INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA

Sistema para a otimização do escalonamento do bloco operatório

DIOGO FERNANDO FERREIRA COELHO
Outubro de 2021

POLITÉCNICO
DO PORTO

System for surgical block schedule optimization

Diogo Fernando Ferreira Coelho

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Information and Knowledge Systems

Supervisor: Dr. Afonso Pedrosa

Porto, October 24, 2021

Dedicatory

To my loving family, that carried me throughout some of the most difficult moment of my life.

Abstract

Proper distribution and utilization of operating rooms is one of the biggest factors when combating the ever growing waiting lists for surgical interventions. In this ecosystem, the incorrect prediction of a procedure's duration will imply the remaining scheduled procedures, further more when this prediction is an underestimation. This problem is is exacerbated by the sheer amount of different interventions with their specificities and conditions.

Tackling this question, we developed an application running along side the main surgery schedule of Centro Hospitalar de São João, in charge of applying regression algorithms to better calculate the expected surgery duration. With these, we were able to apply a scheduling algorithm that produces a viable surgery table.

Our final implementation was able to work independently of human interaction, producing a possible alternative to the manual methods common on these situations.

Keywords: Supervised Learning, Surgery Scheduling, Hyper-Parameter Optimization, General Population Surgery

Resumo

A distribuição e utilização adequada dos blocos operatórios é um dos principais fatores quando se visa diminuir o constante aumento das listas de espera para intervenções cirúrgicas. Neste ecossistema, uma má previsão da duração de um procedimento trará problemas na restante calendarização dos procedimentos, ainda mais quando esta previsão é menor que o tempo real de cirurgia. A resolução deste problema é exacerbada pela quantidade de diferentes intervenções, com as suas especificidades e condições.

Para fazer face a esta questão, desenvolvemos uma aplicação que corre ao lado do calendário principal do Centro Hospitalar de São João, encarregado de aplicar algoritmos de regressão para melhorar o calculo da duração da cirurgia. A nossa implementação final foi capaz de trabalhar independentemente da interação humana, produzindo uma alternativa apossável aos métodos manuais comuns nestas situações.

Acknowledgement

I would like to be briefly thank everyone that helped me in the realization of this work. This project could not be done on one man's back alone. I would like to pay special tribute to:

Professor Carlos Ferreira and João Pedroso, my orientators for helping me thorough this job. My only regret is to not have taken more advantage of their knowledge.

Professor Susana Vargas, anaesthetist and chief of the main operation block in Hospital São João, for all she taught me, both from a professional and friendly standpoint. I hope she sees in me a friend, as I see in her.

Dr. Afonso Pedrosa, chief of the Business Intelligence Service in Hospital São João, for he guided me through my professional growth, always understanding of my medical limitations.

And last but not least, to my colleagues, Pedro 'Nobita' Pinho and João 'Bazalto' Rocha, for they were my bedrock when I wanted to give up, and my support when I finally came back to this project.

Contents

List of Figures

List of Tables

List of Source Code

List of Acronyms

BI	Business Intelligence.
CHUSJ	Centro Hospitalar Universitário de São João.
CLI	Command Line Interface.
GBRT	Gradient Boosted Regression Trees.
LIC	Lista de Inscritos Cirurgia.
ML	Machine Learning.
MLR	Multiple Linear Regression.
MSE	Mean Squared Error.
MSSQL	Microsoft SQL.
MSSS	Master Surgery Speciality Schedule.
ODBC	Open Database Connectivity.
OR	Operating Room.
RFR	Random Forest Regressors.
RMSE	Root Mean Squared Error.
SMTP	Simple Mail Transfer Protocol.
SSL	Secure Sockets Layer.

Chapter 1

Introduction

In this chapter, we will be presenting the reader, with a short brief on the developed work, and, the dissertation that culminated from it.

1.1 Context

This document is destined to evidence the work of Diogo Fernando Ferreira Coelho, student nº. 1160722, at the Informatics Department of the Instituto Superior de Engenharia do Porto (ISEP), Porto, Portugal, for the curricular unit of Thesis / Dissertation / Internship, of the Master's Degree in Information and Knowledge Systems. The submission of the dissertation is one of various requirements to attain the degree of Master of Science.

This project was developed for the hospital Centro Hospitalar Universitário de São João Centro Hospitalar Universitário de São João (CHUSJ), located in Porto, Portugal, as a internship, in addiction to the student's current day-job on this institution.

1.2 Problem's Description

On the beginning of the twentieth's-first century, the World Health Organization started tracking surgeries performed around Europe (WHO 2016). That value has been increasing ever since, reaching around 60000 in Portugal alone, in 2020 (OdM 2020). This is, to be expected, after all, improvements in technology and tools, leading methodologies, new medical discoveries and a further understanding on ailments, would eventually result in the increase of number of treatments, and the quality of which.

This, however, presents a complicated problem. Patients that, in the past, would be denied treatment, as these were merely not possible, now occupy the limited number of operating rooms available, putting pressure on an already stressed structure. Public health systems are specially vulnerable (Fehlberg et al. 2019), as most of the populace might not have the financial resources to search for care in a private hospital (Hoel and Sæther 2003). This over saturation of the public infrastructure results in the exponential growth of waiting times for surgeries, specially those of minor importance.

The crux of the problem, lies on the complexity of these types of treatment. Surgeries are, in fact, some of the most complex and diversified activities done in an hospital, requiring specialized equipment, facilities and staff, making the solution of 'throwing money at the problem' lackadaisical at best. In 2015, Portugal spent around 36 000 000€ on the usage of the operating room alone (SNS 2021). Operating rooms one of the biggest money sinks, present on Public Health Systems, and, as such, good usage of these and efficient planning go a long way.

Surgical scheduling is the act of appointing a number of interventions, to a group of available operating rooms, in a surgical block, based in factors, like nature of intervention, priority, waiting time on queue (Lista de Inscritos - LIC), between others. During this process, one of the most important phases, consists in forecasting the time needed to perform said surgery, allowing managers to chart the occupancy of a operating room. But, these estimations, are based on hundreds of factors, and errors do occur, wasting time that could be use to perform other interventions. When a surgery goes 'overtime', every next one gets pushed back, at risk of needing to be canceled. If a surgery does not occupy the full extent of the vacancy allowed, the medical team must wait, as the next patient will not be ready for the intervention.

This problem affects, not only the institutions on cause, but the patients, subject to increasingly larger waiting times, regular rescheduling, unnecessary exams and fasting causing discomfort (Nygren 2006).

1.3 Objectives

Considering our problem, we devised a list of three objectives, the fulfilment of which, will present a solution to the issue in study. As such, we propose:

- Creating an autonomous system, capable of estimating the necessary time for a specified intervention to take place, based on available information on the patient, medical team and intervention to be performed.
- With this knowledge, build an optimized schedule, to be presented as a suggestion to the manager/personnel in charge.
- *A posteriori*, gather information between prediction and elapsed time, in order to improve the algorithms implemented, and, to brace for any significant changes, on the clinical reality of the hospital.

As a secondary objective, it would be advantageous to show, in a simplified manner, how our system arrived to its conclusions.

1.4 Contributions

The final product, described in this dissertation, consists in a intranet-based platform, where both the LIC and the duration predictions are presented. With these, automatic scheduling options can be built. This system, by nature, serves only as a information dashboard, being fed data through a back-office capable of said calculations, triggered by request.

Thus, we believe to be working towards a scheduling scheme, less prone to instability or errors, which may incur a large impact on operation room occupancy, a relevant improvement on the hospital's production and considerable enhancement on patient care quality.

1.5 Approach

This project consists in the production of a information tool, inserted in the current technological infrastructure of CHUSJ. This final product is divided in:

- Knowledge presentation via an intranet-based dashboard;
- Extract Transform Load (ETL) processes on the hospital's data infrastructure;
- Intervention's duration prediction via machine learning algorithms;
- Optimized schedule generation;

For the dashboard, with which the final user interacts, we iterated on already implemented version of LIC, on the hospital's Business Intelligence (BI) server. The data used for our solution, is prepared via ETL processes, originating from Microsoft SQL Server, Oracle-SQL and data warehouses for OLAP cubes. The intervention's duration predictions are based on an ensemble approach of machine learning, grouping by surgery speciality. At last, process mining algorithms are integrated for the generation of the optimized schedule.

1.6 Dissertation's Structure

This dissertation has been segmented in 6 chapters, each being a self contained documentation of the topic at hand:

- **Chapter 1 - Introduction**: The current chapter, serving as an introductory view of the problem, and what we propose, to achieve a viable solution.
- **Chapter 2 - Context and State of the Art**: In this chapter, we scrutinize the problem more thoroughly, going over case-specific terminology, and closing on a study on academic literature and available technology.
- **Chapter 3 - Value Analysis**: In this chapter, we produce a detailed value analysis of this project, probing the benefit our solution might bring to fruition.
- **Chapter 4 - Solution Design**: In this chapter, we document our design plan, based on the best practices of Software Engineering.
- **Chapter 5 - Solution Implementation**: In this chapter, we present the steps taken, to materialize our proposed solution, recounting any step of the development procedure.
- **Chapter 6 - Solution Appraisal**: In this chapter, we demonstrate a detailed case study, in order do properly evaluate the capabilities of the produced tool.
- **Chapter 7 - Conclusion and Future Work**: At last, here we review the project as whole, going through what, in our eyes, was achieved, concluding with a small proposition of future work, that could build on the foundation, that is this dissertation.

Chapter 2

Context and State of the Art

2.1 Context

The human body is often compared to a complex machine, segmented in distinct, yet complementary systems, working towards what we, as people, would call normalcy. Yet, we do not live in a perfect scenario, and as such, these machines might need maintenance.

Throughout medical history, the complexity of ailments that afflicted us kept increasing, not because our environment changed, but as a result of, our ever growing understanding of the human body. However, this complexity also expressed itself on the treatments performed. Surgical procedures are one of the most evident cases of this complexity, exponentially growing throughout the advance of medical knowledge (Cardoen, Demeulemeester, and Beliën 2010). The more we know how to intervene on an ailment, the more unfeasible procedures become available or even common place.

These procedures are highly specific, and, as such, need highly specific professionals, instruments and locations to be performed. Focusing our efforts on the location, the operating rooms Operating Room (OR), are some of the most expensive facilities to build and run in an hospital. In America, one such facility can cost between 29$ to 80$ per minute (Erekat et al. 2020). Due to this, the vacancy of ORs is limited, and its usage is deeply monitored.

This stands as the essence of the problem in analysis, the ever growing number of surgical procedures, competing with the limited number of available ORs.

To better understand this, we underwent a two week-long study, over on CHUSJ's central operating block, where we analysed the typical workflow of the unit. Throughout this endeavour, we inquired both administrative and medical personnel, realizing both our requirements engineering process, detailed on 4.1, and our problem analysis.

From this study, three major obstacles arose:

- The scheduling schemed used, is based on organizing various 30 minute intervals, into a time block, originating of the surgeon's own discretion; We observed how these were grossly misaligned with the reality inside the OR, to the point, when inquiring medical staff, we saw that "finishing (the afternoon shift) one hour late, is just a known fact";
- Different surgeons have different preferences. When operating there is an assortment of tools needed, yet, these are disposed in consideration of who is performing the treatment. This ambiguity is most prevalent, in the beginning of the morning and afternoon shifts, as surgeons might not be present on the preparation of the OR. Throughout our two week-long study we noticed a intrinsic delay, recurring on the first intervention of each day, between 30 minutes to 1 hour;

- When a patient is proposed for intervention, depending on the complexity of treatment, extra exams must be performed. These are fundamental, and in cases they happen to be left out, the intervention will be canceled, leaving the room unoccupied;

After a long bartering process, with the administrative chief of the central block, we chose to tackle the scheduling act, as a feasible angle of attack, on our primary concern, increasing the surgical production of CHUSJ.

Currently, scheduling optimization via machine learning and process mining algorithms, is an highly sought after study topic, amongst scholars. The dynamism of factors affecting an intervention duration, multiplied by the sheer number of different treatments available, makes most solutions case-specific, to the reality of the institution in consideration. Yet methodologies may be cross-examined and adapted to our use case.

It's our objective, with this dissertation, to study the state of the art on this area, and, to propose viable solution, capable of assisting medical staff and coordinators, on ORs scheduling, increasing CHUSJ production while improving the number of treatments the hospital can undertake, improving its patients' lives.

2.1.1 Requirements

Our solution is to be implemented in an highly insightful, yet older, less computer literate and extremely time-constrained community. As such, this project must strongly adder to both KISS and POLA principles. KISS, from 'Keep It Simple, Stupid' (Milicchio 2007), expresses that a system is easier to interact with, when kept simple. POLA, or the 'Principle of Least Astonishment'(Isaksen and Bertacco 2006), refers to how a system, more precisely, the user interface behavior, must be kept in check with the users expectations, in order to avoid alienation. Failing to apply these concepts may negatively affect the adoption of our product.

With this in mind, our proposition must keep a rather simplistic set of requirements, from the client-side, adhering to what these users are already introduced to. In this case, the common user, consisting in any medic, anesthetist, nurse or coordinator, with access to LIC. Known the users, we gather the following use cases:

2.1.2 Project's Restrictions

For the production of a viable solution, some restrictions were mandated by the hospital, consisting in technical limitations, the neglection of which, would result on the termination of the project. These restrictions are:

- The usage of technology and infrastructure inside the Microsoft 365 Ecosystem, present on Appendix A, already present in the hospital;
- The solution must be implemented in a interface, already published on the hospital's intranet, which might not be able to take user inputs;
- All of the data storage and processing must be done on-site;
- The hospital has adopted ICD-10, however, this transition only started on July of 2020, and the most common/numerous surgeries haven't yet shifted from ICD-9. This makes historical values, difficult to use, and current values, difficult to compare and process, between each other;

2.2 Major Concepts

2.2.1 Medical Concepts

Surgical procedure

The World Health Organization defines a surgical procedure, as a treatment for a surgical condition, this being a "...condition that requires suture, incision, excision, manipulation, or other invasive procedure that usually, but not always, requires local, regional, or general anesthesia" (WHO 2021). This definition is rather broad, yet captures most of the key points needed to distinguish between treatments. Throughout this work, we refer to this interpretation, with the added condition, of treatment performed in a specialized theater.

These treatments are usually segmented various moments, of which we focus in two phases, largely determining intervention duration: the application of anesthetics, and the intervention itself, consisting from the first incision until the last suture.

Operating Room

Operating rooms, are specialized chambers for the purpose of surgical procedures. These contain tooling required for the treatments usually performed, meaning, an OR will usually perform interventions for a single medical speciality or the ones sharing most equipment. Key word, usually; in cases of necessity, the disposition of the room may be altered or new tools be brought.

Lista de Inscritos Cirurgia

The Lista de Inscritos Cirurgia (LIC) is the definite surgery waiting list, used by CHUSJ, structuring all planned interventions, on a priority queue. Amidst many metrics, four pertinent items should be noted:

- **Surgical speciality** - The medical speciality responsible for evaluating, scheduling, preparing and executing a surgical procedure. This key value is used to cross the scheduling act with the Surgery Speciality Master Schedule.
- **Priority** - The urgency group to which the patient/procedure is situated. As priority increases, the metric 'Time to Leave' experiences an exponential cutback.
- **'Time to Leave'** - A countdown, in days, to the legal date limit, a patient can remain on waiting list. In cases where this deadline isn't fulfilled, the patient may request treatment in the private health sector, at the expense of the public establishment. This metric has a max value of 365 (three-hundred sixty-four) days.
- **Diagnosis and procedure codes** - Depending on the version of ICD-*x*, numerical or alpha-numerical codes, ranging from three to seven characters, clearly defining the patient's pathology or the correspondent treatment to be scheduled. While both diagnosis and procedure codes appear similar, they do not share any structural/logical connection.

