One explanation may be that the validation set has 'harder' cases to learn, however, the split done is completely random. The initial split between training and validation set is done when fitting the model by order of the index, and *Keras* takes the first 80% of data points as training data and the last 20% as validation data. However, firstly, the split between testing and training data was performed using a random state of 123, so the validation data is generated from random surgeries of the training set and is a mixture of the four years present in the dataset. In Figure 4.11 is demonstrated the transformation applied to dataset.

Finally, for the general model, the value chosen for the number of dense layers was six, the specialty-specific models were four, and the surgeon-specific models were two. This decreasing need for layers may be related to the reduction of the data to generate specific models and hence, fewer layers are needed to predict the output from the features.

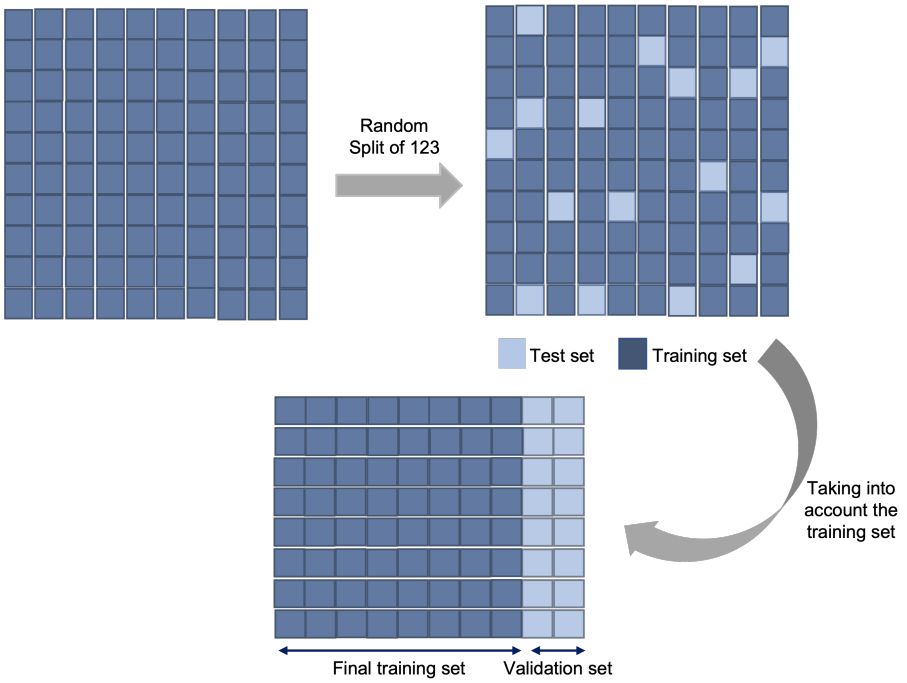


Figure 4.11: Split of data into test, train and validation set.

After the tune of the newly mentioned parameters through *RandomSearch*, batch size was also adjusted. This parameter is related to the number of training examples per single forward/backward pass before updating the model. This parameter is tuned during the model configuration for training. However, due to different datasets dimensions introduced in each model, individual ranges were created for each approach, as shown in Table 4.8.

Table 4.8: Range of Batch Size used in Feedforward Neural Network Tuning.

Model	Batch Size Range
General Model	$\{16, 32, 64, 128, 256, 512, 1024\}$
Specialty-specific Models	$\{8, 16, 32, 64, 128, 256, 512\}$
Surgeon-specific Models	$\{2, 4, 8, 16, 32, 64\}$

Afterwards, the models were trained with 200 epochs, therefore the learning algorithm learned through the entire training dataset 200 times. In addition, an early stopping epoch using the callbacks associated with the MSE error of the validation set was added to avoid overfitting of the model, so if the validation set error does not improve in 10 epochs, the training ends.

In Figure 4.12 is possible to analyze the final training graph for the general model with the respective loss and validation loss. Compared with figures previously shown when all the parameters were not yet optimized, it is possible to observe differences mainly in the noisy effect. This significant improvement is

related to the proper use of a batch size of 256 instead of the default value of 32. Furthermore, we can see that overfitting issues have been significantly reduced, as they only start to happen after 70 epochs.

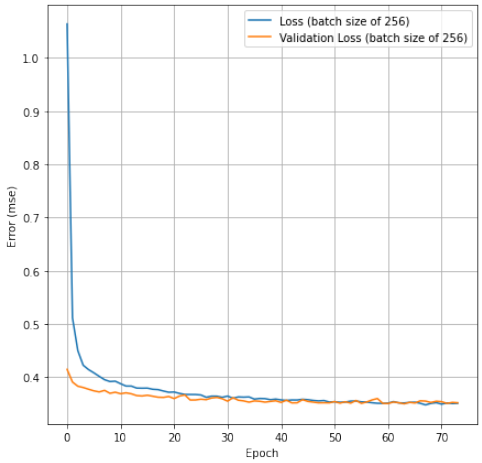


Figure 4.12: Validation loss and training loss of general model after tuning of all parameters including batch size.

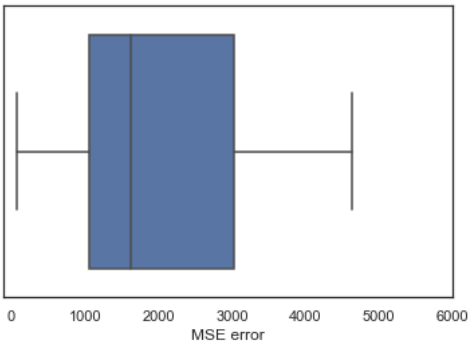
Subsequently, the three approaches were generated: 1 general model, 381 surgeon-specific models and 18 specialty-specific models. MSE models error and the percentage of within cases obtained by each model when applied to the validation set are shown in Tables 4.9 and 4.10.

Table 4.9: MSE results for surgeon-specific models, specialty-specific models, general model and CUF model.

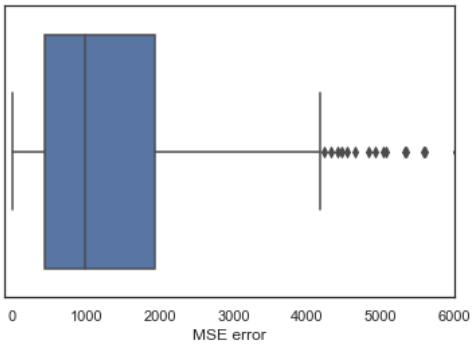
Model	Validation MSE Error	CUF MSE Error
General model	1193.13	2170.76
Specialty-specific models	1306.81	2232.96
Surgeon-specific models	1.31×10^{12}	1109.44

The reduction in MSE measure, the metric used as the objective function, is evident in the general model compared to the CUF model, however specific models presented a very high and significant error. Let us look only at Table 4.9. It seems that the models predict the surgeries reasonably poorly. However, if we additionally look at the boxplots represented in Figure 4.13, which show MSE error distribution for specific models, we can better observe the interquartile range and the outliers. These values fall outside this range and cause the explosion of MSE value. Boxplots summarize the distribution of a continuous variable, where we can identify the first (Q_1) and third quartile (Q_3), outliers and adjacent values. The whiskers, the lines extended from both sides, represent these adjacent values, observations within a distance of 1.5 times the interquartile range i.e. the high adjacent value is given by adj_H=Q_3+1.5$\times$ IQR, where IQR=Q_3 - Q_1.

In fact, this problem is common in neural networks where an error in an instance can cause an explosion of the total error in the final calculation of the MSE because there are no parameter in the forecast output handling the error explosion and informing the neural network that something is wrong. Thus, neural networks can fail, ending up propagating the error. Analyzing boxplots, we can observe



(a) MSE distribution for specialty-specific models.



(b) MSE distribution for surgeon-specific models.

Figure 4.13: MSE distribution in Boxplots. Box represents the data that exists between the first and third quartile.

that the error distribution within the 381 surgeon-specific models is highly different from the final value presented in Table 4.9 and there is also a reduction in the MSE concerning the CUF model looking to the median.

On the other hand, the error of specific models does not improve significantly in relation to the general model. This may happen because the architecture tuning in specialties was not performed individually. Given the number of surgeons and specialties, it was not doable to perform the tuning for each individual model and for that reason, the ideal number of layers was achieved with a limited set of models tested. However, the dataset sizes present in each specialty are much more variable than the size across surgeon models. Thus, this may be one reason for the inferior MSE compared to the general model. Nevertheless, the value is reasonable since it presents improvements regarding the MSE of the CUF predictions.

Considering the same metric used in previous models, the results from neural networks are also analyzed considering the within cases, the ones with less than an error of 10%. Thus, in Table 4.10 is noticeable the error distribution resulting from each model.

Table 4.10: Validation error obtained for each approach with FNN and from CUF model.

Model	Within	Overutilization	Underutilization
CUF	0.20	0.31	0.49
General Model	0.24	0.31	0.45
Specialty-specific models	0.24	0.34	0.42
Surgeon-specific models	0.24	0.33	0.43

Through this method, we are not able to verify a better forecast when we use specific models compared to the general model, and, unlike XGBoost and RuleFit models, the within cases in the surgeon-specific model do not present a significant improvement in relation to the specialty-specific model. These results may be related to the difficulty of the neural network to adapt to imbalanced datasets [65] since it works based on the calculation of errors and assumes equal costs. Therefore, neural networks end up adapting more to a particular class, in the case of classification, or to a range of more frequent labels in the regression case.

In addition, taking into account the specialty models, it is possible to notice the distribution of within predictions within each specialty model to understand which specialties can demonstrate more ease when learning with FNN. Surgeries such as dermatology, plastic surgery, cardiac surgery, internal medicine, and gastroenterology obtained worse results concerning CUF estimates.

Dermatology is a specialty with short durations of surgery with an average time between more than 77,000 surgeries of 25 minutes, so for these to be correctly estimated and evaluated as within cases, the error has to be less than 2 minutes and 30 seconds, which may justify the slightly lower accuracy than the CUF model.

Gastroenterological surgery and internal medicine are specialties that do not present a considerable amount of data, a total of 156 and 105 surgeries respectively, which implies that the training sets are at the limit of 100 surgeries, the minimum necessary for us to consider the development of the model. Thus, the small training dataset may explain the lower accuracy. In addition, CUF predictions are also barely accurate for the validation set provided in internal medicine specialty, so the small dataset that only contains 12 surgeries can also include more difficult surgeries despite the random split.

Lastly, as mentioned in the XGBoost section, plastic surgery and cardiac surgery are specialties with variance and standard deviation expressive, and therefore the wide distribution of times may explain the lower precision.