When a patient enter the LIC, the four metrics described, define how much time the hospital has to give a 'response' to the surgery request, this being the actual schedule of an intervention. This means that, even though there is an hard limit on how long a patient may wait, they may need to wait longer as the act of scheduling is the one monitored.

Master Surgery Speciality Schedule

As there are more surgical specialities, then rooms to perform said surgery, the Master Surgery Speciality Schedule maps who can use the operating rooms, on a OR/day of the week/hour grouping. Usually, there are two Master Surgery Speciality Schedule (MSSS) alternating on a weekly basis.

These tables (example on appendix C), divide every OR, for every day of the week, in two time frames, morning and afternoon. Each slot is allocated to a surgical speciality, split in blocks of 30 minutes, between 08:00 - 13:00 and 14:00 - 19:00.

ICD-*x*

ICD, standing for International Classification of Diseases, is a categorization tool supported by the World Health Organization. This system started as a tool for billing purposes, for insurance companies and hospitals, being later adopted as the major classification tool for treatments. To be noted that, in the beginning, ICD did not discern procedures, until the release of Volume 3 of the ICD-9-Clinical Modifications revision. The *x*, after the initialism ICD, represents the version of the codification in use. In this dissertation, we'll be working with both ICD-9 and ICD-10, as these are both still in usage.

- ICD-9 - Published in 1978 (WHO 1978), this version of the ICD consists in seventeen thousand codes, up to six characters long, where the first digit is alphanumeric and the following values are numeric. In the numeric portion of the code, there may be introduced a dot, separating the code in two parts, the leftmost one classifying the ailment/procedure, the rightmost part relating extra information, of finer granularity, about the ailment/procedure. The following example consists in the diagnosis of pneumonia, by the virus SARS:

Ailment or Procedure		Detail
480	.	31
Viral Pneumonia	Separator	SARS associated coronavirus

Table 2.1: Example of ICD-9 structure

  As shown, the leftmost section of the code identifies the disease, while information after the separator relates extra information, based on the first section. This equates to the code '.31' having several meanings, depending on the disease it relates to.

  Even though, ICD-9 as been replaced by ICD-10 in the summer of 2020, all the data that has been coded in the older version, still remains, because, most of the times, there is no direct or 100% correct translation between the two coding systems. The consequence of this, is that, even though deprecated, ICD-9 must remain in usage until, every patient and/or procedure scheduled before July of 2020, either is completed or expires.

- ICD-10 - Beginning in 1983 (WHO 2011), this version of the ICD consists in close to one-hundred thousand different codes, in two variants, the ICD-10-CM containing data on the patients ailments, and the ICD-10-PCS categorizing any treatment there may used. This coding system is built with 7 alphanumeric characters, read left to right, where each value consists in a different level of granularity on the topic. The following example consists in the surgical procedure of bypassing the right axillary artery, into the upper arm artery, through a synthetic substitute:

Category			Etiology, anatomical site and severity			Extension (Extra Detail)
Section	Body System	Root Operation	Body Part	Approach	Device	Classifier
0	3	1	5	0	J	2
Medical & Surgical	Upper Arteries	Bypass	Axillary Art., Right	Open	Synthetic Substitute	Upper Arm Art., Bilateral

Table 2.2: Example of ICD-10-PCS structure

This coding system is divided in three parts, where any character's meaning, from 'Extension' is based on 'Etiology, anatomical site and severity', which in turn, is based on 'Category'. As in ICD-9, the parts that build an ICD-10 do not have a hard-coded value or significance. This high level of detail makes it impossible to accurately translate between ICD-9, which is more abstract, and ICD-10. Lets take a step back, if we ponder over this coding system, ignoring the explicit categories, a pattern emerges. Splitting the code in two, by the fourth value (Body Part), we obtain the specific procedure, down to the body part, while the last three spaces describe the "how" a procedure is performed.

2.2.2 Technical Concepts

Machine Learning

Machine Learning Machine Learning (ML) (Dey 2016) is an subsection of Artificial Intelligence. It consists on the study, creation and usage of algorithms capable of discerning patterns, based on given examples. These examples may be labeled or unlabeled, originating supervised or unsupervised learning, respectively. These algorithms, using the patterns found on their learning processes, can then be applied to predict, the value a certain variable might take, based on the already available information.

There are numerous different types methods, some of the most relevant being:

- **Regression algorithms** - Consist in finding and mapping statistical relationships between different variables, to find a pattern on the values;
- **Instance-based algorithms** - These methodologies, use examples of already classified instances, to predict the class of new values, avoiding explicit generalization;
- **Decision tree algorithms** - Decision trees build a model of decision forks, that when flowed through output the most probable class. One strong advantage these models have, is the high rate of readability, meaning, it's possible to follow how a classification was generated;
- **Clustering algorithms** - Algorithms responsible to find groups of instances with a high commonality, grouping them into classes;
- **Regression tree algorithms** - Mixing the statistical evaluation of data, with the added logical forks of decision trees, regression trees enable us to find the numerical patterns of a dataset, with the added dimension of complex information and arguably high readability;

- **Ensemble algorithms/approach** - Some models are better suited for different tasks, and with ensemble methods, we may build a complex model of weaker/simpler models, combining their singular conclusions, into a unique prediction;

In the scope of this project, we will focus on regression, regression trees, and and ensemble approaches, more precisely, Multiple Linear Regression (MLR), Random Forest Regressors (RFR) and Gradient Boosted Regression Trees (GBRT).

Multiple Linear Regression

Expanding on the single dependent/independent variable limitation of the simple linear regression, MLR enables the regression of an unknown value, given x inputs, and the formulation of a mathematical equation capable of being represented by a regression surface. This can be described in its simplified form as:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ...\beta_p x_p + \varepsilon \tag{2.1}$$

Where Y is the dependent variable, in essence, the value to be regressed, p is the number of independent variables and $\beta_n x_n$ is the linear function formed from the β slope and x value of the p-th independent variable. While, as all mathematical regression algorithms require numerical values, categorical data (i.e. text describing a class) may also be used, with the adoption on one-hot encoding.

Random Forest Regressor

RFR is an ensemble algorithm where, as the name may implies, a multitude of different decision trees average out a concise value. Decision trees by themselves are valuable ML algorithms, enabling the classification or regression of a instance, based on logical rules, yet, when presented with highly complex problems, they tend to grow bloated, being both unreadable and overfitted to their training models. This limits their utilization, as the generated model may be too specific to be applied to any real case of classification/regression.

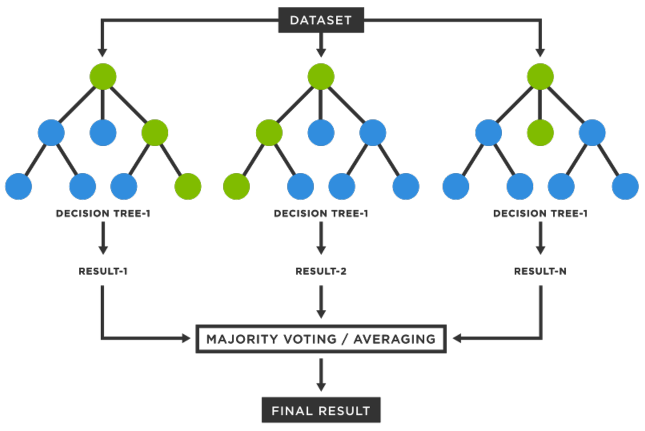


Figure 2.1: Visualization of a random forest (source: Kumar 2021
)

RFR, generalizing, uses n number of trees instead, where the output is not the outcome of any single tree, but the majority vote of the "forest" (in case of classification), or its average numerical value (in case of regression). These trees are generated from training sets, created by Bagging, a sampling algorithm where a given dataset A is divided into n A' datasets, where x number of instances have been replaced for values of the original A source. This methodology targets both stability and performance of the created model, serving as a quasi-normalization of the underlying dataset.

Gradient Boosted Regression Tree

Similarly to AdaBoost, GBRT builds a regression tree based on the errors of previous iterations of itself, growing more precise by minimizing a loss function, output of which being called pseudo residual. To be more precise, GBRT first creates a single leaf node, with the average value of the datasets regression target. Then, it builds tree terminating on leaf nodes with the new residual, calculated via the loss function. This process repeats as many time as the defined limit of trees.

Var A	Var B	Var C	Value
True	False	False	10
False	True	False	4
False	False	True	1

Table 2.3: Example of dataset for GBRT

On broad strokes, lets imagine the dataset at the table 2.3, where our GBRT must predict "Value". The first step will be calculating $argmin \sum_{i=1}^{n} L(Y_i, \gamma)$ where L is the loss function. Assuming we are using the most common one, $L = 1/2(O-P)^2$, where O means "Observed Value" and P means Predicted Value. The first leaf will always result on the average of all O. This happens as P begins at 0, and the $argmin$ of any set of simple numbers is the average.

This means the first tree believes all values are $P = 5$. We then generate an actual tree, terminating on a new set of pseudo residuals, calculated once again from the same loss function, this time with the updated value of P. From this set of values, we restart the process, using the equation $argmin \sum_{x_i \epsilon R_{ij}}^{n} L(Y_i, F_{m-1}(x_i)) + \gamma)$. This is similar with the first instance, just limiting the summation of $L(Y_i, \gamma)$ values to the current branch of the tree. In our case, our example, at 2.2 will minimized in few iterations, as per its simplicity.

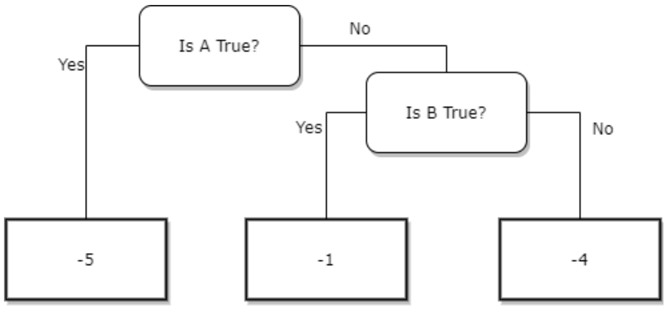


Figure 2.2: First tree generated via GBRT

On a last caveat, to avoid overfitting, a value between 0 and 1, called learning rate, is applied in every iteration of this equation, meaning the loss function will minimize slower, reducing the impact every iteration may have on the final tree, improving the overall model's performance.

Genetic Algorithms

The genetic algorithm (Lambora, Gupta, and Chopra 2019) is a research heuristic that falls under the heading of evolutionary algorithms. It is based on methods from evolutionary biology, such as inheritance, mutation, selection, and crossover. That concepts are inspired in Charles Darwin's theory of natural evolution. Based in evolutionary biology, the genetic algorithm has some hyper-parameters. The population size is one of the main one hyper-parameters and represents the number of entities to be evolved. The generations are another main parameter as each represents an iteration the genetic process. In each iteration of the genetic algorithm, a new population will be generated from the current one. The hyper-parameter, crossover chance, represents the chance of a gene being crossed between two elements of the population. On the other hand, mutation chance is the probability for a gene to spontaneously change into a random value.

Extract, Transform, Load procedures

ETL procedures (Thirumagal et al. 2014) describe an assortment of processes, used to transport information, from one or multiple data sources, to another destination. One key element of this, is versatility of these methods, as they permit a jumble of sources and destinations to transfer information, even when they differ their data structure.

Business Intelligence

Business Intelligence (BI) is a set of abilities, tools, techniques and solutions that help managers in business situation. This method aims helping companies make intelligent decisions, using data and information gathered by various information systems. (Rouhani, Asgari, and Mirhosseini 2012)

In technology perspective, BI allows massive diverse data collected from various sources to be transformed into useful information, increasing increasing effectiveness and efficiency of production. (Carlisle 2018)

Automatic Machine Learning

Automatic Machine Learning, shortened to AutoML, is as Microsoft describes it "...the process of automating the time-consuming, iterative tasks of machine learning model development. It allows data scientists, analysts, and developers to build ML models with high scale, efficiency, and productivity all while sustaining model quality." (Microsoft 2021)

Hyper-Parameters Optimization via Machine Learning

The search to predict the best methods that provide the best results has turned machine learning into one of the popular prediction methods that we can use. To make a higher accuracy of prediction it is necessary to find the best hyper parameter of machine learning methods. In machine learning, a hyper-parameter is a parameter whose value is used to control the learning process. (Wicaksono and Afif 2018)

2.3 Literature Review

When we first started our review process, a factor came to mind, from the requirements and limitations present at 2.1.1 and 2.1.2, respectively: the end user, more precisely, the fact he or she, most likely, won't be able to interact with underlying system, not even through parameters. In any case, having a complex system, with a high level of interactivity, would cause our product to fall in disuse. This information affected our academic research, steering us away of solutions with a focus on the user interaction.

While probing published literature, the problem of surgery duration prediction seemed largely segmented on two methodologies: case-specific predictions and generalized predictions.

Case-specific predictions refer to the application of ML algorithms, when forecasting intervention times, for a small and well defined cluster of surgeries, usually related to each others. Because of this limited scope, a more granular study around the patients' data could be performed, meaning, their models could be better fitted to the available information.

On Devi, Rao, and Sangeetha 2012, the researchers created a system to analyse and predict the intervention times for three surgeries, on a ophthalmology department, using algorithms like Artificial Neural Networks. In this article, they use the predictions to evenly distribute patients, per bed. Yet even though successful, the thigh surgical focus, targeting only three interventions, and using variables specifically related to these(i.e. eye infection) makes the generalization of these solutions, or their implementation on a real world scenario, difficult.

Generalized approaches, on the other hand, find themselves distributed between various medical/surgical specialities at the same time. This means also generalizing the parameters taken into account, in order to avoid overfitting to a single speciality. This approach is more heavy handed, yet easier to implement.

On Huang et al. 2021, the researchers study close to 800 different interventions, sprawling over 25 medical specialities. In comparison, is an assortment of Linear Regressions, Random Forests, and Extreme Gradient Boosting, pitted against each other with the surgeon's prediction as a baseline. On this study, we see the evaluation and segmentation of predictions in three groups: within, meaning their prediction - real time, falls inside their success equation (2.2); underage, going under their prediction; overage, going over their prediction. Another interesting detail, even though not very impactful, was the parameters relating to the primary surgeon's previous production, on the same day and week.

$$f = MIN\{MAX\{ElapsedTime * 15\%, 15\}, 60\} \tag{2.2}$$

Going back to their prediction success equation, while by itself, this is a viable proposal, it contains a major drawback, not differentiating severity, making gradient-based learning, unreliable. Neil Master (Master et al. 2017) documents this phenomenon as the "Prediction Problem" section 3.2, proposing an alternative on how to evaluate the algorithmic performance of f, even if "the loss function (is) non-differentiable but on the regions where it is differentiable, the derivative is zero". Both these generalized approaches take advantage of regression, random forests and gradient boosting, with positive outcomes.

2.4 Technological Review

The reality of this project is strikingly tied to the infrastructure already present on CHUSJ. When we take into account, both the academic literature on the topic, and the technological limitations imposed on 2.1.2, the amount of vectors we must consider, on our technological analysis, diminishes substantially. As such, we'll focus on the tools required for the surgical time prediction and the generation of the optimized scheduling scheme.

2.4.1 Surgical Time Prediction

Weka

Not to be confused by the bird of the same name, the Waikako Environment for Knowledge Analysis (Weka) (Holmes, Donkin, and Witten 1994), is a open source software, developed by the University of Waikako, New Zealand, under the GNU General Public License. This software, more specifically, the version we are studying (Weka 3), began development in 1997, still being updated to this day.

Weka is product developed in Java, providing tools for data preprocessing, data mining, machine learning model generation, instance prediction, as all-in-one solution. Furthermore, Weka can both interact with SQL databases or be taken advantage of as a API, making it incredibly versatile.

The key features, Weka brings to the table are:

- As a pre-built, all-in-one solution, Weka simplifies the model creation process of the project, having a rather linear learning curve;
- Being built as a open source Java program, while still providing an API ready to deploy, ensures our solution is able to run on-site, in pretty much any machine, capable of employing Java Virtual Machines;

Still, relying on a software as Weka, means we are fixed to its ecosystem, and with that, its limitations. Firstly, we'll be dependent on the published packages of ML algorithms, ported to Java; while the list of supported algorithms is sprawling, complex methods, like Gradient Boosted Decision Tree, are not yet implemented. Albeit, there are other Java libraries, filling these holes, yet, if we begin filling a project which these, the strongest point of Weka, the "all-in-one", becomes irrelevant.

Python

Python is a high-level programming language, developed by the Python Software Foundation, first released in 1991 by its designer, Guido van Rossum. In its core philosophies, high readability and versatility, supporting multiple programming paradigms and the usage of external libraries.

Thanks to this ease of use, Python continuous growth on the market is evident(JetBrains 2020), yet, in the context of our work, its diverse ML libraries and processing lightweightness, are the focal point. Libraries like Scikit-learn, TensorFlow, Keras and RPy2, alone cover most common ML approaches.

In summation, Python takes advantage of:

- Being a general-purpose language, making it extremely versatile;
- Ease of use and integration with any system;
- Capable of running lower-level languages, with the Python wrapper, making the usage of older, yet to port algorithms, possible;
- Having a varied bulk of ML libraries, very much capable of sheltering any use case;

Python stands as a compromise between the complexity of a full blown, highly customizable, statistical and ML tool, and the pre-built alternative with limited ecosystem.

R

R is both a programming language and a software environment (Ihaka 2020), first released in 1993, by Ross Ihaka and Robert Gentleman. Its design is first and foremost for statistical computation, having most functionalities related with this field.

Similarly to Python, R permits the integration of packages, adding new features and functionalities, yet, unlike its counterpart, R features a more complex and explicit language, with richer data analysis tools and larger focus on the configurability of its ML algorithms.

R major strengths are:

- Having pre-integrated data exploration tools, making the data exploration phase of the project, easier;
- R can be directly implemented on CHUSJ's Microsoft Reporting Services, where our solution will be interacted with;
- Being built for statistical purposes, its testing and ML algorithms have the highest degree of customization;

R does have the most powerful toolkit of the three studied options, though, its complexity and steep learning curve must be taken into account. Furthermore, being a specialized language might be a hindrance, in future work, if new functionalities are to be executed.

Chapter 3

Value Analysis

A value analysis consists in the application of an array of techniques and methodologies when analysing a possible product or service, in order to maximize its value, while minimizing its costs (Miles 1989). In our project's context, the production of new tools, that implicate the already implanted business processes, may be at risk of being denied implementation, based on budgetary or administrative reasons. Then, to better fight this risk, proposing value becomes a central point of the development process.

3.1 Value

3.1.1 Definition of Value

Firstly, we should define value. Value can be described in the relation between the function and the cost of the product or service. Function consists in any benefit, tangible or not, in the realization of the product/service functionality. On the other hand, cost is the collection of the compromises needed for the product/service to work, being monetary, temporal or others.

We can visualize this relationship with the following equation:

$$Value = \frac{Function}{Cost} = \frac{Tangible\, and\, Intangible\, benefits}{Monetary + Temporal + Intangible\, costs} \tag{3.1}$$

We can observe that, value is highly related to how the client perceives the benefits from the solution in analysis, being this awareness positive when benefits overcome deterrents.This, can be further analysed, from a longitudinal point of view (Woodall 2003), this being, studying value across a temporal line. For this, we divide said temporal line in four categories:

The categories on the table 3.1, describe how value should be discerned, throughout the life-cycle of the product/service:

- **Pre-purchase** - In this phase, value lies on the perception of what the proposed solution can be used for;
- **Purchase** - Better named as implementation, in the optics of software engineering, describes value as the capability to carry out the proposed solution;
- **Post-purchase** - After implementation, the product/service is finally able to be check on how well it can respond to to problem at hand;

- **Post-usage** - On a later stage, besides the inherent value, the product/service retains, quality insurance preserves the value of the solution;

	Benefits	**Sacrifices**
Pre-purchase	Tangible benefits Intangible benefits	Initial budget Design Time Design Work
Purchase (Implementation)	Quality of implementation	Implementation costs Implementation time
Post-purchase	Solution's efficacy Solution's efficiency	Post-purchase support
Post-usage	Quality insurance	

Table 3.1: Longitudinal analysis of value

3.1.2 Perceived Value

Value, as described at 3.1.1, is built upon subjective blocks(Sánchez-Fernández, Iniesta-Bonillo, and Holbrook 2009). These are the functional value, relating with executing the intended task; the esteemed value of subjective factors, alike looks; and how much the product itself is worth, in the eyes of a any potential client. Even functional value, while seeming a concrete metric at first, does muddle after crossing the barrier of 'being able to complete the task'. The customer's perception on how these blocks are assessed, builds the significance of the product, to themselves. Key word, themselves; when analysing value, more actors must be taken into account, for example, the business responsible for the product, will have different characteristics in mind when doing their evaluation (Fernández and Bonillo 2007).

3.1.3 Proposal of Value

The proposal of value is one of the most useful "tools" when approaching a client. Alexander Osterwalder (Y. Osterwalder A. P. 2011) defines the proposal of value as "... the motives for why, clients choose a company over another." and "a Proposal of Value is an aggregation or assemblage of the benefits, one company offers to its clients"

Its imperative that, for a product/service to be adopted, not only responds to the needs of a potential client, but that the client itself perceives that fact. Nevertheless, the proposal of value must inform the interested parties about the value at hand.

We can approach this presentation by virtue of the Value Proposition Model (A. Osterwalder et al. 2014). This paradigm, drills down on the value proposition and the customer segments, of the CANVAS model, described at 3.1.4, further explaining how these elements interact. Observing the figure 3.1, both blocks are built from three items: first, we have the customer's jobs, responded by our business' products and services; secondly, we have our customer's pains, referring to undesired or troublesome tasks that our pain relievers address, by mitigation; lastly, our customer's gains are the target our solution pertains to, explicating the benefits and outcomes we introduce. When all the parts of the customer segment are answered by the premises, on the proposition of value block, a problem-solution fit is reached, meaning, our solution has the potential in addressing the client's problem.

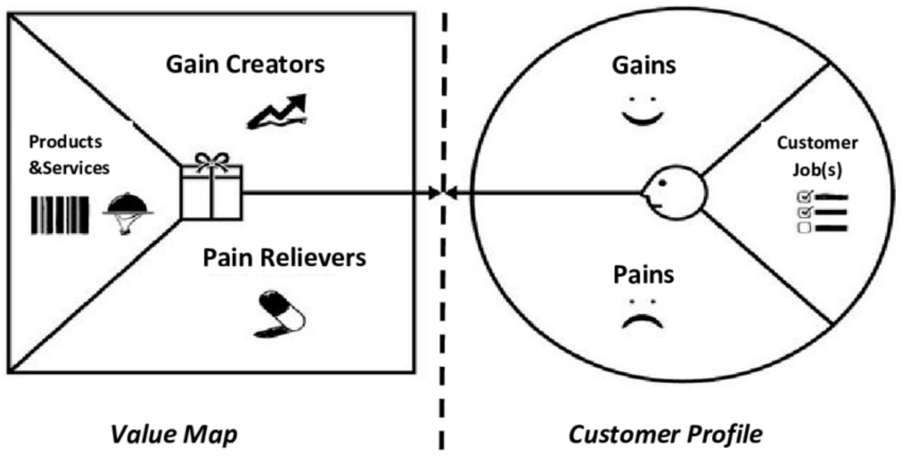


Figure 3.1: Value Proposition Model (source: A. Osterwalder et al. 2014)

Following this model, the service that culminates from this dissertation, targeted specifically at CHUSJ, is an on-site system capable of analysing the LIC and the patients' clinical history, and from these, predict the time needed for an intervention to be performed based on factors, as the surgeon to perform the treatment or the location of intervention. These predictions are later used to generate an operating theater scheduling scheme, taking advantage of process mining algorithms, in order to maximize its occupancy. With these capabilities, will be undertaking both the inerrant uncertainty of the scheduling act, while empowering decision-makers of the institution, with a tool capable of proposing, high surgical production mappings.

While this kind of project, is not either new or novel, hospitals may have extremely different information structures, as such, the specificity of the data available in this kinds of systems, makes it that, there is no 'one size fits all' solution for the problem. Adding the complexity of the medical area, a tailor-made solution, produced in-house, for the ecosystem present at CHUSJ, may greatly impact the surgical production of the institution, culminating in the growth of the number of treatments, the operating block may perform, having a positive impact, on the patients that attend these.

3.1.4 Business Model Canvas

As in any business, there's a complicated chain of key players and components that integrate this project. To better help us visualize the overview of our system we take to the usage of the Canvas model.

The Canvas model is the rationalization and mapping, of what Alexander Osterwalder first proposed in 2005, as the "9 building blocks to help us describe a business model" (A. Osterwalder 2005), being revised in 2008 as shown on figure 3.2. In summation, these blocks serve as an accessible synopsis on how a company does business.

- **Value proposition** - The value our project brings forth;
- **Customer segments** - Whom we mark ourselves to;
- **Distribution channels** - How we reach our target demographic;
- **Customer relationships** - The relationship we keep we our customers;

- **Key activities** - The activities intrinsic to our value;
- **Key resources** - What resources we require;
- **Key partners** - Pivotal associates for our business;
- **Cost structure** - The costs in building and keeping our project;
- **Revenue streams** - How do we profit;

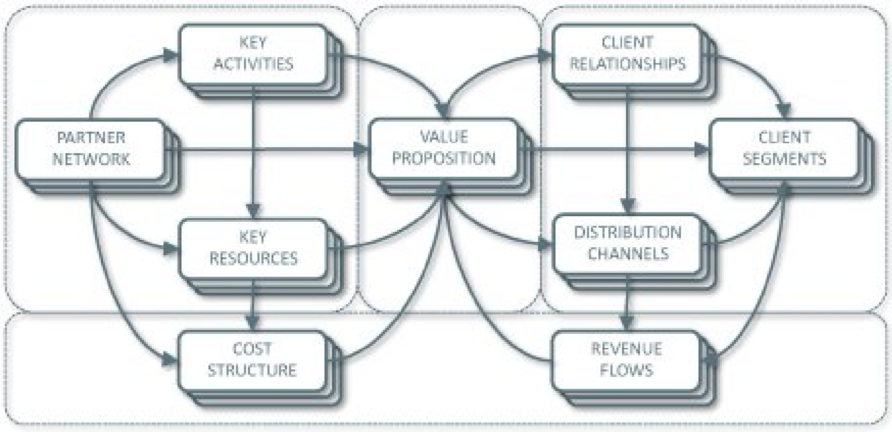


Figure 3.2: Business Model Canvas, 2008 revision (source: A. Osterwalder 2008)

You can see particular segments flowing through each other, making logical connections between the parts that describe a business. This is the basis of a Canvas model, a cluster of interconnected elements that culminate in a business overview. Applying the same steps, our project can be observed as shown on Figure 3.3.

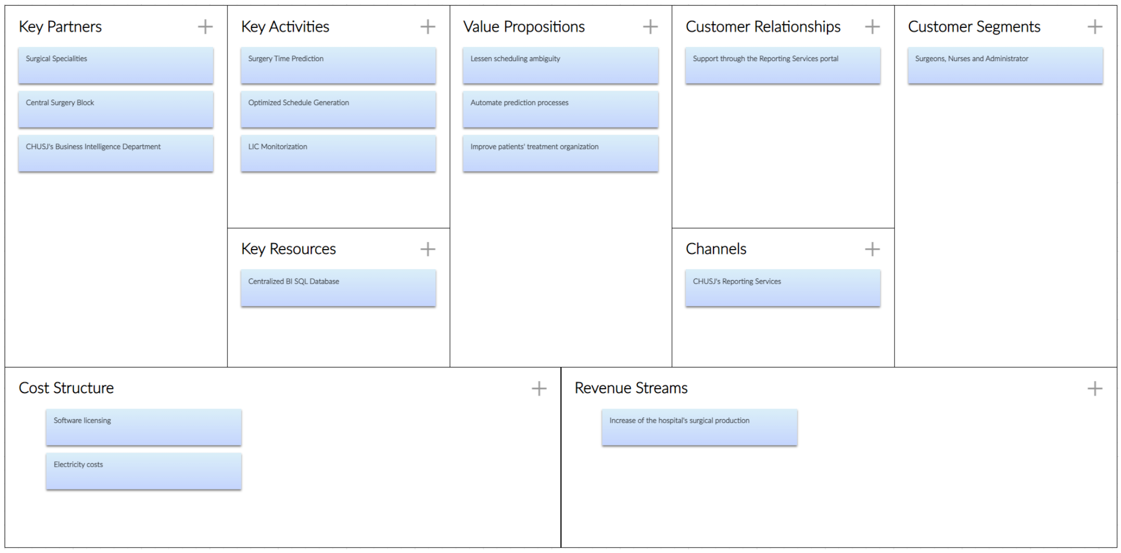

Key Partners
Surgical Specialities
Central Surgery Block
CHUSJ's Business Intelligence Department
Key Activities
Surgery Time Prediction
Optimized Schedule Generation
LIC Monitorization
Key Resources
Centralized BI SQL Database
Value Propositions
Lessen scheduling ambiguity
Automate prediction processes
Improve patients' treatment organization
Customer Relationships
Support through the Reporting Services portal
Channels
CHUSJ's Reporting Services
Customer Segments
Surgeons, Nurses and Administrator
Cost Structure
Software licensing
Electricity costs
Revenue Streams
Increase of the hospital's surgical production

Figure 3.3: Our Business Canvas model

3.1.5 Analytic Hierarchy Process

Analytic Hierarchy Process (AHP) is a decision making tool, using the relative weight(value) between the defining criteria of a product, resulting in a priority scale on the best alternative (Thomas L Saaty 2008). This process is built from(Thomas L. Saaty 1988):

1. Defining the problem on study;
2. Define the criteria, that affect the problem;
3. Compare criteria pairwise, resulting on the relative priority weights;
4. Arrange problem solutions (alternatives) and pass them through the priority scale calculated prior;

Following these steps will result in a vector, where each alternative is mapped a priority value, based on the problem's criteria.

First, our problem consists in choosing the best methodology and tools, to fulfill our project requirements. From this, we derive the following criteria:

- **CR1** - Prediction accuracy: How accurate the prediction is to observed reality?;
- **CR2** - Prediction completeness: How many interventions can the system predict?;
- **CR3** - Ease of use: How easy it is to interact with the system?;
- **CR4** - Ease of implementation/future-proof: How easy it is to implement or alter the solution?

Defined our criteria, we can map these on a pairwise comparison matrix, valuing relative weight from 1/9 to 9.