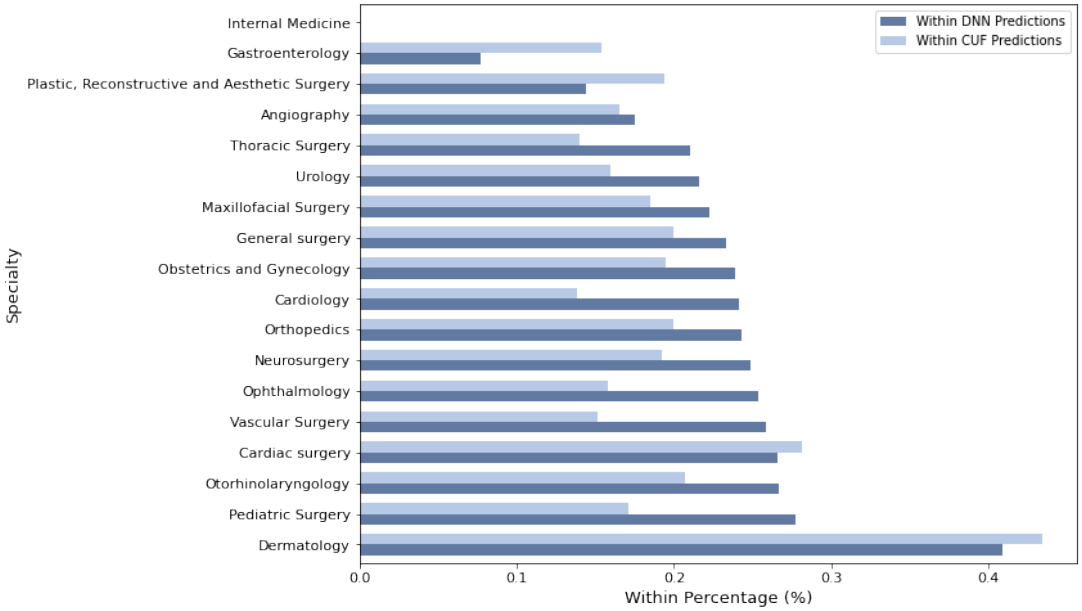
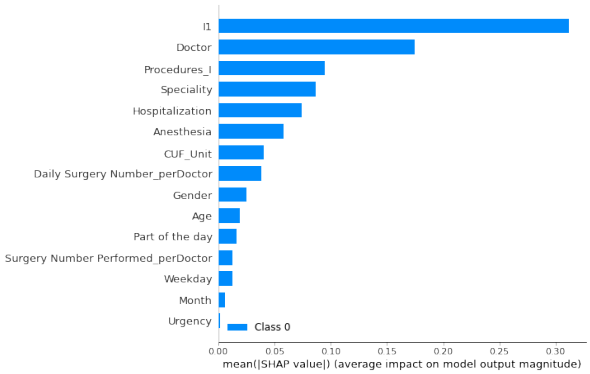


Figure 4.14: Distribution of within cases using Feedforward Neural Network and CUF predictions for each specialty. The 18 specialty-specific FNN models varied in their within cases. 77% of all models have an accuracy greater than or equal to that of CUF schedulers.

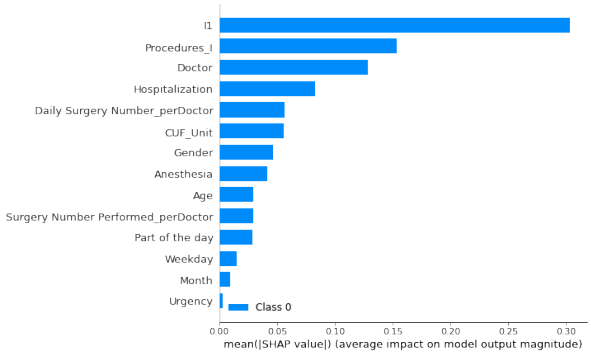
To conclude, an essential step after developing the models is the generation of an additional explanation to support the model interpretability. This *post-hoc* analysis is imperative in some risk-sensitive domains such as finance and medicine, where professionals are reluctant to trust a model without any plausible explanation. Hence, SHAP is crucial to assist in interpreting unexplained black-boxes as neural networks [66].

As mentioned earlier, neural networks are more complex than interpretable models due to the multiple interconnected layers that hinder the user's ability to comprehend models. Deep SHAP, a faster algorithm to compute SHAP values for deep learning models, comes to open some doors and point some lights into the model to be able to interpret some relevant points and slightly deconstruct the model.

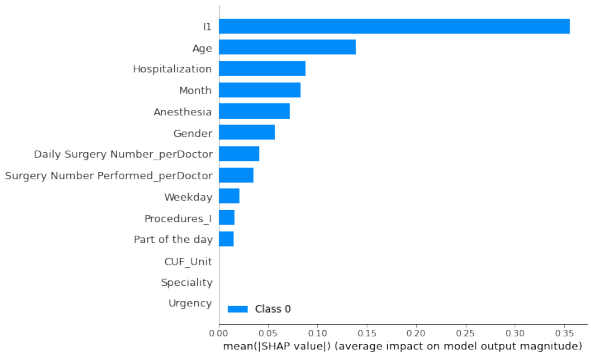
In the following Figure 4.15 is possible to deduce the different average contributions of each variable to the overall mean model's output. This proper interpretation method helps us understand the contributions in terms of units from each feature variable. Regarding the general model, we can visualize that the first procedure and the doctor are very relevant to generating the forecast, which is in line with what we get with XGBoost and RuleFit. Moreover, the ordering of features in the orthopedics specialty is similar to the general model, but gender now plays a relevant role. This variation in gender importance is very interesting since we have specialties such as orthopedics, where gender is one of the most relevant features, but others such as pediatric surgery, where gender is not so crucial for the final estimate.



(a) Average contribution of each feature to general model.



(b) Average contribution of each feature to orthopedics specialty model.



(c) Average contribution of each feature to 0094051011 doctor model.

Figure 4.15: Loss and validation loss before and after parameters tuning.

Lastly, in Figure 4.15 c), we can also observe the order of relevance of different variables for surgeon ID 776636291 from ophthalmology, however, this is only one of the 381 surgeon-specific models. The ordering of the example is not the same to all doctors, but it is possible to notice a more consistent pattern within doctors of the same specialty.

Furthermore, we can also observe individual attributions inferred by the model, as represented in

Figure 4.16, where we explain the 91st prediction of the general model. The graph refers to the expected value of the dataset as base value and each bar corresponds to a feature's importance value.

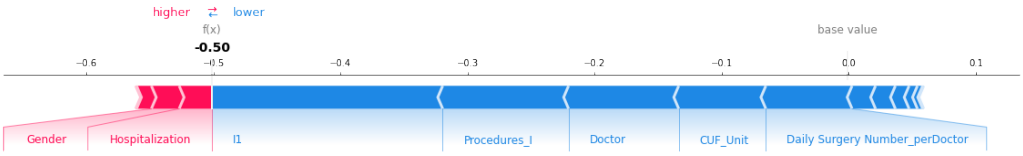


Figure 4.16: Features individual contributions from the 91st prediction of the general model. The "base value" is the mean for the output, which is approximately zero in our case. This is the baseline for predictions, and then the prediction changes according to the value of each feature.

We can infer how to go from the base value to the output value by analyzing each feature's contribution. In this case, most feature variables pull down the expected surgery time, except for the gender and hospitalization variables.

Chapter 5

Balanced Approach

We will now have a data balancing approach in addition to the algorithmic analysis and the use of more robust methods like XGBoost for imbalanced datasets. Our data is imbalanced and has difficulty delivering optimal results for some surgeries presented in less predominant regions. Therefore, we enriched three different datasets by using the GMM to improve the prediction of surgeries and rebalance the dataset.

In the previous Chapter, we worked with three different approaches, however now it is expensive to deal with the imbalance problem in all generated models due to the time involved with the generation of synthetic data. Thus, one model of each approach was selected: a surgeon, a specialty, and the general model. We trained a Gaussian Mixture Model to learn the probability distribution on the minority values of our label.

Finally, the Interpretability Curve was designed and presented, a curve based on Rashomon Curve but for imbalanced problems that will determine the desired model considering the complexity and accuracy. Furthermore, the Utility-Based Algorithm (UBA) imbalanced learning metrics will be applied to compare the results of imbalanced and balanced data.

5.1 Deal With Imbalanced Data

With the development of models so far, subsequent analysis of the forecasts performed and the plot of true labels against predicted labels, we verified that most models showed considerable difficulty predicting surgeries with a longer operative time. Figure 5.1 illustrates this bias where we clearly recognize a trend in long-time surgeries for these to be predicted with much shorter times.

As described in Section 2.6, most methods for dealing with imbalanced data are prepared for classification problems and not for problems where the target is continuous. Thus, in the imbalanced regression, we will have new challenges that will need our attention. Now, the continuous target no longer has hard boundaries, as happened in the classification problem, and certain target values may not have any data.

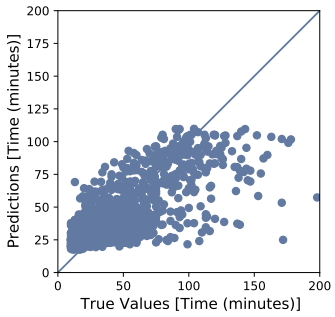


Figure 5.1: Graph of true labels versus predicted labels for the ophthalmology specialty with RuleFit algorithm.

After the detailed analysis we performed in Section 2.6, GMM will be used to the detriment of SMOTE. It presents greater advantages throughout the analyzed papers and thesis, proven to be more robust when generating minority class instances [33].

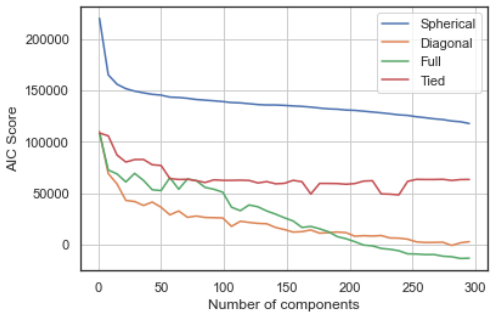
At this point, the method to control the imbalanced is chosen, but the classification problem remains to be overcome. By the same logic that we have done so far, we will want to control this imbalance for the general model with all the data, the specific models for doctors and the specific models for specialties. However, as understandable, due to a large number of surgeons and specialties in the CUF dataset, a doctor and a specialty, being this ophthalmology, were chosen. So, for each model case, we split our training set into three small sets. To avoid an extensive analysis, we exemplify below how we treat the imbalance for the ophthalmology case.

Firstly, a set was generated containing most of the data, including the mode, that is, all surgeries up to 40 minutes, so it is our majority class with approximately 70%-75% of the data. Then a second set with surgeries between 40 and 80 minutes and a last and third set with the remaining surgeries longer than 80 minutes.

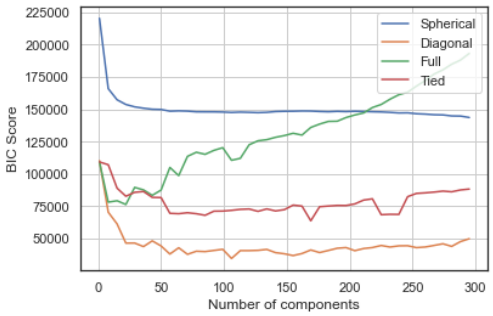
From this moment on, we can consider having three types of surgeries, belonging to class 1, 2 or 3, and so we can treat the problem as a classification problem. In this way, taking the sets of the two minority classes, we will generate new synthetic samples using GMM. Therefore, different types of covariance were analyzed, including spherical, diagonal, full and tied, using Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) as the basis for evaluating the model's complexity and for further model selection.