	CR1	**CR2**	**CR3**	**CR4**
CR1	1	1	7	1/2
CR2	1	1	5	2
CR3	1/7	1/5	1	1/4
CR4	2	1/2	4	1

Table 3.2: Pairwise comparison matrix

With this, we can further normalize every value,

$$M = \begin{bmatrix} 1 & 1 & 7 & 1/2 \\ 1 & 1 & 5 & 2 \\ 1/7 & 1/5 & 1 & 1/4 \\ 2 & 1/2 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 0.24 & 0.37 & 0.41 & 0.13 \\ 0.24 & 0.37 & 0.29 & 0.53 \\ 0.03 & 0.07 & 0.05 & 0.06 \\ 0.48 & 0.18 & 0.23 & 0.26 \end{bmatrix} \tag{3.2}$$

, and calculate the priority vector, by averaging each row of M:

$$v = AVG(M) = \begin{bmatrix} 0.289 \\ 0.359 \\ 0.058 \\ 0.292 \end{bmatrix} \tag{3.3}$$

Known this, we know must ensure the consistency of our values. For that matter, we calculate the consistency ratio (CR), as follows:

$$CR = \frac{CI}{index} \tag{3.4}$$

In order, to complete this equation, we must first calculate λmax:

$$\lambda max = AVG\left\{\frac{M}{v}\right\} = AVG\left\{\frac{1.2048}{0.2892}, \frac{1.5265}{0.3598}, \frac{0.2449}{0.0585}, \frac{1.2848}{0.2925}\right\} \approx 4.24 \tag{3.5}$$

Then we calculate the consistency index (CI), n being the number of criteria on study:

$$CI = \frac{(\lambda - n)}{n-1} = \frac{(4.22-4)}{3} = 0.0822 \tag{3.6}$$

With this, we return CR as:

$$CR = \frac{CI}{0.9} = \frac{0.0822}{0.9} = 9.1\% < 10\% \tag{3.7}$$

CR being lesser then ten percent (0.1), means our weights are consistent, hinting we can trust its outputs. Now we can evaluate our project alternatives, following similar steps. First we discern the options:

- **Weka** - Solution using primarily Weka and Java;
- **Pyth** - Solution using Python and an assortment of external libraries;
- **R** - Solution using R code, and their statistical tools;

These are then segmented on their own, and inserted in comparison matrices with every criterion resulting in:

CR1	Weka	Pyth	R	Priority Vector
Weka	1	3	1/3	0.2864
Pyth	1/3	1	1/3	0.1399
R	3	3	1	0.5736

Table 3.3: Comparison matrix with CR1

CR2	Weka	Pyth	R	Priority Vector
Weka	1	1/2	1/5	0.1285
Pyth	2	1	1/2	0.2766
R	5	2	1	0.5949

Table 3.4: Comparison matrix with CR2

CR3	Weka	Pyth	R	Priority Vector
Weka	1	2	8	0.5934
Pyth	1/5	1	6	0.3412
R	1/8	1/6	1	0.0654

Table 3.5: Comparison matrix with CR3

CR4	Weka	Pyth	R	Priority Vector
Weka	1	1/3	9	0.3159
Pyth	3	1	9	0.6319
R	1/9	1/9	1	0.0522

Table 3.6: Comparison matrix with CR4

Finally using the priority vectors of all alternatives, and multiplying with the criterion's priority vector, we get the best fit, for our problem. These measures, shown on table 3.7 indicate the suitability of a specific programming approach to our problem. In this case, the best methodology is the use of R code, closely followed by Python, then Weka/Java.

	Weka	Python	R
Priority	0.256194	0.344773	0.399033

Table 3.7: Priority vector of alternatives

Still, after discussion in CHUSJ, we decided that Python would be the only viable option, for question of future-proofing.

3.2 Product Innovation Process

As in any market, innovating constitutes a key factor to success. This innovation process must cater both on a strategic and operational level. Drilling down on the process, three stages make themselves evident (Koen et al. 2001), as represented on figure 3.4.

The Front End, often referred to as Fuzzy Front End(FFE), is where the work begins, drafting long-term decisions and analysis and studying likely business opportunities or potential problems to tackle. To be noted, this phase doesn't yet dive on the operational side of the product, but a more visionary take on development. The production itself, comes in the aptly named New Product Development(NPD) phase, where proper technical planning and project building takes precedence. Lastly, the commercialization endeavours, close the cycle transmitting value outwards.

As described by Peter Koen, we can further slice the first phase of our development process, into three symbiotic elements: the "engine" driving the process, the influencing factors and the five activities that characterize the action on the FFE. Denote, on how, both Opportunity Analysis and the Idea Genesis, comprise the entry point of the NCD model, while Concept and Technology Development serves as a gateway into NPD. On a final point, we can observe the connections between the five activities are not polarized, meaning, there is no strict

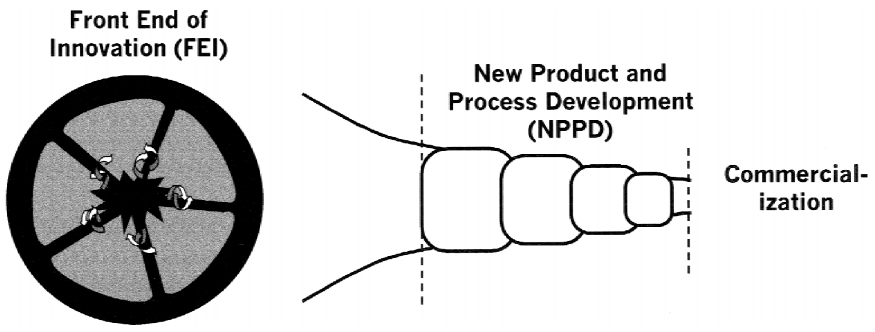

Front End of Innovation (FEI)
New Product and Process Development (NPPD)
Commercial-ization

Figure 3.4: The Product Innovation Process (source: Koen et al. 2001)

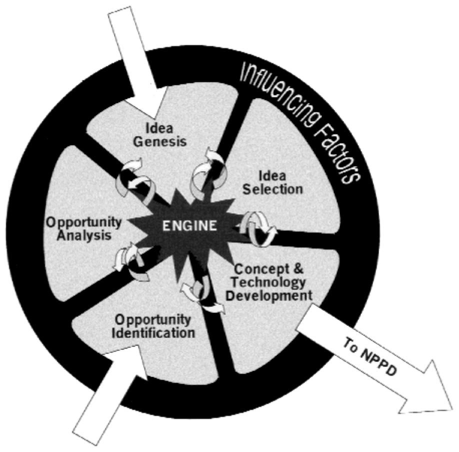


Figure 3.5: The New Concept Development model (source: Koen et al. 2001)

sequential workflow, but a back and forth, amongst these acts. These elements describe the New Concept Development model, as shown on figure 3.5.

Going over each element of the value, we can describe these as:

- **Engine** - A mixture of business logic, internal culture and administrative influence, pushing forward development (Koen et al. 2001);
- **Five Front-End Elements(FEE)**:
  - **Opportunity Identification** - Study phase where technological opportunities are analysed, in the context of the problem at hand. The objective of this endeavour, is to find a viable solution, for a problem or threat, the company is suffering. This study has a structured basis, yet, as the survey subject is highly dependent on the procuring institution, there is fluctuation on the methodologies used(Koen et al. 2001);
  - **Opportunity Analysis** - With the identified opportunities, research on the viability, complexity and attractiveness of each alternative, concludes on how much a company is ready to sacrifice, on execution and implementation(Koen et al. 2001);
  - **Idea Genesis** - Sometimes referred as Idea Generation and Enrichment, is the process refine and endow the studied opportunities, into an idea. This refinement

is conditioned by internal or external factors, like technological infrastructure or key partners, respectively(Koen et al. 2001);

  - **Idea Selection** - Businesses usually have a portfolio of products or services, in order to address the needs of customers. Between these, selecting the one tool to bring the most value for the involved parties, depends on competition, risks and costs on implementation, the capability to execute the solution and the possible payback(Koen et al. 2001);
  - **Concept and Technology Development** - Connecting FFE and NPD, it serves as the final work point before technical development. This stage focuses on building a business case with the knowledge gathers through all the study phases(Koen et al. 2001);

- **Influencing Factors** - Uncontrollable components, of internal or external nature, that impact both the engine or the FEE(Koen et al. 2001);

3.2.1 Opportunity Identification

When this project was first described, it carried itself on a deceptively abstract complication: there was a need to increase surgical production on the operating block of CHUSJ. This need is the result of constant technical delay on LIC and the ever increasing need for special treatments on non-so-generalized ailments. Yet, as described on 2.1, there are an assortment of different, internal complications that might affect the efficiency of any operating block.

While studying these problems, we came in contact with CHUSJ extensive patient's care information structure, and with that, an angle from which to approach this problem. Instead of focusing on problems inside the block, we stand on ensuring a precise scheduling scheme.

Going over market research, we found the area of predictive surgery scheduling highly skewed onto a rather active academic side, while enterprise products were mostly missing. As discussed on 2.3, one possible factor for this, may be the diverse array of data on a patient, multiplied by every surgical speciality on these institutions. As such, we found this project as an opportunity to support a critical element, on the lives of patients unfortunate to need these treatments, while broaching an eminently rich research area.

3.2.2 Opportunity Analysis

Furthering our study, opportunity analysis concerns with the proposed idea, and if following through with it, may be both inline with the desired outcome and viable/doable.

This survey involves market and technological valuation. Looking in the optics of this project, we can ascertain there is a need for this kind of product, internally in CHUSJ, while the external market keeps a more scholarly approach to the problem. On a technical stand point, as described in 2.4, we went over three viable technologies, we could rely on, for the creation of our models. On our customer's goals, the increase of surgical production, constitutes both a monetary incentive and a relief on the strain, that is saturating the LIC.

In summation, our project constitutes an opportunity, to tackle a rather dynamic study field, while proposing value in line with CHUSJ's desires.

Chapter 4

Solution Design

Is this chapter's objective to showcase both the developed solution, and the alternatives that were considered. With this, we exhibit the underlying Software Engineering processes that went into designing an applicable solution.

4.1 Requirements Engineering

Requirements Engineering covers the analysis of both functional and non-functional requirements our project might have. In broad strokes, functional requirements refer to "what our solution must do" and the non-functional refer to "how our solution will ensure its quality and reliability".

4.1.1 Functional Requirements

Functional Requirements define the fundamental capabilities of any software solution. In other words, these present what the end-user may perform with our system.

To decide on our functional requirements, we realized three interviews with different elements of the Central Surgical Block, to better understand, what we must be able to do, and most importantly, what the probable users expect us to do. The results of these inquires is present on appendix D.

As described in 2.1.1, there is only one type of end-user interacting with the system. Further more, the list of capabilities must be highly "hands-off" to ensure the ease of access to the information the user expects. Attending tho these limitations, we expanded the requisites to be as such:

- **UC01** - Observe typical information about entries present on LIC. This is, the user must be able to browser the LIC's scheduling, surgical information and block activity;
- **UC02** - 'Predict' the time needed for performing a surgery based on patients' data, clinical forms and exams, surgeon in charge of the intervention, and others. This predict refers to how, from the users standpoint, they are the ones building the prediction, even though, these were pre-calculated asynchronously at the back-end;
- **UC03** - Generate an optimized scheduling scheme, for the next *x* days (7 days maximum), taking advantage of the predicted intervention time, Master Schedule availability and scheduling algorithms;
- **UC04** - From the administrative console, force the creation of new regression models;

- **UC05** - From the administrative console, begin a new regression and scheduling job, based on the most recent information available;
- **UC06** - From the administrative console, change the running configurations of the software;

4.1.2 Non-functional Requirements

Besides a study on the functional aspects of the solution, Requirements Engineering broads over more non-user focused themes. These non-functional requirements, are part of the FURPS+ model (Prieto González, Tamm, and Stantchev 2016).

Usability

Usability requirements captures mostly the user experience when interacting with our tool. To be more precise, usability refers to how easily and intuitively a user can interact with their interface. For this, recalling the analysis on 2.1.1, our solution must be:

- A simple, yet rich interface on the topic at hand, with the POLA principle in mind;
- Fast and responsive;
- Must facilitate the users work, respecting their, usually limited, free time;

Reliability

Probably one of the major branches on this analysis, reliability measures precision, capability to recover, mean time between failures and availability:

- Our tool must be able to endure any failure on the front-end, back-end and server-side, without compromising the entire structure;
- Our solution must always present trustworthy and precise information, to its users;
- The system must be available 24 hours, 7 days per week;
- Support tasks must not interfere with the deployed system;

Performance

As described on 2.1.2 the solution's back-end will run asynchronously from other components on our system. As such, we are afforded some leeway on our prediction and scheduling tasks:

- The prediction algorithm must be able to calculate surgical times in less than 30 minutes, with a max of 4 minutes for any medical speciality;
- The scheduling algorithm must be able to optimize the pre-calculated surgical times in less than 30 minutes;
- The system must be able to rebuild its predictive model in less than 3 hours;
- Populating the Power BI interface must take less than 10 minutes, during hours of low usage;
- The interface itself must be able to present any combination of pre-calculated data, in less than 5 seconds;

Supportability

As for support, we had considerations with both the support of the current system and the possibility to expand our solution. Minding that, adaptability, maintenance, compatibility and expandability run major:

- Our solution must be able to integrate new functionalities and modules, without any major refactoring of the code base;
- Maintenance and iteration tasks must be non-disruptive for the end-user;
- This system must be compatible with most information devices, without the need of installation or configuration by the user;
- Our system must integrate with Microsoft 365 Ecosystem, without being reliant on it. This is, we must be able to port our solution for other applications without major refactoring of the code base;

Constraints

As contemplated by FURPS+, our solution does run into restrictions from a technical and business standpoint. These limit how we can approach the problem at hand, forcing us to circumvent the problems through different design strategies:

- Our system must be integrate on top of pre-existing databases and services;
- The system's front-end must be built on a Microsoft's Power BI dashboard, with Python and R support disabled, and without writeback capabilities (the end-user cannot directly interact with the back-end);
- Only users with access to the machine running the prediction software, may interact with it;

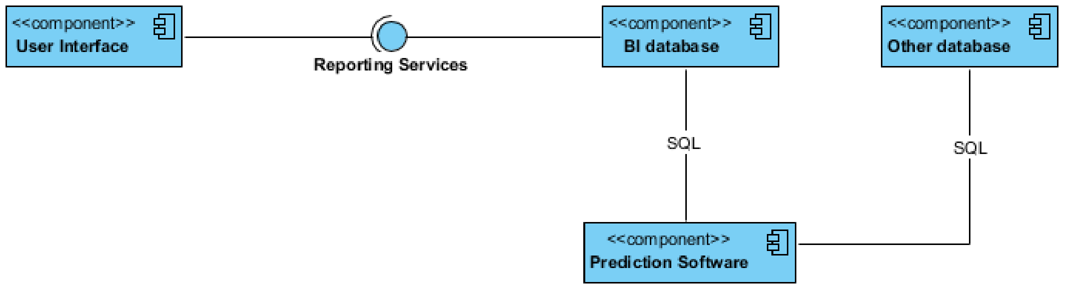
4.2 Design Analysis

In this section will present the design for a viable solution. Firstly we go over the components that will constitute our system. Reminding the limitations described on 2.1.2, our system's interface must be hosted on CHUSJ's Business Intelligence Business Intelligence (BI) platform, where interactions are disabled. This means we must design our solution around this limitation.

4.2.1 Logical view

On the figures 4.1 and 4.2, we can observe how the entire system will communicate between components. These two designs represent the viable alternatives, depending on the utilization of Power Apps.

- **User Interface** - Describes a user interface built on Microsoft Power BI, containing both the LIC and our predictions;
- **Microsoft Reporting Services** - CHUSJ's BI services. Constitutes one of the various Microsoft 365 Ecosystem tools;
- **BI database** - A Microsoft SQL database built specifically for the reporting services. Communicates with the CHUSJ's OLAP cube;



<<component>> User Interface
Reporting Services
<<component>> BI database
<<component>> Other database
SQL
SQL
<<component>> Prediction Software

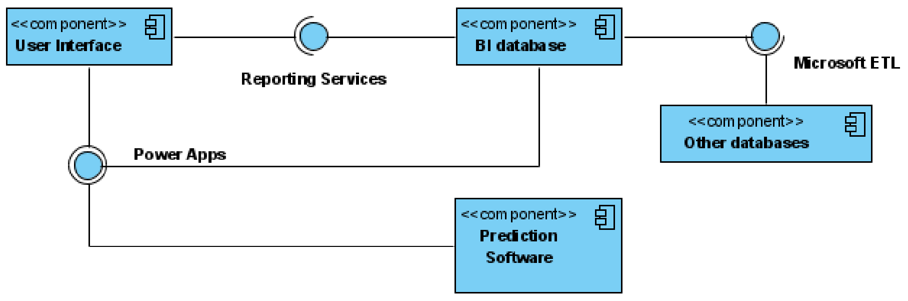
Figure 4.1: Components diagram of our system

- **CHUSJ's other databases** - Cluster of non-centralized databases used by clinical specialities to track patients information;
- **Prediction software** - Our prediction and scheduling tool, built in Python;

First, our tool will interact with the assortment of databases on the back-end, were information needed will be gathered. The predictions and subsequent scheduling schemes are to be stored on the BI database. As you can see we have the user interface, communicating with the Microsoft Reporting Services, which, in turn communicates with the BI database. This implementation contains the limitation of not enabling the user execute commands outside their interface.