Analyzing the BIC and AIC plots presented in Figure 5.2 was possible to verify that the diagonal covariance is the one that better fits the ophthalmology data and which respectively obtain the lowest value of AIC and BIC score. Then, to evaluate the optimal number of components, technically is necessary to calculate BIC and AIC gradients represented in Figure 5.3 b). At some point, the increment in the number of clusters will not change significantly and the slope of the line is practically constant as we can detect around 50 clusters. From the definition of gradient, we know that if two consecutive points have approximately the same value, their gradient is close to zero, which in the context of AIC and BIC scores will mean that when we have a smoother slope and gradient close to 0. Hence, the increase in gain may

not justify further increasing the number of clusters. Therefore, our selection of the optimal number of components will be based on BIC and AIC scores and their gradients.

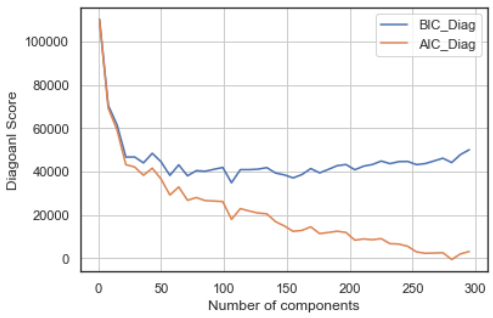


(a) AIC score

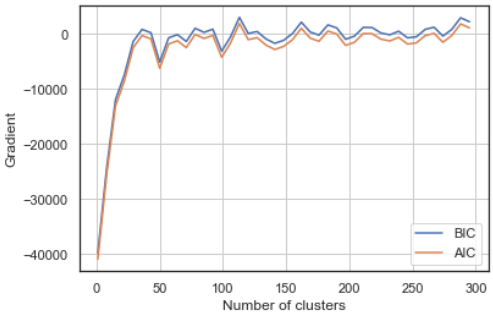


(b) BIC score

Figure 5.2: GMM selection for ophthalmology specialty using AIC and BIC scores.



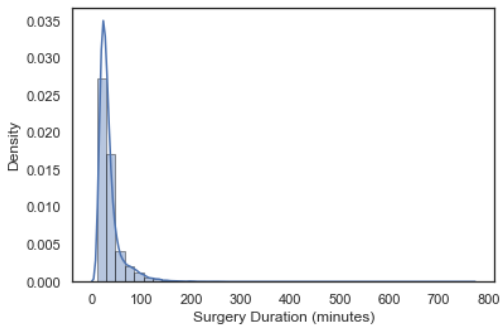
(a) AIC and BIC score



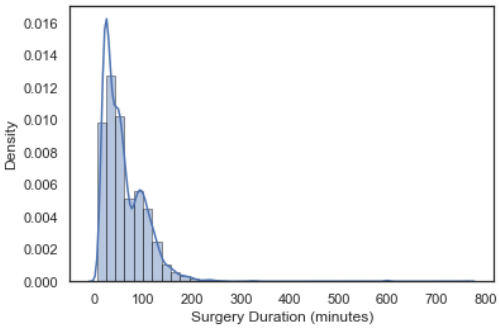
(b) AIC and BIC gradient

Figure 5.3: Selection of best number of components for ophthalmology specialty.

With the number of components and the type of covariance chosen, we proceeded with the generation of synthetic surgeries until each minority dataset had the same number of surgeries as the majority dataset. The final distribution from the original training dataset to a dataset that now contains more surgeries from classes 2 and 3 is shown in the following Figure 5.4.



(a) Surgery durations distribution before generate synthetic data.



(b) Surgery durations distribution after generate synthetic data.

Figure 5.4: Distribution of surgery durations for the ophthalmology specialty dataset before and after the generation of synthetic data in minority classes by GMM.

Finally, with this new training dataset which contains all training surgeries plus those generated by the GMM, our model will no longer have such a bias behavior that it had previously. It will have a better distribution of the surgeries' durations, allowing that the model learns from a more uniform range of values. It is important to emphasize that the test set was kept intact, as the objective is to verify the quality of the predictions in the test and how the model will generalize to new data.

Lastly, as mentioned above, the example we showed was applied to the ophthalmology surgery training set with surgeries over the four years. However, since we are also interested in developing a general model and specific models for each surgeon with more than 100 training surgeries, all data and one surgeon data were used to generate synthetic data and analyze the improvements in these models.

It is also important to mention that the training set, test set, and validation set must have the same percentage of each class in their dataset to interpret the results after the development of models correctly. In fact, the initial random division performed to data presented in Section 4.2 already respects these criteria and therefore no further changes to the datasets were necessary. The two minority classes together always present a consistent percentage between the different datasets of each model. This percentage varies between 25%-30%, with a variation of less than 1% between the training, test and validation datasets in the general model and surgeon model, and a difference of less than 2.5% within datasets in the ophthalmology model.

5.2 Interpretability Curve for Imbalanced Data

In Section 2.8.2 the Rashomon Curve was described, a curve that allows us to find a relationship between simplicity and accuracy. The proposed diagnostic tool allows us to achieve the best balance between the right level of complexity and high accuracy, where properties such as interpretability and generalizability are guaranteed. However, for the purpose of our thesis, the build of Rashomon Curve is of extreme difficulty since the calculation of the Rashomon Ratio is too complex and involves the development of more than 250,000 decision trees per fold to calculate each point of the graph. Furthermore, as our work is a problem of imbalance, the metric accuracy is not prepared to evaluate how we intend the models and a new version of the Rashomon Curve will be presented in this Section.

Thereby, to obtain an equally explanatory graph at the level of the same trade-off, the Interpretability Curve is designed to better respond to the problem at hand. The curve will evaluate the relationship between complexity as a function of the empirical loss of the train set in the minority classes to assess the Root Mean Squared Error (RMSE) in the class that the model has more difficulty in predicting. The Interpretability Curve is represented in Figure 5.5.

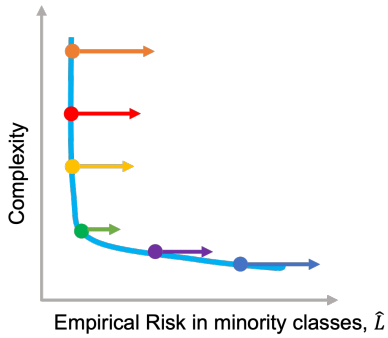


Figure 5.5: The proposed curve illustrating the generalization ability of the elbow on lower left corner and the empirical risk effect of increasing model complexity.

Regarding the y-axis, the complexity will be represented by the number of rules generated by RuleFit or the depth of the XGBoost trees. Thus, to generate the respective Interpretability Curve for each algorithm, a set of trees with different depths and different number of rules are initialized to generate the final curve. The Interpretability Curve will be used for the model selection and to choose appropriate complexity, for the case of XGBoost the ideal depth, and in the case of RuleFit it will help choosing the ideal number of rules. As described by Semenova et al. [47] the elbow of the curve seems to be a reliable model selection criterion, an important selection from the interpretability point of view.

Unlike the Rashomon Curve, the arrows will represent the difference in RMSE in the minority classes between the validation dataset and the training dataset to identify the transition between the model's underfit and overfit. Arrows will be crucial to find the sweet spot where there is most likely to be a suitable generalization of unseen data.

Hence, choosing the ideal depth for our model will not only take into account the suitable point where complexity no longer has a major impact on the training error of the minority class, but will also bear in mind whether there is overfitting in the minority values during training with a negative impact on the performance of the model on new data.

The designed Interpretability Curve is able to adjust on an imbalanced dataset and ensure valuable properties such as interpretability. Inscrutable models fail to accomplish the work purpose since we cannot explain individual decisions and the generated output. This key factor limits the adoption of ML models in healthcare. In areas of healthcare, the explanation and comprehension of predictions for end-users allow healthcare experts to make reasonable decisions to the higher quality of their services [67].

5.3 Balanced Model Results

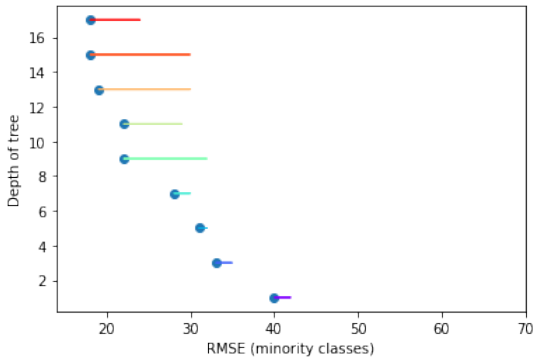
5.3.1 Model Selection

A new version of the Rashomon Curve, Interpretability Curve, and described in Section 5.2, will be used to find the optimal model. This approach will be applied both to the model generated from imbalanced

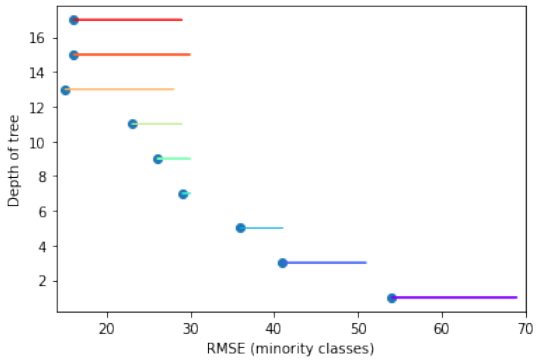
and balanced data and in both algorithms, XGBoost and RuleFit. Although XGBoost is not interpretable, we can select the suitable depth and understand the XGBoost behavior with balanced datasets through this curve. The curve will be used for model selection and to choose appropriate complexity.

To generate the respective Interpretability Curve with XGBoost, a set of trees with different depths were initialized to generate several models. The chosen and tested depths range within a nine fold step increment of two units each, ranging from 1 to 17 on depth. Then, several final metrics such as the final percentage of within surgeries, the error in minority classes in the validation and training dataset, in which RMSE was used, were calculated in order to choose the model. Now, we use the RMSE instead of the MSE because it is easier to interpret statistically and is in the same units as the target variable.

Therefore, presented in Figure 5.6 is the Interpretability Curve generated to ophthalmology using the XGBoost algorithm. As expected, we can observe a trend of increasing RMSE in the training set as we have a lower tree profundity, since we will have fewer splits and a higher tendency to underfit.



(a) Proposed Interpretability Curve for ophthalmology model with imbalanced data.



(b) Proposed Interpretability Curve for ophthalmology model with balanced data.

Figure 5.6: XGBoost model selection for ophthalmology.

As described previously, the choice of the optimal model will incorporate the behavior of the curve and the moment when the complexity does not justify the minor reduction of RMSE. However, we will also observe the behavior of the RMSE in the training and validation sets in order to avoid the overfitting of the model. Consequently, we will only accept models where the difference between both sets' error is inferior to 15%.