During our project, we were in discussion with CHUSJ's Informatics Department, in order to obtain licensing of Microsoft Power Apps. This would permit direct communication between our software and the user interface. This enables interaction with the back-end, as Power Apps enables writeback capabilities. This in conjunction with triggers, ensures interaction between the layers of the system.

In that scenario, our components diagram would be closer to figure 4.2.



<<component>> User Interface
Reporting Services
<<component>> BI database
Microsoft ETL
<<component>> Other databases
Power Apps
<<component>> Prediction Software

Figure 4.2: Alternative components diagram, when using Power Apps

In this version, Power Apps manages the connections between the user interface's data source, and our software, allowing for parameters to flow between them. As Power Apps will be in charge of requesting prediction and scheduling jobs, our ETL process may be done by the BI database itself. The rest of the system remains the same, as LIC information will still require Reporting Services. In the end, we still haven't been granted the usage of Power Apps, and as such, our focus was the first interpretation described.

Diving into the component Prediction Software we observe how this component is subdivided, segmenting its responsibilities(see 4.2.5). The information is pulled into the component by the an automatic scheduling, into Connections. After populating dataframes, this data is consumed by our CorePredict and CoreSchedule. Finally, the output is pushed towards the BI database, where Reporting Services will take advantage, constituting the Presentation Layer.

Taking a step back, this component exists between the Business Layer and the Persistence(Data) Layer of a Client-Server *n* -Tier Architecture (Zuck and DONLON 1997), being responsible for both the access to the data sources, and all the business logic itself. The remainder of work, in this case, the Presentation, is handled by Power BI.

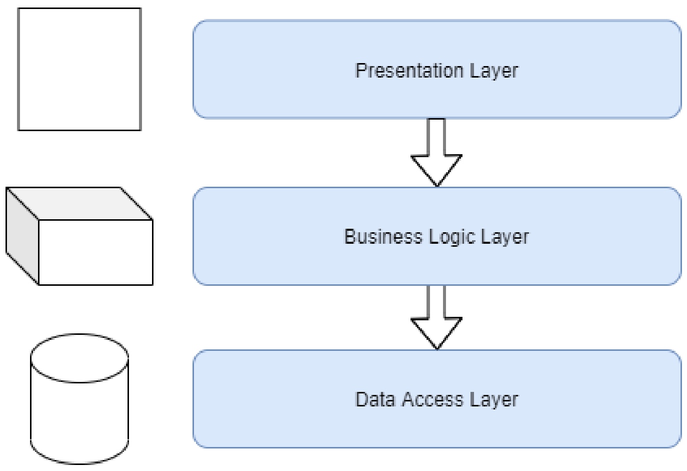


Figure 4.3: Client-Server Three-Tier Architecture

4.2.2 Use Cases

As described earlier on 2.1.1, our system must consider two types of user, the regular user and the administrator:

The regular user is firstly able to check the current state of the LIC, on details like current scheduling and waiting list or the current state of the operating block. These interactions are functionally similar, as such, they have been grouped into a single use case. Furthermore, these represent activities already present in the current structure.

After this, the regular user is able the verify the predictions for surgical times. These will, by default, display the best outcome for any surgical procedure. This information and all its variations, are asynchronously calculated, still, the user may change any number of parameters (i.e. the surgeon) to "build a new prediction".

The regular user may then verify the optimized scheduling for the next 7 days. This scheduling takes into consideration the best case scenario surgical time predictions. This schedule is built on top of the surgical scheduling rules present at the hospital, minimizing overfilling the operating timetable, which would result in rescheduling.

The administrator has access to the prediction software and thus, he or she may actually interact with the software. Still, these interactions are limited and intended for troubleshooting only, as the program must be able to run alone. Then, this user may request the creation of new models based on the implemented algorithms, with their own parameterization; after

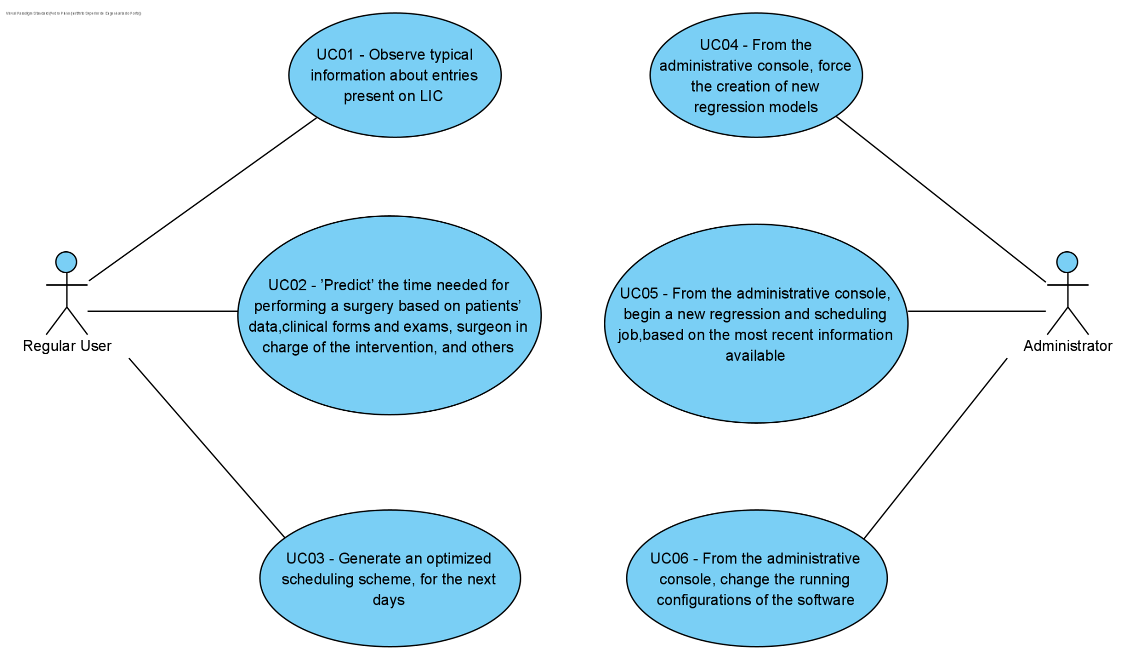

Visual Paradigm Standard(Pedro Pinho(Instituto Superior de Engenharia do Porto))
UC01 - Observe typical information about entries present on LIC
UC04 - From the administrative console, force the creation of new regression models
Regular User
UC02 - 'Predict' the time needed for performing a surgery based on patients' data,clinical forms and exams, surgeon in charge of the intervention, and others
UC05 - From the administrative console, begin a new regression and scheduling job,based on the most recent information available
Administrator
UC03 - Generate an optimized scheduling scheme, for the next days
UC06 - From the administrative console, change the running configurations of the software

Figure 4.4: User stories diagram

this, he or she may be able to request a new set of regressions and scheduling tasks; lastly, this user may change the software running configurations.

4.2.3 Domain Model

In this subsection we analyse the domain model defined to be developed:

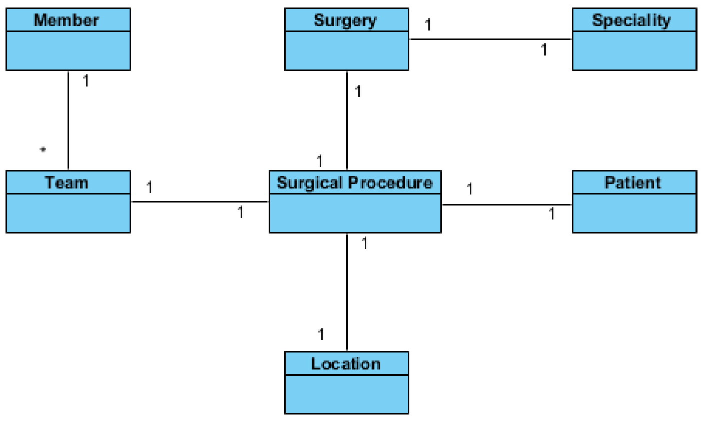


Figure 4.5: Domain Model

Here we can describe the following classes as such:

- **Surgical Procedure** - This is the focal point of the domain model. The surgical procedure consists on a potential/future surgery, performed by the defined team, on a specified location to the patient at hand;

- **Surgery** - The surgery is procedure to be performed, related to a medical speciality. This is characterized by its identification code, present in the table 2.2. The inherent structure of the code contains information relevant for the prediction tasks;
- **Location** - Where, in the surgical block, the surgery is to be performed. Related to the room availability for scheduling;
- **Team** - Surgical team to perform the procedure, consisting in surgeons and nurses present on the act. In the team, the primary/leading surgeon is defined. They are the ones usually performing the act;
- **Member** - User integrated in a team. These are the surgeons and nurses;
- **Speciality** - The medical speciality in charge of the surgeries. Surgeries, locations and team members all belong to a speciality;
- **Patient** - The person undergoing the surgical procedure, with their relevant information;

4.2.4 Entity Relationship Diagram

The following diagrams describe how the relational database was envisioned, to represent our structured data. These models depict the output data of the Prediction Software component that is to be serve to the end user.

The structure of the input databases that have been used won't be displayed, as these exist unrelated to the project at hand and/or contain privileged data from CHUSJ's data structures.

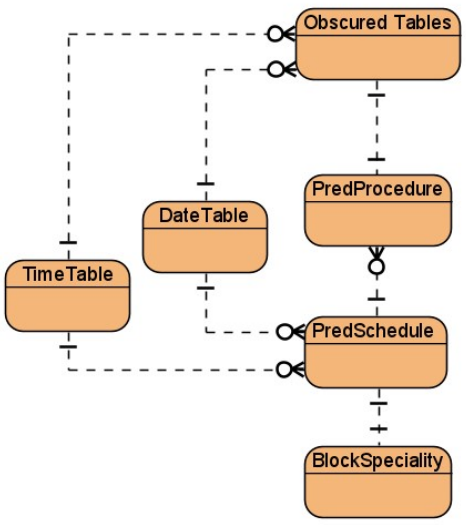


Figure 4.6: Entity Relationship Diagram

The output model diagram 4.6, existing on the BI databases of our structure consists on the following tables:

- **PredProcedure** - This table hosts all the predictions, in regards to the duration of the intervention. In each intervention best-case scenario (prediction with the smallest duration), a bit Boolean flag is defined as true. Furthermore, this table contains keys to communicate with the rest of the data model, that has been obfuscated on the diagram. Lastly, a column keeping track of the date/time when the prediction was processed;
- **PredSchedule** - This table hosts the optimized schedule, for the occupation of the operating theaters. It uses a composite key defined as its own ID and the procedure ID, from PredProcedure. It also contains both a column defining the procedure order on the schedule and date/time for time intelligence;
- **BlockSpeciality** - Table containing the Master Surgery Speciality Schedule (see 2.2), refreshed on weekly basis;
- **DateTable & TimeTable** - Support tables to help structure the information of other tables;
- **Obscured Tables** - Any other tables needed for the correct operation of the dashboard, that were not created for this project alone. In general, these tables are in charge of user's data, past surgeries and surgical team's data;

To conclude these tables are not truncated at anytime, meaning, they serve as historical data to our solution.

4.2.5 Implementation Design

This subsection shows the design of a possible implementation for this project. We will begin with a generalized overview of the program's codebase.

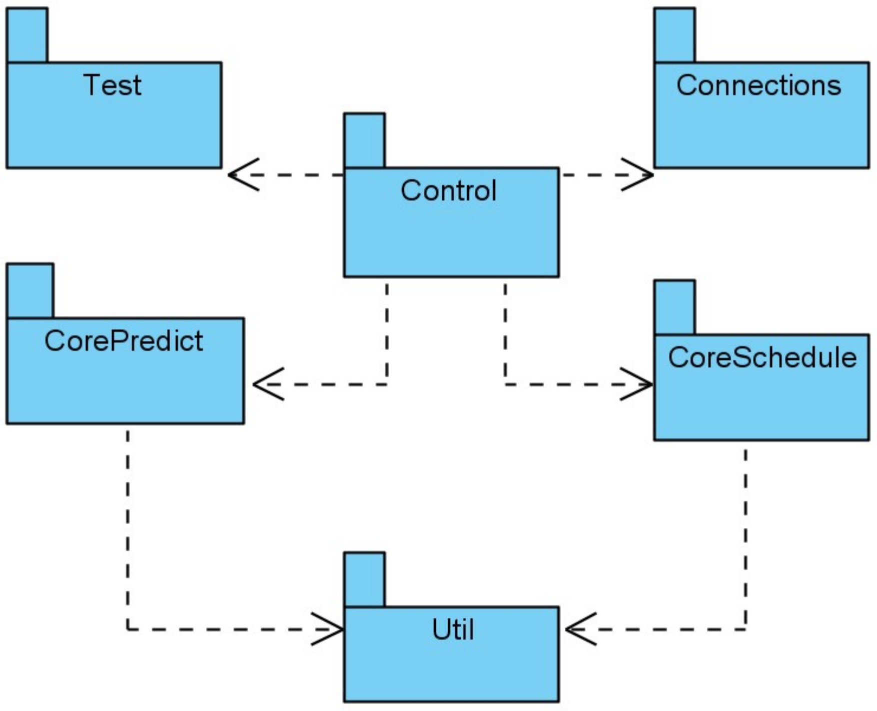


Figure 4.7: Implementation Diagram

On the diagram presented we can observe the packages that constitute our solution, each segmenting the system by task. Firstly, we have the Control package, responsible for for the normal flow of the program. Inside this package we also have the bootstrapper in cases where the application needs to be restarted and error logging and reporting (via log files) and a class responsible for a Command Line Interface (CLI).

On the package Connections we have the necessary code to connect to either a database, in our case Microsoft SQL (MSSQL) or a flat file provided in any time of need. This package is also responsible for the data transformation that enable us to train and run our classification/scheduling models.

Then, on the Utils package, we have an assortment of tools needed for a solution to run properly, such as configurations, enumerators, regularly used functions, between others.

On the package CorePredict, the logic and algorithms responsible for the surgical procedure time prediction. This means both creating, cross-checking and running these models (see 5.1.7). The package CoreSchedule does the same work for the scheduling tasks.

Lastly, the Test packages contains the necessary code for testing the quality and integrity of our solution (i.e., unit tests) and for manual model testing by a command line interface (usually Windows' CMD).

4.2.6 Deployment View

Lastly, we must consider how our solution will be deployed:

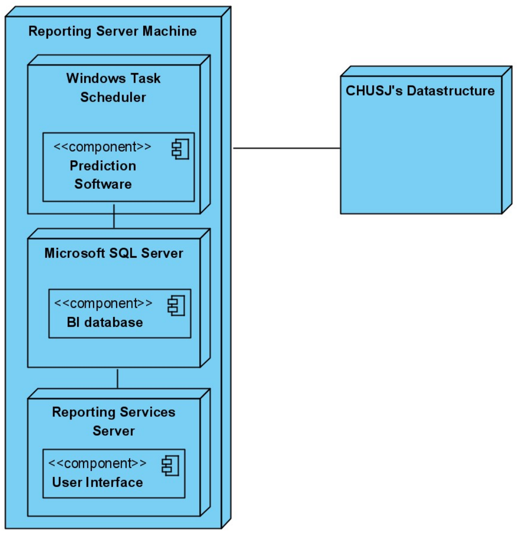

Reporting Server Machine
Windows Task Scheduler
<<component>> Prediction Software
Microsoft SQL Server
<<component>> BI database
Reporting Services Server
<<component>> User Interface
CHUSJ's Datastructure

Figure 4.8: Deployment diagram

In this diagram, machines are evident:

- **Reporting Services Server** - This element refers to the machine responsible for hosting the Reporting Services Server, the BI database server and our prediction and scheduling system, running as a scheduled task.
- **CHUSJ's Datastructure** - This element of the diagram refers to all the databases that act as a data source for our project. They have been purposely obfuscated;

These elements were the ones made available, when discussing the project with CHUSJ

4.3 Machine Learning Considerations

As we intent to implement ML algorithms to calculate our surgical time, we must consider what to evaluate as 'good' or 'bad' regression. To be more specific, how do we evaluate if, for example, a difference in twenty minute between the real and the predict values is acceptable. When classifying, this problem is more straightforward, as an instance either is the correct class or not. Some might say a twenty minute delay on a six hour intervention is superfluous, but the same delay in one hour surgery does carry impact.