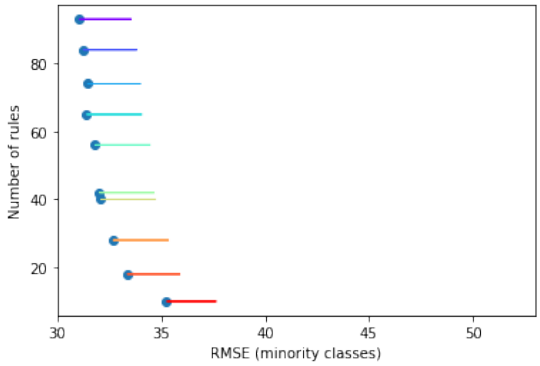
Therefore, based on Figure 5.6, the selected XGBoost model for imbalance data has a depth of 7 and for the ophthalmology model with balanced data generated from GMM, the tree depth will be 9. In the case of balanced data, 5.6 b) presents a minimal error when the depth is 11. However, not only 11 is a very high depth and hard to interpret, as the difference between the errors of the minority classes from training and validation datasets is almost 50%.

Moreover, observing both figures, it seems that models with imbalance behave better than models with balanced data. However, the data points belonging to the minority classes differ in both training datasets since the balanced training also has synthetic data, and thus its final training MSE handles more data. Therefore, we will analyse and compare both performances later with the UBA.

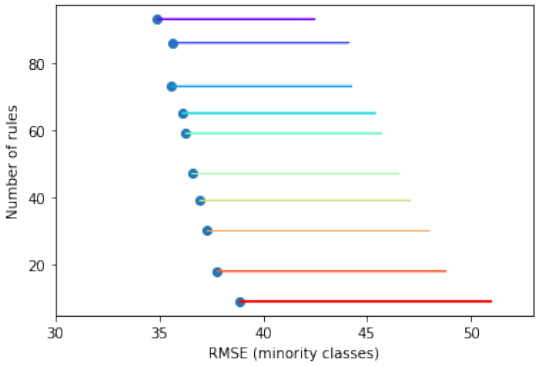
Regarding RuleFit, as the Interpretability Curve depends on the number of rules associated with each model, 9 models were generated each with rules between 1 and 100. In descending order, models

were generated first, one with a total number of rules between 90 and 100, followed by a second one between 80 and 90 rules, and so on until a last simpler model between 1 and 10 rules.

Figure 5.7 shows the proposed Interpretability Curve for the ophthalmology specialty with the RuleFit algorithm. Unlike XGBoost, arrows now have longer lengths, which means that a RuleFit model up to 100 rules has difficulty training the dataset. Thus, we have a high error in both training and validation sets, seeming that the increase in rules is not significant for the model to learn to model the data.



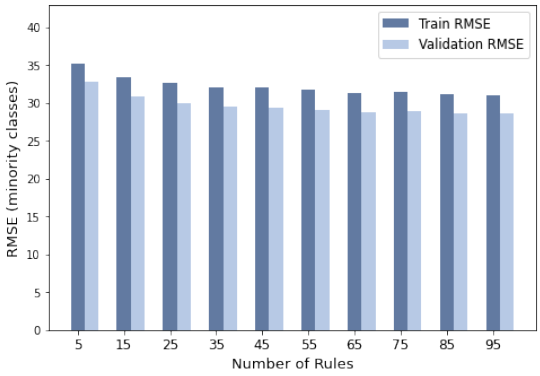
(a) Proposed Interpretability Curve for ophthalmology model with imbalanced data.



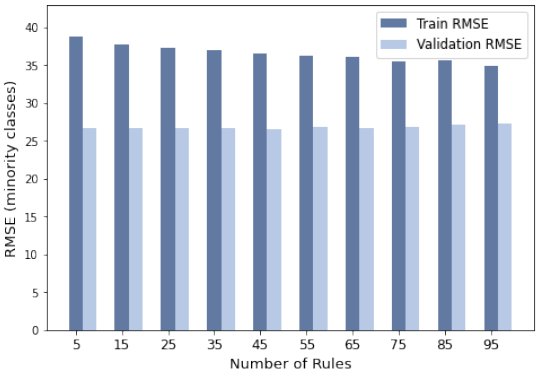
(b) Proposed Interpretability Curve for ophthalmology model with balanced data.

Figure 5.7: RuleFit model selection for ophthalmology.

Again, the difference between validation and training error is higher in balanced datasets, so the arrows have long lengths. However, observing the performance of the errors in Figure 5.8, we can notice that this significant difference is due to the error in training. Training in a balanced dataset contains more data belonging to the minority classes, having in total two times more data than the original dataset and a higher data dispersion, which may explain the increased error. In the next Section, through the implementation of utility metrics, we will be able to observe if models trained with the GMM can better predict less common surgical times.



(a) Train and validation error for ophthalmology model with imbalanced data.



(b) Train and validation error for ophthalmology model with balanced data.

Figure 5.8: Illustration of train and validation error performance as a function of rules number.

Moreover, it is important to highlight that the decision tree generated by RuleFit has a depth of 4, so the limitation in the learning process and the difficulty in reducing the RMSE training error may be associated with the complexity of the problem and the low depth of the tree. For a fixed depth, the

number of rules does not improve the final model error and the arrows are constantly high in length, which means that more rules are not impacting and changing the final prediction. Figure 5.8 shows the constant error for depth 4 even with more additional rules.

Additionally, we developed the Interpretability Curve for a RuleFit of depth 5 and 6 to verify the error behavior. The performances were similar, with a low error reduction with the addition of rules. This behavior does not occur in the surgeon's model, as this one with less data is intrinsically easier to apprehend from a small structure.

In summary, Table 5.1 presents the selected ideal model based on the Interpretability Curve for each approach and algorithm studied. It is possible to notice that the value of the balanced data is consistently higher than the value associated with the imbalanced data, coherent results considering that it contains a more considerable amount of data in the training set.

Table 5.1: Model selection for each model approach and algorithm type for imbalanced and balanced data based on Interpretability Curve. The value represents the tree depth for XGBoost and the chosen number of rules for RuleFit models.

Algorithm	Data	General Approach	Ophthalmology Speciality	Surgeon ID 96440008
XGBoost	Imbalanced	9	7	3
	Balanced after GMM	11	9	3
RuleFit	Imbalanced	67	42	28
	Balanced after GMM	79	47	20

5.3.2 Performance Evaluation

In Section 2.7.3 was described the UBA tool designed for regression problems in non-uniform costs domains with the goal of developing evaluation metrics based on utility for regression due to the inadequacy of the standard errors metrics. The evaluation methodology gives more importance to points that are more difficult to predict, and, at the same time, it allows providing different costs to regions that represent underutilization or overutilization.

The idea is to assign different importance to each surgery prediction provided by the model. For example, if it is more important for CUF to predict more accurately shorter procedure times than longer surgeries, or if it is preferable to underestimate the time rather than overestimate, as it does not affect the following surgeries, probably it makes no sense to use metrics that give the same rate of importance to each forecast and thus we will base our metrics on the application's target.

In our case problem, the penalization costs factor (p) was set to 0.90 since opportunity costs are considered more severe than false alarms. In other words, overutilization is more costly than underutilization of operating rooms. Thus, when the estimated time for a given surgery is longer than the real one, these false alarms are less punished and lower the cost. On the other hand, if the predicted time is less than the real one since this can cause congestion in the flow of surgeries and operating rooms, the cost associated with this region is higher.

The "extreme" method is used for the relevance function since it interpolates points from the box plot statistics for extreme values. The function maps the domain of the destination variable to the interval

[0, 1], giving more relevance to the zone where we have a lower concentration of points, that is, to the adjacent values and rare values.

In Figure 5.9, we notice the relevance function for the surgeon's data. Surgeries with less than 50 minutes are considered of low relevance since they are part of the interquartile range (from the 25th percentile to the 75th percentile), containing 50% of the data. On the other hand, surgeries with more than 50 minutes will gradually have increasing relevance since the longer the time, the greater the difficulty in making the prediction. However, we expect to improve this problem after the generation of synthetic data by the GMM in minority classes.

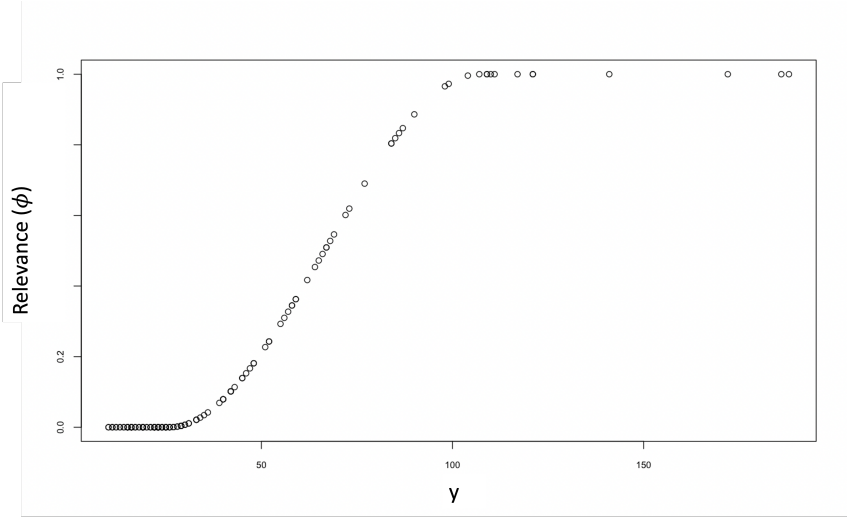
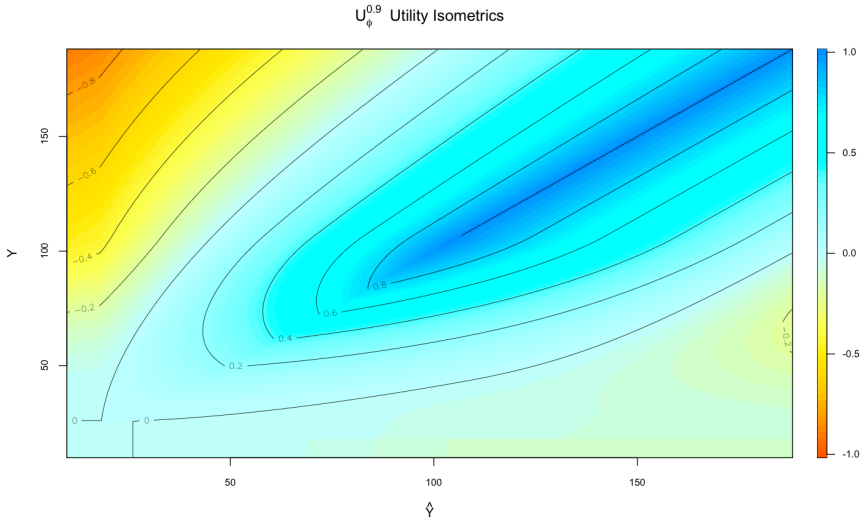


Figure 5.9: The relevance function, ϕ, for the prediction of surgeries times in case of surgeon ID 96440008.

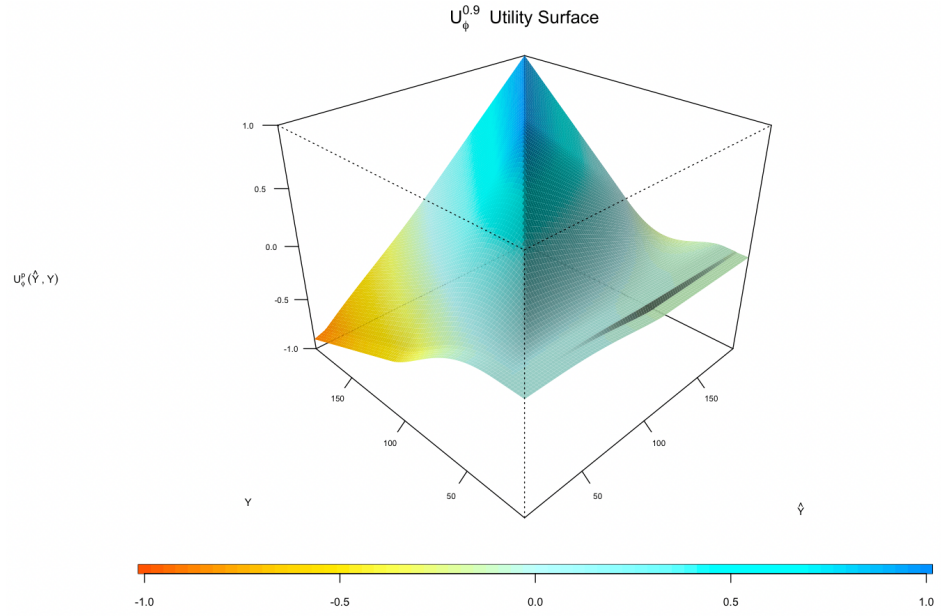
Figure 5.10 was designed and presented the surface, $U_{\phi}^{0.9}$, from the relevance function and its utility isometrics generated from surgeon data. In both Figures, y represents the true labels and $\hat{y}$ the predicted values. With a closer inspection of the utility isometrics of the surface, we can observe the lines that share the same utility value and better comprehend the costs and benefits. Firstly, when the true and predicted values are similar in the diagonal, we observe a higher positive utility for high surgical times, so these values are important to predict accurately. In the bottom right corner, the region of false alarms, the target value is less relevant, so any prediction for these values will not achieve a significant benefit but does not have any expensive cost itself. Nevertheless, surgeries with significant errors and in the overutilization region, left upper corner, will be more penalized than false alarms, so isometrics show the largest cost variation.

UBA metrics give us a better understanding of how models with balanced data can improve the prediction accuracy of models in minority class zones compared to models developed with imbalanced data. These include precision, recall, mean utility and Area Under the Receiver Operating Characteristic Curve (AUC-ROC).

Firstly, the mean utility is expected to have a small value because most of the points in the validation dataset belong to class 1. These points have a small utility score or even zero, so they do not influence and add a considerable value to the metric. Models developed with imbalanced data are expected to



(a) Utility isometrics



(b) Utility surface

Figure 5.10: An utility surface for the the prediction of surgeries obtained with the relevance function shown in Figure 5.9, with p = 0.90. $U_\phi^{0.9}$ exhibits higher costs associated to large errors bellow the actual real time and much smaller costs associated with false alarms.

have a small mean utility because it is a method that predicts classes 2 and 3 very poorly, so presents fewest points in the highest scoring area.

Furthermore, precision and recall are metrics that have higher values because their formulas only consider points above a certain threshold in the relevance function. Thus, if the threshold is 0.9, only points with a relevance score higher than 0.9 will be considered in the metric calculation.

AUC-ROC is a curve that measures performance for classification problems at various threshold settings. This one tells how much the model is capable of distinguishing between classes. AUC-ROC measures the power of discrimination for a binary classifier and its value varies between 0 and 1. If AUC-ROC is equal to 1, all positive and negative samples are predicted correctly. Contrary, a value of 0

means that all samples are wrongly classified.

Tables 5.2, 5.3 and 5.4 represent the results obtained with UBA library in imbalanced and balanced datasets for the general case, ophthalmology specialty and surgeon ID 96440008, respectively. The results of the two models, XGBoost and RuleFit, can now be compared to understand which algorithms can better compress predictions and better estimate surgeries belonging to the classes that we consider part of the minority class.

Table 5.2: Results of utility metrics for general model with imbalanced data and with balanced data from GMM.

Algorithm	Data	Mean Utility	Precision	Recall	AUC-ROC
XGBoost	Imbalanced	0.092	0.804	0.672	0.771
	Balanced	0.092	0.789	0.677	0.779
RuleFit	Imbalanced	0.099	0.784	0.686	0.778
	Balanced	0.113	0.788	0.704	0.791

Table 5.3: Results of utility metrics for ophthalmology specialty models with imbalanced data and with balanced data from GMM.

Algorithm	Data	Mean Utility	Precision	Recall	AUC-ROC
XGBoost	Imbalanced	0.071	0.759	0.661	0.765
	Balanced	0.071	0.751	0.676	0.772
RuleFit	Imbalanced	0.079	0.758	0.673	0.776
	Balanced	0.089	0.732	0.726	0.816

Table 5.4: Results of utility metrics for surgeon ID 96440008 model with imbalanced data and with balanced data from GMM.

Algorithm	Data	Mean Utility	Precision	Recall	AUC-ROC
XGBoost	Imbalanced	0.010	0.486	0.505	0.686
	Balanced	0.071	0.711	0.674	0.794
RuleFit	Imbalanced	0.092	0.725	0.704	0.800
	Balanced	0.092	0.697	0.722	0.806

In general, through the Tables presented, we can confirm that balanced models are an improved version of imbalanced models, particularly on observations with rare extreme values, and thus get better scores.

Firstly, recall is one of the most important metrics because it evaluates the y points considered with a high variance and are well predicted, thus it estimates how good the model is at verifying that a certain value belongs to minority classes. Hence, a higher recall value means that our model is predicting better points considered highly relevant. Analyzing the imbalanced models with the ones from balance data, this metric has a consistent improvement. According to the algorithm, RuleFit can better estimate the set of relevant points than XGBoost, showing a maximum value in the RuleFit algorithm with balanced data in all model approaches. Therefore, RuleFit can better compress the data.

Regarding precision, we do not constantly have a higher value for this metric in balanced models.

The value presented represents the proportion of points estimated as highly relevant by the model correctly predicted. In this context, most balanced models predict this parameter slightly lower than imbalanced models. Furthermore, the mean utility allows studying the performance of different models according to the relevance function and penalization factor. For the majority of the approaches, the mean utility value for balanced models is higher or equal to the imbalanced value. Lastly, AUC-ROC was consistently inferior for the imbalanced data in the totality of the models, meaning a better balanced model performance in identifying minority classes.

Ultimately, Table 5.5 presents the results of RMSE in minority classes for each balanced model approach and respective algorithm, establishing their comparison with CUF predictions. The RuleFit algorithm outperforms XGBoost on all approaches when analyzing the error in minority classes. In line with what was found through the UBA library, RuleFit reveals fewer difficulties in learning from imbalanced data.

Table 5.5: Summary of the RMSE in minotiry classes for each model approach and its comparison with CUF predictions.

Approach	RMSE in minority classes		
	CUF	XGBoost	RuleFit
General Model	62.09	60.89	57.39
Ophthalmology Model	31.59	29.54	26.62
Surgeon ID 96440008 Model	27.78	18.37	16.71

However, in an overall analysis and looking at the error of the validation set in its entirety, XGBoost presents better results, as we notice in Table 5.6. The validation set is used for algorithm selection.

Table 5.6: Summary of the RMSE in all classes for each model approach and its comparison with CUF predictions.

Approach	RMSE of all data		
	CUF	XGBoost	RuleFit
General Model	46.59	35.19	37.35
Ophthalmology Model	21.37	15.85	17.75
Surgeon ID 96440008 Model	17.01	10.83	11.46

Chapter 6

Results from an Operation's Perspective, Generalization Error and Conclusions

This Chapter describes the proposed interpretable prediction model for an operating room decision support system. In addition, the test set error is displayed to verify that the models are able to predict outcome values for previously unseen data accurately. Lastly, from the perspective of the final consumer, the hospital, we developed a cost function to explain more practically the benefits that the results of our proposal can bring.

6.1 Final Model and Generalization Error

With this thesis, we intend to develop an interpretable machine learning algorithm that can help CUF health professionals in estimating the time associated with each surgery and thus reduce the uncertainty and high errors correlated with the surgical times.

Earlier, with the analysis of balanced data, we concluded that the GMM technique improves the results of the models and makes the forecasting method less susceptible to overestimation, a parameter that we intended to reduce. As we explained in Chapter 5, we used GMM to produce synthetic samples however this implementation was not possible for all models of each approach, but for only one model of each due to time constraints. The duration of the synthetic samples generation, the identification of the three classes of each model, the design of the Interpretability Curve for the choice of the best model and the guarantee of the same percentage of each class within different sets are limitations that lead us to present the final model in a theoretical concept.

Hence, the final model presented to CUF will be an algorithm that uses the three different approaches depending on its data. If the initial features contain a doctor whose model already exists, that is, a doctor with more than 125 surgeries registered in CUF, the specific surgeon model will be used because

it presents the smallest error among the approaches. On the other hand, if a doctor has no trained model, we will move to the specialty models. Following the same reasoning, we will use this model if there is already a specialty model for the input specialty. Ultimately, if none of the above conditions are possible, the general model will be used for forecasting the time required. Algorithm 1 represents the entire process described but further studies are needed before incorporating machine learning-based decision support systems into clinical practice.

Algorithm 1 Operating room decision support system.

$F \leftarrow Features$
$S \leftarrow Specialty$
$N \leftarrow SurgeonNumber$
if N has more than 100 surgeries in training set **then**
    $SurgeonModel(F, N)$
else if S has more than 100 surgeries in training set **then**
    $SpecialtyModel(F, S)$
else
    $GeneralModel(F)$
end if

The algorithm advised is RuleFit because it allows the creation of a set of easily interpretable rules with different importances, being easy from an explanatory point of view its application in the hospital environment. Furthermore, it presented very interesting results with balanced data, even presenting better recall and mean utility values than XGBoost.

Unfortunately, although XGBoost is not a complete black-box like FNN, it seems complex to explain why it made a prediction. The SHAP values helped us to identify the relevant features and explore our models with confidence. However, RuleFit generates binary decision rules that are easily interpretable if the rules do not have more than 3 or 4 conditions. Therefore, its potential is more remarkable since RuleFit is an interpretable model and the discrepancy between the RMSE errors and within percentages in the different algorithms was not significant.

Moreover, generalization errors are critical to understanding the performance of machine learning models. However, as it was not possible to develop the final algorithm with all models and approaches with balanced data, it will not be possible to find this value. For this reason, we will only present the test error on generalization set for the three balanced models conducted to exemplify how we would have done it if it had been possible to develop all balanced models. Table 6.1 shows the generalization errors for each model in minority classes. The results meet the results present in Table 5.6 related to the validation set, so models are able to predict outcome values for a new data set accurately.

Table 6.1: Summary of the generalization error measured by RMSE in minotiry classes for each model approach and its comparison with CUF predictions.