On that vain, we based ourselves on the works of Master et al. 2017, using their accuracy metrics, where p is ratio between 0 and 1, and $M > m \geq 0$:

$$\left|Y - \hat{Y}\right| < \min\left\{\max\left\{p\hat{Y}, m\right\}, M\right\} \tag{4.1}$$

as $\tau(\hat{Y})$ where an intervention is classified as on-time, overestimated or underestimated as

$$W = \begin{cases} \text{underestimate}, & \hat{Y} < Y - \tau(\hat{Y}) \\ \text{overestimate}, & \hat{Y} > Y + \tau(\hat{Y}) \\ \text{correct}, & \text{otherwise} \end{cases} \tag{4.2}$$

This formula allows for a fairer cut off when considering if a prediction is correct or not, meaning that, both small and large surgeries have a hard defined limit of time in minutes(in m and M), while procedures that fall somewhere in between have a ratio of their own total time (in p).

Other topic we considered is what we would be modeling. When analyzing the available literature we observed the trend to create models that encapsulate multiple surgeries, some times of multiple specialities. This comes to mean that, for its shear nature, the intervention identifier will have a biased weight during regression, devaluing other information available. Then, considering that, we always know the surgery to be performed ahead of time, we wanted to instead try and create a model per intervention type. This methodology does carry positive and negative effects; for example, our datasets are spread thinner this way and our computational overhead might grow exponentially with the rise in intervention numbers, but, with this our models may better fit each procedure, as it doesn't need to worry itself with unrelated information (i.e. considering surgeons from different surgical specialities).

Lastly, we must examine how our project may work independently of human interaction. It would be unfeasible to force a administrator to daily run our application, and, as such, we designed the following steps, ensuring the continuity of the regression and scheduling tasks even though no user may be interacting with the software.

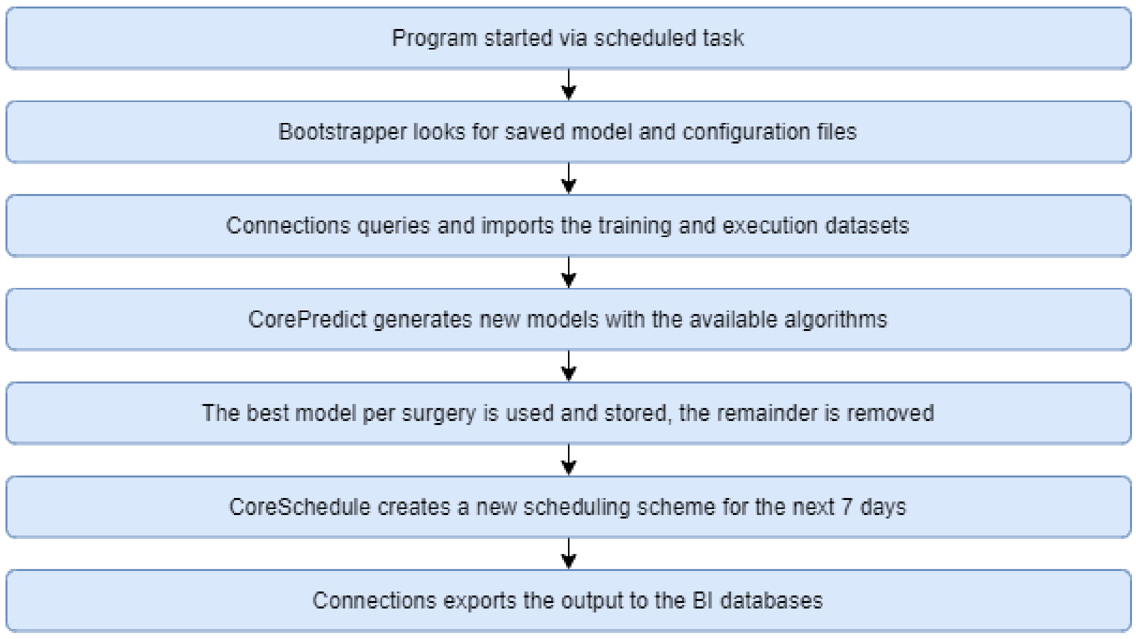

Program started via scheduled task
Bootstrapper looks for saved model and configuration files
Connections queries and imports the training and execution datasets
CorePredict generates new models with the available algorithms
The best model per surgery is used and stored, the remainder is removed
CoreSchedule creates a new scheduling scheme for the next 7 days
Connections exports the output to the BI databases

Figure 4.9: Headless operation steps

Chapter 5

Solution Implementation

On this chapter, it's our objective to describe and document the work implemented at the time of writing. Our solution required intervention in three fronts; firstly, the Python application responsible for all the ML and scheduling tasks; the Microsoft SQL Database hosting the results; finally, the Power BI dashboard, serving as user interface for the data.

In the following sections, these elements will be described in depth, through diagrams, pseudo-code and actual code, with the intuit of permitting to an outside reader to replicate our solution to the best of their abilities. Some information may be privileged by Hospital São João, and as such, obfuscated.

5.1 Prediction Software

"Nurupo", called after an old software engineer joke, is the name of the Python application created. This program was first developed in Raspberry Pi 3 cluster, running FreeBSD 12.0, being then completed and tested on a Windows machine. When coding the scripts responsible for our ML models, we used Anaconda as our coding environment. Functionality in Linux distros has also been tested, Pop!_OS and Manjaro specifically, though, the likelihood of usage is quite low. Lastly, while a CLI has been developed, "Nurupo" can and usually will be ran as an headless application.

5.1.1 Program Structure

This program, as defined in 4.2.5, is divided in six packages.

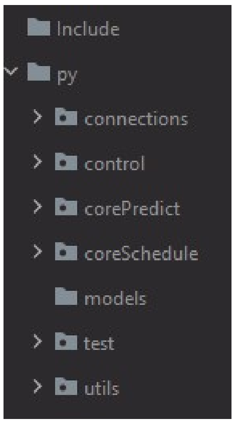

```
Include
py
  connections
  control
  corePredict
  coreSchedule
  models
  test
  utils
```

Figure 5.1: Directory tree of the project

Control Package

Control consists in four '.py' files, these being bootstrapper, mainRunner, cron and admCLI.

bootstrapper.py is the start of our program, being responsible for the initialization of our software in case of restart, starting the admCLI(if enabled), looking up .xml configuration files, loading pre-produced ML models and initializing error logging. Depending on the information gathered, bootstrapper may create several operation related objects, for example, the software's runningConfig object. These are passed along to mainRunner.

The mainRunner.py is responsible for application workflow, containing the main application thread and any required data for the normal operation of our app.

Even though its currently unused, taking the Windows Scheduled Task approach, we maintain a simple script capable of tracking time. The cron script ensures the execution of the regression and scheduling tasks on defined moments.

Finally, admCLI.py controls the administrative CLI, when not running headless. From here, an user may force new regressions or schedules, train new models, change to older models and change the running configurations or connection details.

Connections Package

Being responsible with communicating with the exterior, Connections has three .py documents, dbCommunicator, dbTransformer and mailman.

dbCommunicator.py is responsible with connecting and querying our data sources and destinations, via the Open Database Connectivity (ODBC) API. It can also import or export data from/to flat files, when requested via administrative console.

After importing our data, dbTransformer.py does most of our data transformation/processing into an usable state inside the application, populating variables and objects as needed.

mailman.py is a basic script using the Simple Mail Transfer Protocol (SMTP) to send an automated mail to the administrator defined email in case of any error logging.

CorePredict Package

CorePredict is currently the largest package available, consisting in 7 .py files, coreP, modelMeta, hyperGenetic, trainerP, runnerP, mlr, rfr and gbrt.

First, coreP.py can be seen as a controller for all tasks performed by CorePredict. It also contains the logic of our AutoML implementation and the tools for serializing and saving a ML model.

modelMeta.py through ABCMeta, permits us to create interfaces, to enforce certain object structure in our program. To be more specific, this interface is consumed by mlr, rfr and gbrt, guaranteeing the abstraction and correct behavior required, for normal operation.

hyperGenetic.py hosts a simple evolutionary algorithm, responsible of automatically generating good hyper-parameters, used on our modeling efforts.

mlr.py, rfr.py and gbrt.py all contain the implementation of the regression algorithms, described on 2.2.2. These object may be able to do some extra data transformation/processing, if needed. Lastly, they are bound by modelMeta object structure.

Lastly, trainerP and runnerP, as the names may imply, serve as controllers for both training new models or running already produced models.

CoreSchedule Package

CoreSchedule contains the needed code for the automatic scheduling processing. It consists on coreS and runnerS. These serve similar purposes to their regression counterparts.

Utils Package

Through production, Utils has been simplified, standing now with 4 .py files, math, tools, enumerators and runningConfig.

While the libraries used in our project contain most of any mathematical equations needed, math.py hosts all the remainder that had to be manually written.

tools.py's is responsible for creating the SMTP connection, for mailman.py to sent mails and writting down any arising runtime error, for logging purposes.

enumerators.py is rather self explanatory, containing all enumerators used on our program.

runningConfig.py mantains the current application configurations and writes any changes to their config.xml, fed to the bootstrapper, on reboot.

Test Package

Lastly, our Test package contains both unitTest.py and modelTest.py

unitTest.py refers to the Unit Test common on Software Engineering Good Practises.

modelTest.py consists in a package that, when called, will force the production of new regression models, with predefined seed values, hyper-parameters and datasets, to ensure consistency. These models do not reflect neither the best, nor a realistic outcome, but a predictable one.

5.1.2 Rebooting and bootstrapping

When starting anew, our program requires some preamble, and for that effect, we resorted to bootstrapping. First, the class will create an admCLI object, if CLI was requested by parameter (–cli) on start. This object will eventually be passed along to mainRunner.py.

Then, it will look for any document named config.xml on the root folder of execution context, taking advantage of the minidom library. If found, the datastream is passed along to the runningConfig builder, else, the method is called without parameters to indicate an initialization with predefined values.

After this, the bootstrapper will look for serialized ML models, with the following naming convention of "$./models/\alpha - \beta - \omega.joblib$" where α refers to the algorithm used, β refers to the intervention being evaluated and ω refers to the model creation timestamp. These files are both created and read back through joblib's dump and load methods and passed along to mainRunner.py, to eventually be used by their respective class.

5.1.3 Main Runner

After the program has been correctly initialized, mainRunner.py is fed the runningConfig, any models found during bootstrapping, and, if initialized, the admCLI. From here, the objects for dbCommunicator, dbTransformer and mailman are initialized, taking as parameters either any relevant information from runningConfig, or nothing, meaning a default initialization.

After this, mainRunner will check if a admCLI is present; in case it is, it will call for the creation of a modelTest object and then draw the main menu, via admCLI.menuWait(). Until a valid option is selected the thread will be locked inside a while loop. When a valid option is selected, the program will jump to the chosen functionality.

If admCLI is not present, mainRunner will start the automatic workflow (see 4.3). On the code at 5.1 we can observe second step of this flow, we first call dbCommunicator.establishConnection(), building a session with our source database and our output database(in this case, via MSSQL Trusted Client). This method will return a Boolean outcome, depending in successfully connecting to a data source and data destinations. If admCLI is not present and the connection is unsuccessful, the program will attempt to call mailman.sendEmail(errorLogging.lastLog()) and then exit. If a successful connection is established, we then query the database with dbCommunicator.queryInput(), returning once again a Boolean outcome. After this may be called dbCommunicator.getDF(), returning the dataframes of our query, defined on Pandas' library. This dataframe is the main data structure we'll use when manipulating information. The method dbTransformer.dfPreparation(sourceDF) will then process the information retrieved, into an usable state.

```
if dbCommunicator.establishConnection("sqlTrusted"):
    if !dbCommunicator.queryInput():
        errorLogging.newLog(5)
else:
    errorLogging.newLog(4)

if errorLogging.newErrors():
    if onADM:
        admCLI.pushError(errorLogging.getLastBlunder())
    else:
        errorLogging.errExit()
```

Listing 5.1: MainRunner calling for connection

After having our data prepared, we can start the model creation and subsequent regression/scheduling work.

5.1.4 Data Query and Processing

Taking a step inside dbCommunicator, we have the required method to establish a connection to the data source and destinations. The default behavior is to connect to our MSSQL, authenticated via Microsoft's Trusted Connection. Still, we have the option to connect via database's credentials and flat file. In the case of flat files, the default behavior is to split the datastream by ';', being able to change this via runningConfig. Our connection is then established thanks to the pyodbc.connect(), from pyodbc library, resulting in an connection object dbCon, which is used by pandas.read_sql_query() or pandas.read_csv().

```
if sourceType == "sqlTrusted":
    name = "SQL-" + str(time.time())
    dbCon = pyodbc.connect(
        'Driver={SQL Server};''Server=REDACTED;''Database=DB_Galen;''
    Trusted_Connection=yes;')
elif sourceType == "sql":
    name = "SQL-" + str(time.time())
    username = runningConfig.getUser()
    pwd = runningConfig.getPwd()
    dbCon = pyodbc.connect(
        'Driver={SQL Server};''Server=REDACTED;'
        'Database=DB_Galen;''Trusted_Connection=no;'
        'UID='+ username +';''PWD='+ pwd +';'
    )
# Only possible when going through admCLI
elif sourceType == "csv":
    fileDiscovery = True
    while fileDiscovery:
        try:
            fltFile = runningConfig.getFlatFile()
            sepp = runningConfig.getFlatSep()
            dtFrame = pandas.read_csv(
                str(path.Path(__file__).parent.absolute()) + "/
    localSources/" + fltFile + ".csv", sep= sepp,
                index_col=0)
            fileDiscovery = False
            name = fltFile + "-" + str(time.time())
        except:
            errorLogging.newLog(3)
            admCLI.pushError(errorLogging.getLastBlunder())
```

Listing 5.2: Connecting to a datasource

There are three data importation queries, in this case for trainingData, toPredictData and the masterScheduleData, with the following structures:

Variable name	**Type**	**Note**
NUM_LISTA_ESPERA	int	ID number of entry in LIC
SEXO	String	Gender of the patient, in text value
IDADE_ACTUAL	int	Age, in years, of the patient
ASA	String	Patient's health risk
PRIORIDADE	int	Patient's priority value, ranging between 1-4, being 4 the highest
COD_INTERV_CIRURGICA	String	ICD-10 intervention code
TIPO_CIRURGIA	String	Text identifying the surgery as 'internment' or 'ambulatory'
NUM_CIRUR	int	Number of surgeons to perform the intervention
CIRUR_PRINCIPAL	String	Main surgeon
ANESTESISTA_PRINCIPAL	String	Main anesthetist
CIRUR_INTERNO	bit	Flag if the main surgeon is an intern/student
ANESTESISTA_INTERNO	bit	Flag if the main surgeon is an intern/student
TEMPO_INTERVENCAO	int	(Only on training) Intervention time, in seconds

Table 5.1: Training and Running Datasets

Checking now dbTransformer, after being passed both trainingData and toPredictData, we clean up our bulk dataset, eliminating cases where surgery time equaled zero or no medical staff was assigned to the intervention. These both reflect uncommon, yet harmful cases of database errors.