Approach	RMSE in minority class	
	CUF	RuleFit
General Model	61.67	57.17
Ophthalmology Model	35.54	28.28
Surgeon ID 96440008 Model	29.34	18.21

6.2 Results from an Operation's Perspective

Explaining the results of our model using interpretable concepts for the hospital is fundamental since both end-users and the critical nature of the prediction require a certain amount of transparency. The results presented below can be discussed from a different perspective, closer to the point of view of specialist knowledge. We consider it essential to contemplate the relative cost of overutilization and underutilization activities, which are changeable costs that will depend on the hospital's perspective. Thus, to calculate the proposed solution's costs and establish a comparison with the cost previously supported by CUF, we generate a set of equations that take into account several factors.

The total cost for underutilization presented in Equation (6.1) considers the percentage of surgeries that falls 10% below the real time, the number of surgeries with undertime (#Under) divided by the total number of surgeries (#Surg), and the average loss of time in minutes. The assigned cost will be considered a cost per minute (C_{under}).

$$Underutilization_{cost} = C_{under} \times \frac{\#Under}{\#Surg} \times Avg(min)_{under} \tag{6.1}$$

Similarly, Equation (6.2) represents the total cost for overutilization, where now we consider the number of surgeries with overtime (#Over) and the average in minutes of this overuse.

$$Overutilization_{cost} = C_{over} \times \frac{\#Over}{\#Surg} \times Avg(min)_{over} \tag{6.2}$$

Finally, the total cost is given by

$$C_{total} = Underutilization_{cost} + Overutilization_{cost} \tag{6.3}$$

Both very long and very short time planning can lead to undesirable consequences for the organization of operating rooms. From the domain knowledge, we know that C_{over} is higher than C_{under} because the idle operating room produces underutilization costs, and indirectly, we are not maximizing the use of the room with a surgery that could be scheduled. In contrast, the overuse costs represent increases in the additional overtime payments and schedule reorganization costs [68]. Thus, assuming that the ratio between underutilization and overutilization costs is given by $C_{over} = r \cdot C_{under}$, C_{total} is defined by

$$C_{total} = C_{over}\left[r \times \frac{\#Over}{\#Surg} \times Avg(min)_{over} + \frac{\#Under}{\#Surg} \times Avg(min)_{under}\right] \tag{6.4}$$

The operational cost is given by the difference between our proposed model and CUF baseline, where we desire to obtain a cost reduction as presented in the following inequality (6.5). The chosen cost values will be based on the opinion of the hospital's stakeholders and will be kept as unknown variables as they may have slight variations depending on the purpose of its use. We will isolate these variables as much as possible to estimate their relationship by finding a minimum r value.

$$C_{model} < C_{CUF} \tag{6.5}$$

The inequality is applied to both CUF and model predictions to understand if our model outperforms the current model from an operational point of view. In Table 6.2 is also possible to observe the average time values in minutes considered for both overutilization and underutilization situations.

Table 6.2: Summary of the RMSE in each such model and its comparison with RMSE of CUF predictions.

Data	Predictions	Average overtime (minutes)	Average undertime (minutes)
All	General Model	37	27
	CUF	38	29
Ophthalmology	Ophthalmology Model	16	14
	CUF	24	16
Surgeon ID 96440008	Surgeon ID 96440008 Model	13	8
	CUF	21	9

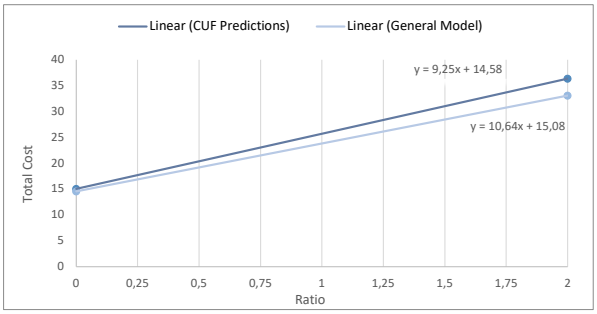
Consequently, the values obtained for r are presented in Table 6.3. The proposed solutions are cheaper than current standards when $r > r_{min}$. The acquired r_{min} values are considered small since our results overcome the current estimates for any hypothetical overutilization and underutilization costs, presenting a cost reduction compared to the CUF baseline.

Table 6.3: Ratio between preventive costs for each model in relation to CUF's baseline cost.

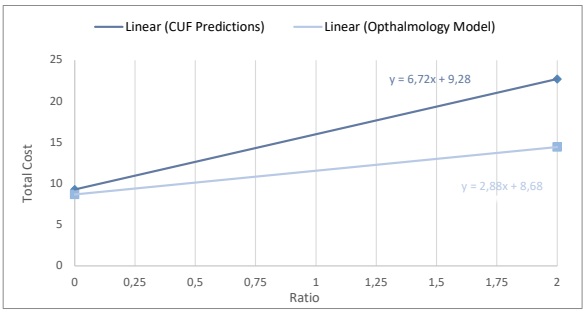
	r_{min}
General Model	-0.36
Ophthalmology Model	-0.16
Surgeon ID 96440008 Model	-0.078

Figure 6.1 shows the cost comparison of the baseline and proposed solutions, where the blue line is associated with the CUF baseline, and the light blue corresponds to the proposed solutions. We designed the graphs for a relationship between C_{over} and C_{under} at most twice, so r varies from 0 to 2. The objective is to get the light blue line below the current estimates line, with the largest possible gap for the proposed model to remain more cost-effective. As we can notice in the proposed solutions, we have a cost reduction compared to the baseline. Moreover, in line with what we found previously, specific models have a lower cost when compared to the general model.

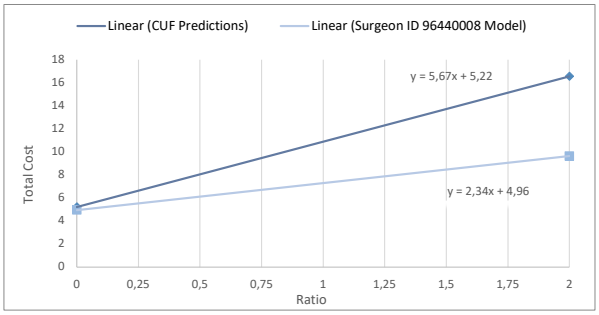
To conclude, it is essential to point out that in a deep analysis, possible indirect costs should also be analyzed and other metrics. The measurement tool developed was based on the type of results we obtained throughout the thesis.



(a) General proposed model compared to the baseline



(b) Ophthalmology specialty proposed model compared to the baseline



(c) Surgeon ID 96440008 proposed model compared to the baseline

Figure 6.1: Cost comparison of baseline and proposed solutions. Total cost in function of ratio. The proposed solutions are cheaper than the baseline in the three approaches. The objective is to be as much as possible below the current estimate line.

6.3 Conclusions

This thesis addressed the problem of inaccurate prediction of surgical procedures times through a data science and machine learning point of view. The major achievement of the work was the development of a model that allows better planning of CUF's operating rooms based on a reliable and interpretable prediction model for an operating room decision support system.

The interpretability concept has been extensively explored because the use of machine learning methods in this domain must be subject to the understanding of the reasons that support the prediction of a given algorithm by the decision-maker, clinician and patient. Without the model's accountability and oversight mechanisms, it is impossible to put into practice any of the systems described. Thus, a fundamental aspect of this proposal is the use of models with the highest precision, which are also explainable such as RuleFit.

The basis of the work was the set of historical data provided by CUF with information available from the last four years about its OR. However, this type of work that involves the use of real data gives rise to several difficulties. Naturally, unprocessed data has errors related to their collection. In the first phase, the challenge was identifying the features we intended to request from CUF and coding some features such as the patient ID and the CUF unit because they could not be delivered as raw data for data pro-

tection reasons. Secondly, the large number of different types of data did not facilitate the analysis. However, throughout the EDA, it was possible to investigate the features in detail and discard those that would be irrelevant to the work. As the last difficulty, identifying human errors related to data collection was not an easy task. However, we analyzed the data in detail to remove all human errors, or at least most of them.

To achieve the objectives of this study, we produced three different approaches: a general, specialty-specific and surgeon-specific models. In the latter, doctors were modeled independently. The models were trained and tested to understand which of the three models could be more precise and advantageous. As expected, surgeon-specific models were superior to specialty-specific models based on overestimation, underestimation and percentage of cases within, reaching an improvement of 6% in last metric with XGBoost and 2% with RuleFit. Compared to CUF's current state of the practice, the ability to predict surgeries within the 10% threshold was improved from 20% to 26% with RuleFit surgeon-specific models and 33% with XGBoost.

Subsequently, three models, one of each approach, were selected and their training dataset was enriched by sampling a GMM density trained on the minority data. Then, predictive models were trained and tested on original and enriched datasets to assess the models' performance in both conditions.

Finally, the results were analyzed using the UBA library containing metrics developed for regression problems in non-uniform cost domains. The enriched training datasets show better accuracy, recall, and AUC-ROC performance compared to the original dataset. RuleFit is the algorithm with higher performance based on utility-based performance metrics and RMSE of minority classes, an interesting finding from the perspective of interpretability.

An ideal proposed method would be a combination of the three different approaches described. The more specific the model, the more accurate the estimations, causing less congestion in the operating room organization. This interpretable prediction model for an operating room decision support system involves the use of balanced data for the learning process and would use RuleFit as a basis for its development.

Through this study, we offer a comparison of different techniques of ML for predicting the surgical durations based on a large dataset, exploring the impact of features on final predictions. Our research work highlights, on one hand, the value of applying machine learning techniques in the context of operating room management, and on the other hand, that imbalance-aware methods can make models more accurate, in particular for surgeries that are more difficult to predict.

6.3.1 Future Work

To conclude, we point out several future directions. As stated in Section 1.3.3, several approaches could be taken with the available data. However, we had to manage the work taking into account the time involved with data preparation since it represented a large proportion of our work. Therefore, the direction chosen was the predictions of surgical procedure times. Approaches related to scheduling, staff management and management of PACU capacity could also have been considered.

In a future work could be interesting to use not only all columns of procedures but also the probabilities of one procedure preceding another. Moreover, adding more data about doctor demographics, previous surgery, and other patient factors such as hypertension can influence the algorithm's decision-making and help estimate more accurate procedures times.

Regarding the balanced approach, we used a ML method to populate minority classes artificially. However, add real clinical data with the same features to enrich the dataset can yield better results. Additionally, using other hospitals to validate our ML approach could also be interesting to understand the model's behavior once we can not ensure the same performance if applied to a different organization.

Furthermore, although the total number of surgeries in 2020 is similar to the previous years, understanding the impact of COVID-19 on surgeries times, such as if there are more delays in surgeries due to more time involved in cleaning, can be an interesting study.

Lastly, regarding FNN, the approach was worse than interpretable models due to algorithm limitations with imbalanced data. Thus, to achieve better results on the model, it could be interesting to use a cost-sensitive neural network where the backpropagation algorithm can be updated to weigh errors in proportion to the importance of the class. Therefore, giving different costs to samples of each class will allow paying more attention to examples from minority classes.

Bibliography

[1] M. A. Bartek, R. C. Saxena, S. Solomon, C. T. Fong, L. D. Behara, R. Venigandla, K. Velagapudi, J. D. Lang, and B. G. Nair. Improving operating room efficiency: Machine learning approach to predict case-time duration. volume 229, pages 346–354.e3. Elsevier Inc., Dec 2019. doi: 10.1016/j.jamcollsurg.2019.05.029.