Variable name	Type	Note
DATA	Date	Date for the first Monday of the week
ESPECIALIDADE_ID	int	ID for the surgical speciality occupying the operating room
LOCAL_ID	String	ID for the operating room
BLOCK_TIME	int	Time, in minutes, of when the room may be occupied (i.e. 08:00 = 480)

Table 5.2: Master Surgery Speciality Schedule dataframe

We then separate this dataframe into multiple ones, using the intervention code as the discriminator(see 4.3). The intervention time(on training) is then passed by the logarithmic transformation described by Neil Master (Master et al. 2017).

Lastly, we perform one-hot encoding on any categorical data, besides the intervention code. As we are using regression models, categorical data will be unusable, in most cases. We do this through a pandas.get_dummies(dtFrame) method.

As the correct data querying is necessary for a proper execution of our program, if at any time these methods fail, false will be returned to mainRunner and errorLogging.newFailure(code) will be called.

5.1.5 Regression Model Production and Testing

Diving into CorePredict, we can observe how regression models are created and compared in between each other.

This class works in stages, firstly, all required information is gathered, meaning the input dataframes, the errorLogging object, the admCLI(if present) and any past model objects that the bootstrapper may have loaded. From here here can be two possible outcomes, either admCLI is available, and as such we are in a non-automatic mode or we are running headless instance, meaning we're following the autonomous workflow.

Once again let's follow the headless functionality, beginning with generating or loading from the bootstrapper, our evolutionary algorithms for hyper-parameter optimization. We then focus on the creation of new models, taking advantage of the interface modelMeta, which guaranties that every object is up to par and capable of communicating with coreP; using this abstraction we can generalize all the method calls, facilitating the integration or removal of algorithms as we see fit. After processing all models, we calculate their accuracy according to the equation 4.1. Finally, the best performing model, for every given surgical intervention, following our AutoML defined rules, is classified as "current" and used for the next regression tasks while others are discarded.

5.1.6 The Abstract Model Object

As of the current implementation, three algorithms for regression are present at any given time. These follow the interface modelMeta containing the following external methods:

- __init__(hyperParameters) - A builder method responsible with instantiating the object, passing along the optimized hyper-parameters for the model creation;

- buildModel(dataFrame) - Starts the modeling act, around the supplied dataset, also creating the testing dataframe, to be used later;
- runModel(dataFrame) - Starts the regression act, around the supplied dataset, returning their value in a dict structure;
- testModel() - With the testing dataframe from earlier, call runModel(dataFrame) and then evaluate the output on its accuracy;
- An assortment of getter methods - To return object related information, such as algorithm implemented, intervention evaluated, creation timestamp, time to generate model and accuracy.

Being similar between each other, using gbrt.py as an example, we can dive deeper into the model objects. First, we can see the libraries used and some hard-coded variables, in this case just the name of the model. We then have the init method, responsible for setting the hyper-parameters, with certain defaults, if needed; after parametrized, we set an k-fold cross validation where $k = 4$; the small value of k was mostly chosen because of the small datasets available.

```
from abc import ABCMeta, abstractmethod

class model(metaclass=ABCMeta):
    @abstractmethod
    def \_\_init\_\_(self):
        pass

    @abstractmethod
    def buildModel(self):
        pass

    @abstractmethod
    def runModel(self):
        pass
    # . . .
```

Listing 5.3: Abstract Class

On buildModel() we start by building the training and testing sets, dividing the original dataframe into four, two sets of training and testing where the result, intervention time has been removed. From here we "fit" our model.

With runModel(dataFrame), we iterate through the supplied dataframe, using predict() on every instance to build our outcome dataset. As reminder, these values are not yet usable, for they still remain altered by the logarithmic transformation. Only after passing an antilog equation can we either read them or use them on scheduling tasks.

Finally, testModel() calls runModel(dataFrame), with the test dataframe created when first running buildModel(dataframe). With the output, Utils.math.successEquation(bulkPredictions) and Utils.math.stdDev(bulkPredictions).

5.1.7 AutoML Implementation

As described in 2.2.2 AutoML is a valuable help on software implementations where model iteration is a given. These tools can automate almost any phase of the ML workflow, from data processing, to choosing the best algorithms and optimization. On our solution we have taken advantage of the sklean-genetic-opt library to optimize our hyper-parameter choices, while the process of evaluating, using and persisting the best model, was done manually.

Hyper-Parameter Optimization using Genetic Algorithm

When working in machine learning, we can't only define what a model learns, but how a model learns. For that end, hyper-parameters ensure we can fit a model to the best of their ability, to the given dataset. Still, most of the times, there's multiples of these values, which may variate hundreds of times. The process to find the best combination of variables, can, and normally does take a lot of time and effort. For this, we once again resort machine learning, adopting easier models to classify the more complex ones.

Based on the sklean-genetic-opt documentation and the works of Wicaksono and Afif 2018 parameterization and methodology, we utilized genetic algorithms to find good fitting hyper-parameters for our regression models, with the caveat that, while, if searched manually, we might had found better results, as most of this program utilization is an unsupervised and headless execution, betaking automation is the best option available.

Using GBRT as an example once again, on the code snippet 5.4 we observe their implementation. Note that, We could use evolved_estimator as a model, as this object is a GBRT instance, but we return its hyper-parameters to be used on the other modeling attempts. The hyper-parameters of the genetic algorithm itself were defined as a population size of 50, 1000 generations, with a crossover probability of 50% and a mutation probability of 10%; the remainder options are left as default and multi-threading is turned on. Lastly, we used k-fold cross validation with 4 splits.

```python
param_grid = {
    'n_estimators': Integer(50, 500),
    'max_depth': Integer(2, 12),
    'min_sample': Integer(2, 12),
    'learning_rate': Continuous(0.01, 0.1),
    'loss': Categorical("ls")
}

cv = StratifiedKFold(n_splits=4, shuffle=True)

evolved_estimator = GASearchCV(
    estimator=gbrt,
    cv=cv,
    scoring='accuracy',
    population_size=50,
    generations=1000,
    tournament_size=3,
    elitism=True,
    crossover_probability=0.5,
    mutation_probability=0.1,
    param_grid=param_grid,
    criteria='max',
    algorithm='eaMuPlusLambda',
    n_jobs=-1, #multithreading
    verbose=False,
    keep_top_k=5
)
return evolved_estimator.best_params_
```

Listing 5.4: Hyper-parameters optimization on GBRT

Model Hot-Swapping

During our headless operations, from time to time our application will retrain itself with ever growing datasets and the system needs to be able to effectively choose between retaining a model generated in the past or the new one, maybe even a model from another algorithm. For this, we have enforced a list of rules that when followed, will keep in check the progression of our algorithms, when working without human interaction. These rules are as follows:

- The software must always try to use models with the best performance metrics;
- New models must not lower their performance more than 1%, than previous iterations;
- New models must be trained with more instances then older ones;
- New models must contain the same or more training columns then older ones (due to one-hot encoding);
- Current models may not be older then one month, being replaced as soon as new model is available, if not breaking any other rules;

This set of rules does open the door for some performance slipping, but, we must find a balance between precision and adaptability, as an hospital is an ever-changing "creature". With some supervision from time to time, we can ensure this balance, while retaining a semblance of human independence. Note that, when interacted via admCLI, these rules are circumvented in their totality.

5.1.8 Scheduling Algorithm

After the regression tasks are done, we can save our output and start the automatic scheduling work. Inside coreS.py we first need to join our regression dataframe to some scheduling relevant information, acquired on 5.1.4. These contain the surgical speciality, the patient's priority value and their 'time to leave' (see 2.2.1).

This scheduleInputDataframe is then passed through scheduling model. This will output x number of lists, where x is the number of specialities being accounted. Lastly, these values will then populate the supplied MSSS, by surgical speciality, ending the scheduling process.

Taking a step into algorithm, we observe the following rules:

1. Split the scheduleInputDataframe, based on surgical speciality. There is no relation in between splits;
2. Order each list by the predicted surgery time column;
3. Split these lists by their 'time to leave', in groups of '0-7 days', '8-24 days' and '+25 days'; These new splits are orders between themselves, where '0-7 days' is first and '+25 days' is last;
4. Split once again by patients priority; Higher priority lists come first;
5. Join all lists (from the same surgical speciality) into a single dataframe and populate the MSSS accordingly;

These can be reduced to the following pruned tree, starting after the surgical speciality split:

With this, if we supplied the 5.3 to our scheduling algorithm, we would accomplish the following results:

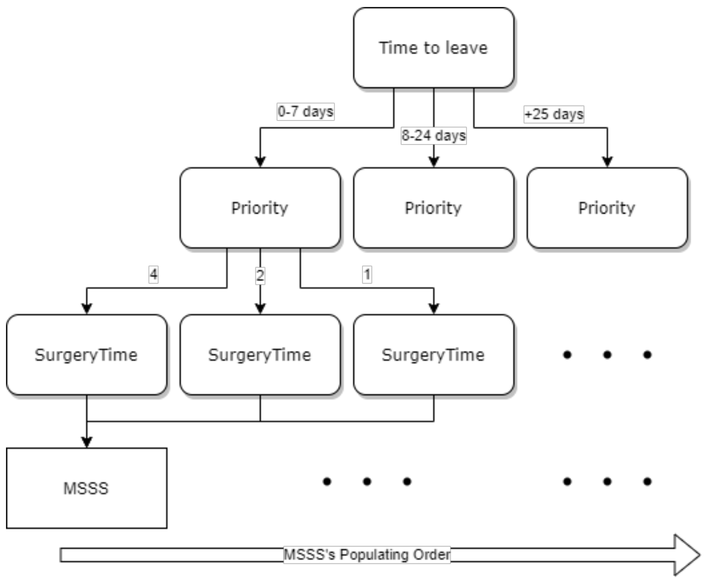


Figure 5.2: Scheduling algorithm visualization as a pruned tree

ID	Surgery	SurgeryTime	Speciality	Priority	TTL
1	XXXXXXX	140	33	2	35
2	YYYYYYY	300	33	2	15
3	XYXYXYX	45	33	1	2
4	ZZZZZZZ	30	33	2	40
5	ZXZXZXZ	80	33	1	15
6	XXXXXXX	100	33	4	6

Table 5.3: Example dataframe, pre-scheduling

As observable, our output has been ordered following this 'Time to leave'/Priority/Duration model. From where, we take one last step of adding 15 minutes to every surgery duration (as there is a mandatory waiting time between surgeries) and we may populate the MSSS. Naturally, as we intend to avoid surgery rescheduling, while populating the MSSS, if an intervention would surpass the time slots available, the next surgery able to fit the slot is to be scheduled first.

ID	Surgery	SurgeryTime	Speciality	Priority	TTL
6	XXXXXXX	100	33	4	6
3	XYXYXYX	45	33	1	2
2	YYYYYYY	300	33	2	15
5	ZXZXZXZ	80	33	1	15
1	XXXXXXX	140	33	2	35
4	ZZZZZZZ	30	33	2	40

Table 5.4: Example dataframe, post-scheduling

5.1.9 Other Functionalities

Command Line Interface

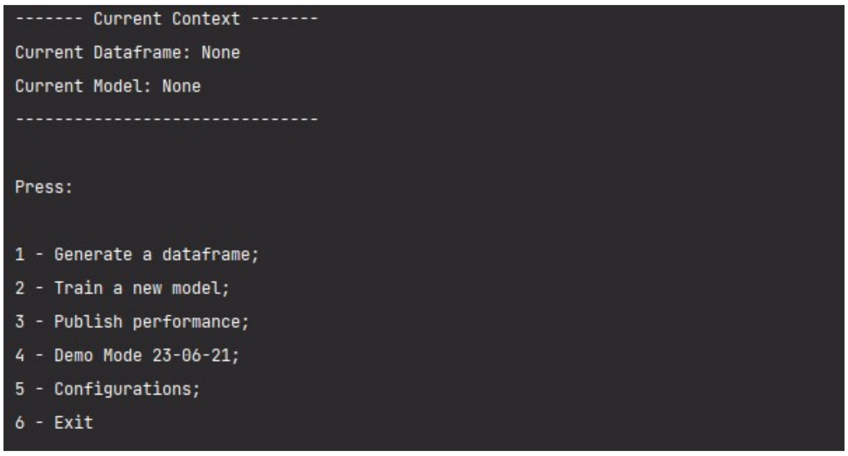
On cases were the system needs to be interacted with, the CLI, controlled by admCLI is our user interface. This classes contains a menu tree interacted with through scanners. Each element may dive deeper into the tree's branch or call for the execution of specific methods, the output of which, being draw if necessary.

On the snippet 5.5 we have the straightforward implementation of our CLI, refering to the first level of our menus. It's based on printed text and input scanners, looped while the Boolean running is true. On choosing one element of the menu, we enter a nested loop, in this case innerRunning, where a new set of option might be displayed. Note that, not all options might have nested menus, for example, option 3 calls a single method.

```python
def menuWait(self):
    running = True
    while running:

        clearConsole()
        printCurrentContext()
        menuOption = int(
            input(
                "Press: \n\n1 - Generate a dataframe;"
                "\n2 - Train a new model; \n3 - Publish performance; "
                "\n4 - Demo Mode 23-06-21; \n5 - Configurations;"
                "\n6 - Exit\n"
            )
        )
        if menuOption == 1:
            innerRunning = True
            while innerRunning:
                # ...
        elif menuOption == 2:
            innerRunning = True
            while innerRunning:
                # ...
```

Listing 5.5: CLI Menu Logic



```
------- Current Context -------
Current Dataframe: None
Current Model: None
-------------------------------


Press:

1 - Generate a dataframe;
2 - Train a new model;
3 - Publish performance;
4 - Demo Mode 23-06-21;
5 - Configurations;
6 - Exit
```

Figure 5.3: CLI's look, first menu after bootstrapping

Error Logging and Email Alerts

When something goes outside the planned execution flow, mostly on headless execution, it's classified as an error. If this is to happen, the errorLogging instance first writes(or appends) to a .txt file the error code and information; then, mailman.sendEmail(lastBlunder) is called to notify the administrator and lastly, the exit() method is called to terminate the application. If running through the admCLI, the application will not be terminated, instead choosing to print the error on the screen.

In 5.6 we can see how the error information is based on our Utils.enumerators, referred by code. Then the log document, in this case hard-coded to errorLogging.txt is opened and written to. lastBlunder is then defined to be sent via email. On that note, error related to the mailman class won't try to send emails, as the necessary instance or preparation might not be completed.

```
def __init(self):
    self.newErrors = False
    self.lastBlunder = ""
    self.mailable

def errorLogging.newLog(code)
    errorString = ""
    if code == 0:
        # . . .
    elif code == 1:
        # . . .
    self.writeDownError():
    self.newErrors = True
    self.lastBlunder = errorString
    self.mailable = not code == 7 #erro no setup do email

def errorLogging.newErrors():
    return self.newErrors

def getLastBlunder():
    self.newErrors = False
    return self.lastBlunder

def writeDownError():
    log = open("errorLogging.txt","w")
    log.write(self.lastBlunder + "\n")
    log.close()

def errExit()
    if mailable:
        self.mailman.sendMail(self.lastBlunder)
    exit(0)
```

Listing 5.6: Error Logging

In mailman.py we see a straightforward implementation of a SMTP through an Secure Sockets Layer (SSL) connection, via the libraries smtplib and ssl. As of now, we haven't incorporated with the hospital's internal mailing system, and as such, we resorted to using an Gmail account, created for this purpose alone. This serves more as a proof of context.

```
def __init(self, senderMail, pwd, targetMail, port = 465):
    self.senderMail = senderMail
    self.pwd = pwd
    self.targetMail = targetMail
    self.port = port

def sendMail(self, message):
    context = ssl.create_default_context()
    with smtplib.SMTP_SSL("smtp.gmail.com", self.port, context=context)
   as server:
        server.login(self.senderMail, self.pwd)
        server.sendmail(self.senderMail, self.targetMail, self.message)
```

Listing 5.7: Basic Mailing Implementation

Configurations

Lastly, our running configurations, controlled by runningConfiguration.py, is a basic object with four types of methods. First the __init()__ called when a new running configuration is created, populating all metrics with their default values or past values(if loaded on bootstrapper); getters and setters for each configuration and lastly a save method, that writes into the .xml file the current values. Note that the save method is called at the end of any setter operation.