[2] V. Bellini, M. Guzzon, B. Bigliardi, M. Mordonini, S. Filippelli, and E. Bignami. Artificial Intelligence: A New Tool in Operating Room Management. Role of Machine Learning Models in Operating Room Optimization. *Journal of Medical Systems*, 44, Jan 2020. doi: 10.1007/s10916-019-1512-1.

[3] D. J. Lee, J. Ding, and T. J. Guzzo. Improving operating room efficiency. *Current Urology Reports*, 20, Jun 2019. doi: 10.1007/s11934-019-0895-3.

[4] A. Abedini, W. Li, and H. Ye. An Optimization Model for Operating Room Scheduling to Reduce Blocking Across the Perioperative Process. *Procedia Manufacturing*, 10:60–70, 2017. doi: 10.1016/j.promfg.2017.07.022.

[5] D. M. Laskin, A. O. Abubaker, and R. A. Strauss. Accuracy of predicting the duration of a surgical operation. *Journal of Oral and Maxillofacial Surgery*, 71(2):446–447, Feb 2013.

[6] J. P. Tuwatananurak, S. Zadeh, X. Xu, J. A. Vacanti, W. R. Fulton, J. M. Ehrenfeld, and R. D. Urman. Machine Learning Can Improve Estimation of Surgical Case Duration: A Pilot Study. Jan 2019. doi: 10.1007/s10916-019-1160-5.

[7] N. Rozario and D. Rozario. Can machine learning optimize the efficiency of the operating room in the era of COVID-19? *Canadian Journal of Surgery*, 63:E537–E529, 2020. doi: 10.1503/CJS.016520.

[8] R. Sarikaya, G. E. Hinton, and A. Deoras. Application of Deep Belief Networks for Natural Language Understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):778–784, 2014. doi: 10.1109/TASLP.2014.2303296.

[9] B. L. DeCost, H. Jain, A. D. Rollett, and E. A. Holm. Computer Vision and Machine Learning for Autonomous Characterization of AM Powder Feedstocks. *JOM 2016 69:3*, 69(3):456–465, Dec 2016. ISSN 1543-1851. doi: 10.1007/S11837-016-2226-1.

[10] L. Galway, D. Charles, and M. Black. Machine learning in digital games: a survey. *Artif Intell Rev*, 29:123–161, 2008. doi: 110.1007/s10462-009-9112-y.

[11] B. Cardoen, E. Demeulemeester, J. Beliën, B. Cardoen, E. Demeulemeester, and J. Beliën. Operating room planning and scheduling: A literature review. Mar 2010. doi: 10.1016/j.ejor.2009.04.011.

[12] K. Taaffe, B. Pearce, and G. Ritchie. Using kernel density estimation to model surgical procedure duration. *International Transactions in Operational Research*, 28(1):401–418, Jan 2021. doi: 10.1111/ITOR.12561.

[13] P. S. Stepaniak, C. Heij, G. H. Mannaerts, M. De Quelerij, and G. De Vries. Modeling procedure and surgical times for current procedural terminology-anesthesia-surgeon combinations and evaluation in terms of case-duration prediction and operating room efficiency: A multicenter study. *Anesthesia and Analgesia*, 109(4):1232–1245, 2009. doi: 10.1213/ANE.0B013E3181B5DE07.

[14] D. P. Strum, J. H. May, and L. G. Vargas. Modeling the uncertainty of surgical procedure times: comparison of log-normal and normal models. *Anesthesiology*, 92(4):1160–1167, 2000. ISSN 0003-3022. doi: 10.1097/00000542-200004000-00035. https://pubmed.ncbi.nlm.nih.gov/10754637/.

[15] D. P. Strum, J. H. May, A. R. Sampson, L. G. Vargas, and W. E. Spangler. Estimating times of surgeries with two component procedures: comparison of the lognormal and normal models. *Anesthesiology*, 98(1):232–240, Jan 2003. ISSN 0003-3022. doi: 10.1097/00000542-200301000-00035. https://pubmed.ncbi.nlm.nih.gov/12503002/.

[16] E. El-Darzi, R. Abbi, C. Vasilakis, F. Gorunescu, M. Gorunescu, and P. Millard. Length of Stay-Based Clustering Methods for Patient Grouping. *Studies in Computational Intelligence*, 189:39–56, 2009. doi: 10.1007/978-3-642-00179-6_3. https://link.springer.com/chapter/10.1007/978-3-642-00179-6_3.

[17] F. Dexter and J. Ledolter. Bayesian prediction bounds and comparisons of operating room times even for procedures with few or no historic data. *Anesthesiology*, 103(6):1259–1167, 2005. ISSN 0003-3022. doi: 10.1097/00000542-200512000-00023. https://pubmed.ncbi.nlm.nih.gov/16306741/.

[18] M. Van Houdenhoven, J. M. van Oostrum, E. Hans, W. Gerhard, and G. Kazemier. Improving operating room efficiency by applying bin-packing and portfolio techniques to surgical case scheduling. *Anesthesia and analgesia*, 105(3):707–714, Sep 2007. ISSN 1526-7598. doi: 10.1213/01.ANE.0000277492.90805.0F. https://pubmed.ncbi.nlm.nih.gov/17717228/.

[19] F. Dexter and R. D. Traub. How to schedule elective surgical cases into specific operating rooms to maximize the efficiency of use of operating room time. *Anesthesia and analgesia*, 94(4):933–942, 2002. ISSN 0003-2999. doi: 10.1097/00000539-200204000-00030. https://pubmed.ncbi.nlm.nih.gov/11916800/.

[20] C. H. Lee and H.-J. Yoon. Medical big data: promise and challenges. *Kidney Research and Clinical Practice*, 36(1):3–11, Mar 2017. ISSN 2211-9132. doi: 10.23876/J.KRCP.2017.36.1.3. http://www.krcp-ksn.org/journal/view.php?id=10.23876/j.krcp.2017.36.1.3.

[21] M. Fairley, D. Scheinker, and M. L. Brandeau. Improving the efficiency of the operating room environment with an optimization and machine learning model. *Health Care Management Science*, 22:756–767, Dec 2019. doi: 10.1007/s10729-018-9457-3.

[22] T. Chen and C. Guestrin. XGBoost : A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016. doi: 10.1145/2939672.2939785.

[23] O. Martinez, C. Martinez, C. A. Parra, S. Rugeles, and D. R. Suarez. Machine learning for surgical time prediction. *Computer Methods and Programs in Biomedicine*, 208:106220, Sep 2021. ISSN 0169-2607. doi: 10.1016/J.CMPB.2021.106220.

[24] N. Hosseini, M. Sir, C. Jankowski, and K. Pasupathy. Surgical Duration Estimation via Data Mining and Predictive Modeling: A Case Study. *AMIA Annual Symposium Proceedings*, 2015:640, 2015. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4765628/.

[25] E. R. Edelman, S. M. J. van Kuijk, A. E. W. Hamaekers, M. J. M. de Korte, G. G. van Merode, and W. F. F. A. Buhre. Improving the Prediction of Total Surgical Procedure Time Using Linear Regression Modeling. *Frontiers in Medicine*, 4(Jun):85, 2017. doi: 10.3389/FMED.2017.00085.

[26] J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008. doi: 10.1214/07-AOAS148.

[27] Y. L. Carel Schwartzenberg, Tom van Engers. The fidelity of global surrogates in interpretable Machine Learning. https://bnaic.liacs.leidenuniv.nl/wordpress/wp-content/uploads/papers/BNAICBENELEARN_2020_Final_paper_59.pdf [Accessed on 2021/09/03].

[28] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[29] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2017. arXiv: 1412.6980.

[30] D. Micci-Barreca. A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems. *ACM SIGKDD Explorations Newsletter*, 3(1):27–32, 2001.

[31] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Oversampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, Jun 2002. ISSN 1076-9757. doi: 10.1613/jair.953.

[32] C. Chokwitthaya, Y. Zhu, and A. Jafari. Applying the Gaussian Mixture Model to Generate Large Synthetic Data from a Small Data Set. Technical report. https://www.researchgate.net/publication/335662593 [Accessed on 2021/06/05].

[33] T. Zhang and X. Yang. G-SMOTE: A GMM-based synthetic minority oversampling technique for imbalanced learning, 2018. arXiv: 1810.10363.

[34] G. V. Ron Weiss. GMM covariances. Scikit-Learn. https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_covariances.html [Accessed on 2021/07/05].

[35] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[36] N. Moniz, P. Branco, L. Torgo, and B. Krawczyk. Evaluation of Ensemble Methods in Imbalanced Regression Tasks. Technical report, 2017. http://www.kdd.org/kdd-cup [Accessed on 2021/06/15].

[37] L. Torgo and R. Ribeiro. Precision and recall for regression. In *International Conference on Discovery Science*, pages 332–346. Springer, 2009. doi: 10.1007/978-3-642-04747-3_26.

[38] R. P. A. Ribeiro. *Utility-based Regression*. PhD thesis, Faculdade de Ciências da Universidade do Porto, 2011.

[39] P. Branco, R. P. Ribeiro, and L. Torgo. UBL: an R Package for Utility-Based Learning. Technical report, 2016. arXiv: 1604.08079v2.

[40] M. Paula Branco. Package 'UBL' Type Package Title An Implementation of Re-Sampling Approaches to Utility-Based Learning for Both Classification and Regression Tasks. Technical report, 2021. arXiv: 1604.08079.

[41] M. M. A. Filipe. Analysis of water quality variables of Cávado-Rabagão-Homem hydroelectric system. Master's thesis, Instituto Superior Técnico, 2020.

[42] A. Kent, M. M. Berry, F. U. Luehrs, and J. W. Perry. Machine literature searching VIII. Operational criteria for designing information retrieval systems. *American Documentation*, 6(2):93–101, Apr 1955. ISSN 1936-6108. doi: 10.1002/ASI.5090060209.

[43] P. Branco. Re-sampling Approaches for Regression Tasks under Imbalanced Domains. Master's thesis, Faculdade de Ciências da Universidade do Porto, 2014.

[44] H. Chen, R. Yang, R. R. Quinn, and Z. Butt. Interpretability of ML Models for Health Data-A Case Study The 3rd International Electronic Conference on Environmental Research and Public Health View project Blood Disorders View project Interpretability of Machine Learning Models for Health Data-A Case Study. Technical report, 2019. https://www.researchgate.net/publication/338487561 [Accessed on 2021/04/24].

[45] S. M. Lundberg, P. G. Allen, and S.-I. Lee. A Unified Approach to Interpreting Model Predictions. Technical report. https://github.com/slundberg/shap [Accessed on 2021/05/01].

[46] F. Doshi-Velez and B. Kim. Towards A Rigorous Science of Interpretable Machine Learning, 2017. arXiv: 1702.08608.

[47] L. Semenova, C. Rudin, and R. Parr. A study in Rashomon curves and volumes: A new perspective on generalization and model simplicity in machine learning, 2021. arXiv: 1908.01755.

[48] L. Breiman. Statistical Modeling: The Two Cultures. *Source: Statistical Science*, 16(3):199–231, 2001.

[49] G. Press. Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says. https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=6b0fda246f63#3643728a7f75 [Accessed on 2021/06/25].

[50] T. Marcelino. *Predicting real operating room occupation, an interpretable ML approach*. 2021. https://github.com/TeresaMarce/InterpretableML_ORoccupation.

[51] C. Wang, C. Deng, and S. Wang. Imbalance-XGBoost: Leveraging Weighted and Focal Losses for Binary Label-Imbalanced Classification with XGBoost, 2021. arXiv: 1908.01672.

[52] X. Shi, Y. D. Wong, M. Z. F. Li, C. Palanisamy, and C. Chai. A feature learning approach based on XGBoost for driving assessment and risk prediction. *Accident Analysis and Prevention*, 129: 170–179, Aug 2019. ISSN 00014575. doi: 10.1016/j.aap.2019.05.005.

[53] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv*, 51(93), 2018. doi: 10.1145/3236009.

[54] S. M. Lundberg and S.-I. Lee. Consistent feature attribution for tree ensembles, 2018. arXiv: 1706.06060.

[55] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Mach Learn*, 63, 2006. doi: 10.1007/s10994-006-6226-1.

[56] C. Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/ [Accessed on 2021/09/06].

[57] X. Zeng and M. A. T. Figueiredo. The Ordered Weighted ℓ_1 Norm: Atomic Formulation, Projections, and Algorithms, 2015. arXiv: 1409.4271.

[58] M. Figueiredo and R. Nowak. Ordered Weighted L1 Regularized Regression with Strongly Correlated Covariates: Theoretical Aspects. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 930–938, Cadiz, Spain, 09–11 May 2016. PMLR. https://proceedings.mlr.press/v51/figueiredo16.html.

[59] P. Bühlmann, P. Rütimann, S. van de Geer, and C.-H. Zhang. Correlated variables in regression: Clustering and sparse estimation. *Journal of Statistical Planning and Inference*, 143(11): 1835–1858, Nov 2013. ISSN 0378-3758. doi: 10.1016/j.jspi.2013.05.019.

[60] Y. Wu. Can't Ridge Regression Perform Variable Selection? *Technometrics*, 63(2):263–271, 2021. doi: 10.1080/00401706.2020.1791254.

[61] M. Nalenz. Horseshoe RuleFit-Learning Rule Ensembles via Bayesian Regularization. Master's thesis, Linköping University, Department of Computer and Information Science, Statistics, 2016.

[62] C. Rudin and J. Radin. Why Are We Using Black Box Models in AI When We Don't Need To? A Lesson From An Explainable AI Competition. Harvard Data Science Review, 2019. https://hdsr.mitpress.mit.edu/pub/f9kuryi8/release/6 [Accessed on 2021/09/23].

[63] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How Does Batch Normalization Help Optimization?, 2019. arXiv: 1805.11604.

[64] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, Jun 2013. PMLR.

[65] C. Zhang, K. Chen Tan, H. Li, and G. Soon Hong. A Cost-Sensitive Deep Belief Network for Imbalanced Classification. arXiv: 1804.10801v2.

[66] E. Štrumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowl Inf Syst*, 41:647–665, 2014. doi: 10.1007/s10115-013-0679-x.

[67] G. Stiglic, P. Kocbek, N. Fijacko, M. Zitnik, K. Verbert, and L. Cilar. Interpretability of machine learning-based prediction models in healthcare. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(5):e1379, Sep 2020. ISSN 1942-4795. doi: 10.1002/WIDM.1379.

[68] A. Fügener, S. Schiffels, and R. Kolisch. Overutilization and underutilization of operating rooms - insights from behavioral health care operations management. *Health Care Management Science*, 20(1):115–128, Mar 2017. doi: 10.1007/S10729-015-9343-1.

[69] R. P. A. Ribeiro. *UBA - Utility-based Algorithms*. 2012. https://www.dcc.fc.up.pt/~rpribeiro/uba/.

[70] L. Torgo. *Data mining with R: Learning with case studies*. Chapman and Hall/CRC, 2011.

[71] L. Torgo. *Functions and data for "Data Mining with R"*. 2013. https://mran.microsoft.com/snapshot/2017-02-04/web/packages/DMwR/index.html.

Appendix A

Technical Nomenclature

A.1 Data Dictionary

- Anesthesia: Type of Anesthesia
- Birth: Patient birth
- CUF_Unit: CUF Unit
- CUFPlannedDuration: Planned surgery duration
- Date_Surgery: Surgery date
- FLG_PROG_URG: If the surgery is urgent or programmed
- FLG_AMB_INT: If the patient is an outpatient or inpatient
- Doctor: Doctor Number
- HR_start_induction_ANEST: Beginning of induction of anesthesia
- HR_end_induction_ANEST: End of induction of anesthesia
- HR_end_Room: Exit from the room
- HR_start_Room: Entrance to the room
- HR_start_Surgery: Beginning of surgery
- HR_end_Surgery: End of surgery
- HR_start_RR: Entrance to the recovery room
- HR_end_RR: Exit to the recovery room
- Gender: Patient gender
- I1/I2/I3/I4/I5/I6: Procedures performed in each surgery. Correspond to TOM codes

- ID_Patient: Unique key for each patient transverse to all CUF units
- Local_Patient: Unique key to encode each patient in each unit
- N_REG_OPER: Operative Registry Number, which is an internal unit code
- OR: Operating room
- RealRoomDuration: Real duration inside of room
- RealSurgeryDuration: Real surgery duration
- Room: Room inside operating room. An operating room can have several rooms
- Speciality: Surgery Speciality
- TimeRecoveryRoom: Time inside recovery room

A.2 Types of Anesthesia

- AE: Epidural Anesthesia
- AG: General Anesthesia
- ALOC: Local Anesthesia
- ALP: Local Anesthesia / Plexus
- AP (child-birth): without anesthesia
- APLE: Plexus Anaesthesia
- AS: Sequential Anaesthesia
- PAE (child-birth): Epidural Anesthesia
- PAG (child-birth): General Anesthesia
- PAR (child-birth): Spinal anesthesia
- RAQ: Spinal anesthesia
- SED: Sedation

Appendix B

Code

This appendix highlights some practical aspects of the code to fully understand the notebooks presented in the repository, publicly accessible in [50]. This project is coded using Python and R.

B.1 Code Organization

B.1.1 Python

- **Notebooks**
  - Exploration Data Analysis
    - ⇒ ***DataAnalysis.ipynb***: contains all the data analysis, where main dataset characteristics are summarized.
    - ⇒ ***FBprophet.ipynb***: time series analysis where we can observe annual, weekly and daily seasonality.
  - Algorithm Implementation
    - ⇒ ***Datapreparation.ipynb***: separation of data into the three different approaches.
    - ⇒ ***XGBoost_implementation.ipynb***: contains the code responsible for XGBoost Implementation in all approaches, including tuning, training, and test error analysis.
    - ⇒ ***RuleFit_implementation.ipynb***: contains the code responsible for RuleFit Implementation in all approaches, including tuning, training, and test error analysis.
    - ⇒ ***FNN_implementation.ipynb***: contains the code responsible for FNN Implementation in all approaches, including tuning, training, and test error analysis.
  - Model Implementation (Balanced vs Imbalanced)
    - ⇒ ***SynteticSamples.ipynb***: contains the code to generate synthetic samples in minority classes through GMM for the three models (one of each approach).

⇒ ***RuleFit_General.ipynb***: contains the code responsible for RuleFit Implementation in imbalanced and balanced datasets of all data. Their Interpretability Curve is drawn to select the suitable model.

⇒ ***RuleFit_Specialty.ipynb***: contains the code responsible for RuleFit Implementation in imbalanced and balanced datasets of the ophthalmology. Their Interpretability Curve is drawn to select the suitable model.

⇒ ***RuleFit_Surgeon.ipynb***: contains the code responsible for RuleFit Implementation in imbalanced and balanced datasets of the Surgeon ID 96440008. Their Interpretability Curve is drawn to select the suitable model.

⇒ ***XGBoost_General.ipynb***: contains the code responsible for XGBoost Implementation in imbalanced and balanced datasets of all data. Their Interpretability Curve is drawn to select the suitable model.

⇒ ***XGBoost_Specialty.ipynb***: contains the code responsible for XGBoost Implementation in imbalanced and balanced datasets of the ophthalmology. Their Interpretability Curve is drawn to select the suitable model.

⇒ ***XGBoost_Surgeon.ipynb***: contains the code responsible for XGBoost Implementation in imbalanced and balanced datasets of the Surgeon ID 96440008. Their Interpretability Curve is drawn to select the suitable model.

- **Classes**

  ⇒ ***RuleFit_customized.py***: contains the required functions to implement the customized RuleFit package with detailed modifications presented in Section 4.3.2.

- **Models**

  – Model Implementation (presented in Chapter 4)

    ⇒ ***Models Folder***: there is a ”Model” folder in the repository of each algorithm, which contains all models to generate predictions. RuleFit models are saved into *.sav* files, FNN models into *TensorFlow SavedModel* format and XGBoost models into *.model* files.

  – Model Implementation: balanced vs imbalanced (presented in Chapter 5)

    ⇒ ***Models Folder***: there is a ”Model” folder in the repository of XGBoost and RuleFit, which contains all models to generate the Interpretability Curve. There is a folder for each general, ophthalmology and surgeon approach. RuleFit models are saved as *RuleFil.sav* files and their name include the number of rules. XGBoost models are saved into *.model* files and their name include the depth of the model.

B.1.2 R

- RuleFit Performance

⇒ ***UBA_general_RuleFit.R***: contains the required code to apply the utility-based performance metrics (precision, recall, utility and AUC-ROC) in predictions from imbalance and balanced general models.

⇒ ***UBA_speciality_RuleFit.R***: contains the required code to apply the utility-based performance metrics (precision, recall, utility and AUC-ROC) in predictions from imbalance and balanced ophthalmology models.

⇒ ***UBA_surgeon_RuleFit.R***: contains the required code to apply the utility-based performance metrics (precision, recall, utility and AUC-ROC) in predictions from imbalance and balanced Surgeon ID 96440008 models.

- XGBoost Performance

⇒ ***UBA_general_XGBoost.R***: contains the required code to apply the utility-based performance metrics (precision, recall, utility and AUC-ROC) in predictions from imbalance and balanced general models.

⇒ ***UBA_speciality_XGBoost.R***: contains the required code to apply the utility-based performance metrics (precision, recall, utility and AUC-ROC) in predictions from imbalance and balanced ophthalmology models.

⇒ ***UBA_surgeon_XGBoost.R***: contains the required code to apply the utility-based performance metrics (precision, recall, utility and AUC-ROC) in predictions from imbalance and balanced Surgeon ID 96440008 models.

- Relevant Packages

⇒ ***uba_0.7.7.tar.gz***: contains performance metrics for regression algorithms in non-uniform costs domains available in [69].

⇒ ***DMwR_0.4.1.tar.gz***: package necessary to run UBA package. Includes functions from the "Data mining with R: Learning with case studies" [70] and it is available in [71]