5.2 Power BI User Interface

Our final user, surgeons in charge of surgery allocation, must have a graphical interface to consume our results. For that effect, we created an addition to the already implemented and in usage, LIC dashboard. We have chosen to add onto this view, instead of producing a new one, on account of our limitations and the fact that this dashboard is already largely adopted by the clinical community of CHUSJ, with an average of 161 monthly connections since January 2021.

5.2.1 Data Importation and Transformation

In Power BI, there are two major phases of data handling (three in dynamic data transformation, not used in our implementation), we have the data importation, written in M code and the data transformation/analysis written in DAX code. In our case, as data is mostly prepared outside of the BI ecosystem, we didn't had a need for major DAX querying, besides some String prettifying and conditional formating.

On the snippet 5.8 we have the M code responsible for importing the data of our BI database. Here we have some refactoring for keying alike our column names with the rest of the model, we then eliminate interventions for dates long past due, next some columns change their value type, to match similar elements on other tables and finally, the table is sorted, for ease of access.

```
let
    Origem = Sql.Databases(<REDACTED>),
    Bi_GestaoBlocos = Origem{[Name=<REDACTED>]}[Data],
    PredProcedure = Bi_GestaoBlocos{[Schema="dbo",Item="PredProcedure"
   ]}[Data],
    #"Linhas Ordenadas" = Table.Sort(PredProcedure,{{"bestCase", Order.
   Descending}}),
    #"Colunas com Nome Mudado" = Table.RenameColumns(#"Linhas Ordenadas"
    ,{{"NUM_LISTA_ESPERA", "N_LIC", type number}, {"COD_INTERV_CIRURGICA"
    , "COD_INTERV", {"CIRUR_PRINCIPAL", "MEDICO"}}),
    #"Linhas Filtradas1" = Table.SelectRows(#"Colunas com Nome Mudado",
    each [DTA_INTERVENCAO] > #date(2021, 8, 01))
in
    #"Linhas Filtradas1"
```

Listing 5.8: Importation of PredProcedure via M Code

5.2.2 Graphical User Interface

Data imported, the final step is to display it to the user. As described on 5.2's preamble, our interface was implemented on top of a pre-existing dashboard, and for that effect it's good to start with understanding this dashboard's design policy and look-and-feel. Do note that, this dashboard is mostly outside of our project's scope.

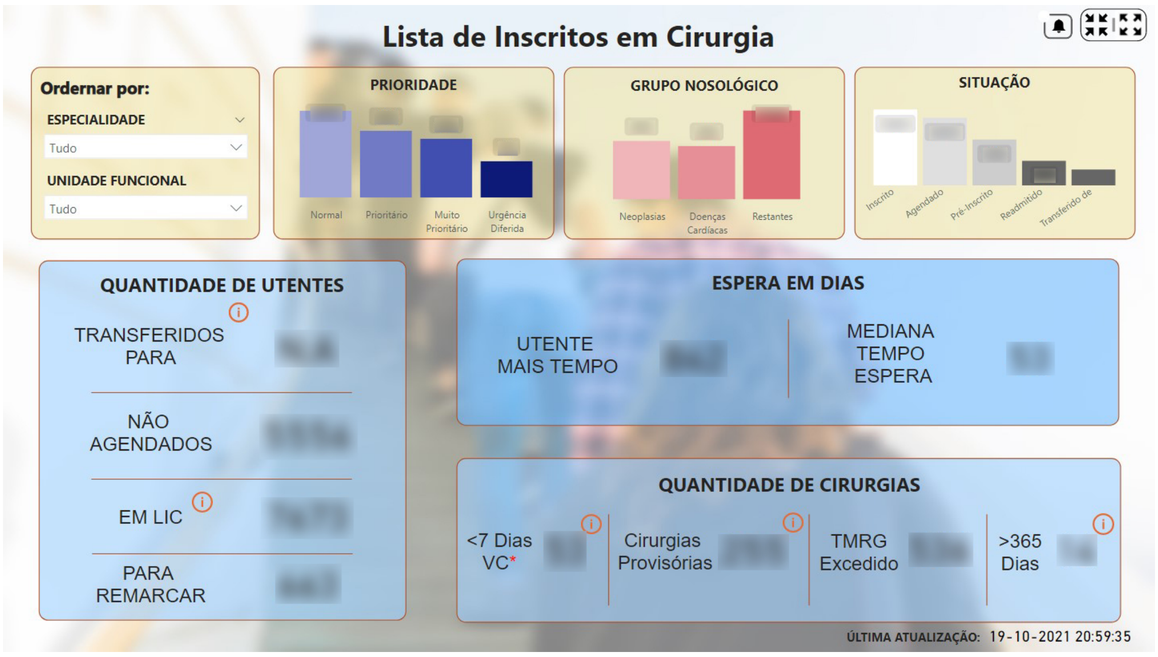
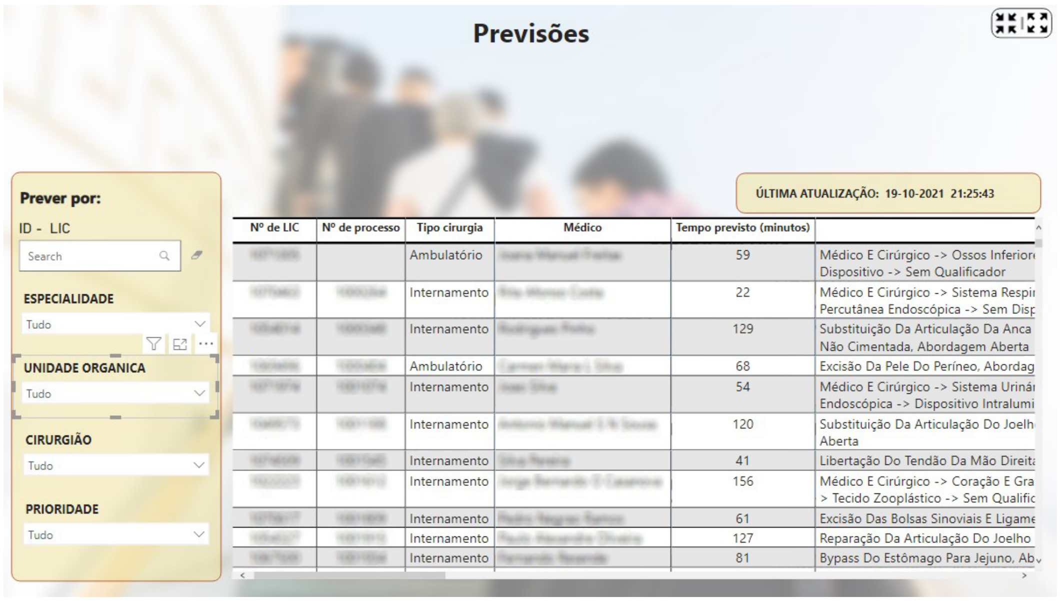


Figure 5.4: First page of the BI dashboard

Image 5.4 shows the opening page of the dashboard LIC. We can ascertain a few key features:

- Few and large information cards - Data is presented in a non-convoluted way;
- Soft colors and rounded corners - The interface keeps a simple yet refined look;
- Explicit slicer/filter options - Using dropboxes, the user can drill through the data;

Keeping this philosophy in mind, our implementation goes as follows. On 5.5 we have the first page of interaction; here, our data is presented on a table format, for the user to explore. On top we have a short synopsis of how many interventions are present on our table and when were they calculated. To the left, slicers are present for the user to either search for a specific intervention, or to analyze the information based on surgical speciality, surgeon, priority and others.



Previsões

Prever por:

ID - LIC

Search

ESPECIALIDADE: Tudo

UNIDADE ORGANICA: Tudo

CIRURGIÃO: Tudo

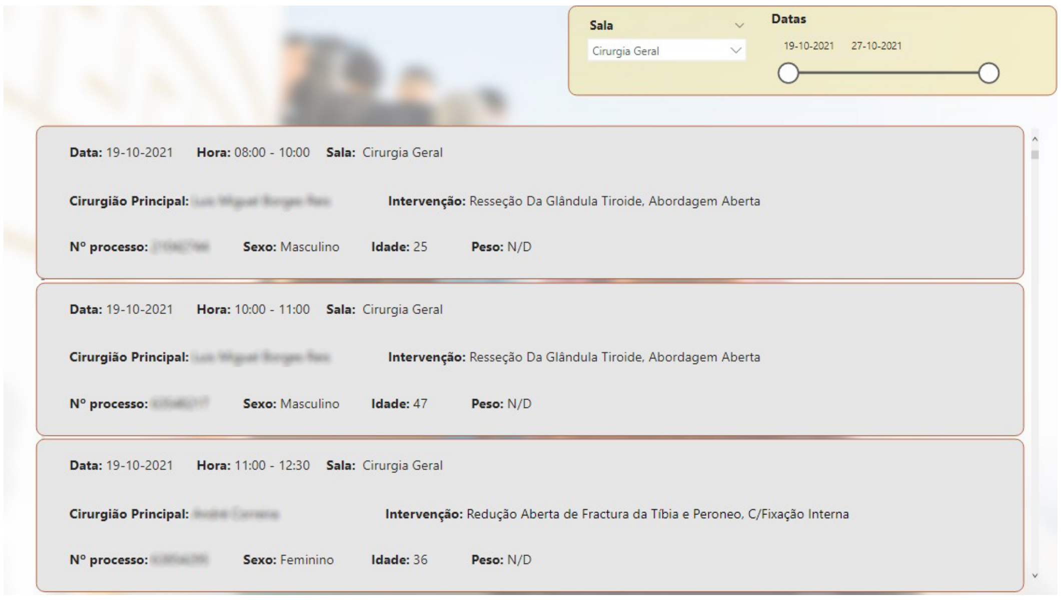
PRIORIDADE: Tudo

ÚLTIMA ATUALIZAÇÃO: 19-10-2021 21:25:43

Nº de LIC	Nº de processo	Tipo cirurgia	Médico	Tempo previsto (minutos)	
[redacted]		Ambulatório	[redacted]	59	Médico E Cirúrgico -> Ossos Inferior Dispositivo -> Sem Qualificador
[redacted]	[redacted]	Internamento	[redacted]	22	Médico E Cirúrgico -> Sistema Respi Percutânea Endoscópica -> Sem Disp
[redacted]	[redacted]	Internamento	[redacted]	129	Substituição Da Articulação Da Anca Não Cimentada, Abordagem Aberta
[redacted]	[redacted]	Ambulatório	[redacted]	68	Excisão Da Pele Do Períneo, Abordag
[redacted]	[redacted]	Internamento	[redacted]	54	Médico E Cirúrgico -> Sistema Urin Endoscópica -> Dispositivo Intralumi
[redacted]	[redacted]	Internamento	[redacted]	120	Substituição Da Articulação Do Joelh Aberta
[redacted]	[redacted]	Internamento	[redacted]	41	Libertação Do Tendão Da Mão Direit
[redacted]	[redacted]	Internamento	[redacted]	156	Médico E Cirúrgico -> Coração E Gra > Tecido Zooplástico -> Sem Qualific
[redacted]	[redacted]	Internamento	[redacted]	61	Excisão Das Bolsas Sinoviais E Ligam
[redacted]	[redacted]	Internamento	[redacted]	127	Reparação Da Articulação Do Joelho
[redacted]	[redacted]	Internamento	[redacted]	81	Bypass Do Estômago Para Jejuno, Ab

Figure 5.5: Prediction page of the BI dashboard

If we dive into the schedule on image 5.6, we'll a see simple interface with two slicers and a table. These slicers allows to check on a single OR, on an array of days, while the table tries to mimic the PDF currently used to note and distribute the schedule. We may then right click on the table to export these values to be utilized as the user may seem fit.



Sala: Cirurgia Geral

Datas: 19-10-2021 27-10-2021

Data: 19-10-2021 **Hora:** 08:00 - 10:00 **Sala:** Cirurgia Geral

Cirurgião Principal: [redacted] **Intervenção:** Resseção Da Glândula Tiroide, Abordagem Aberta

Nº processo: [redacted] **Sexo:** Masculino **Idade:** 25 **Peso:** N/D

Data: 19-10-2021 **Hora:** 10:00 - 11:00 **Sala:** Cirurgia Geral

Cirurgião Principal: [redacted] **Intervenção:** Resseção Da Glândula Tiroide, Abordagem Aberta

Nº processo: [redacted] **Sexo:** Masculino **Idade:** 47 **Peso:** N/D

Data: 19-10-2021 **Hora:** 11:00 - 12:30 **Sala:** Cirurgia Geral

Cirurgião Principal: [redacted] **Intervenção:** Redução Aberta de Fractura da Tíbia e Peroneo, C/Fixação Interna

Nº processo: [redacted] **Sexo:** Feminino **Idade:** 36 **Peso:** N/D

Figure 5.6: Scheduling page of the BI dashboard

Chapter 6

Solution Appraisal

As our implementation cycle draws to an end, we turn our focus on the assessment of our solution. Furthermore, as our product relies on machine learning, only through a thorough evaluation can we ensure our success. As such, in this chapter, we will go over the case study developed, our methodology, the metrics we will be monitoring, results and our analysis over them.

6.1 Case Study and Test Data

As described on 2.2.1, more specifically, when talking about ICD-10, we referred the high granularity of this coding system without commenting on the impact it would have on our solution. ICD-10 has over 87000 different codes and only was implemented mid-2020 on CHUSJ. This makes it challenging to build a dataset we can trust for our models, as we don't have many historical cases and even for those we have, they have been diluted in coding (i.e. the code '0GTK0ZZ' and '0GTK4ZZ' are basically clinical interchangeable, but different ML context).

In order to build our case study, we first searched for the 100 most performed surgeries in CHUSJ. From here, we removed any surgery with less than 50 instances and less than an average of 60 minutes for these would have a low impact on the OR production. Lastly, we searched for surgery with high standard deviation and from these we have chosen those that would be easier to contact surgeons, to enquire about their expert opinion on surgical duration. The results of our search can be verified on Appendix B, but in summation, 8 interventions were classified as potential candidates for proof of concept.

Through this evaluation we will be using all the studied interventions relating to ICD-10, with special attention to '0GTK0ZZ' - Resection of Thyroid Gland, Open Approach. This intervention was part of our original study, having both an high variance and high impact on the OR production. Furthermore, the process of evaluating this procedure and its results was supervised and advised by a surgeon with high expertise on the topic. With this, we can be confident on the meaning of our results.

For this, we built a dataset of the interventions in study, since the ICD-10 implementation until 26/09/2021, containing 817 instances.

Lastly, on our case study, we will pit the algorithms MLR, RFR and GBRT against each other to see which has the best and most reliable performance. Furthermore, we will compare these approaches against a arithmetic average (AVG) of procedure time per surgeon, as a benchmark.

6.2 Configuration and Methodology

For our test we parameterized our models in accordance with our genetic algorithm. The optimizer itself was defined following the hyper-parameters as a crossover rate of 0.5, a random mutation of 0.1, on a population of 50 through 1000 generations. These values are as suggested by Wicaksono and Afif 2018. The defined hyper-parameters, selected via genetic algorithm, are as follow:

MLR		
Parameter	**Type**	**Value**
fit_intercept	Boolean	True
normalize	Boolean	False

Table 6.1: MLR's hyper parameters

RFR		
Parameter	**Type**	**Value**
n_estimators	int	132
criterion	Categorical	"squared error"
bootstrap	Boolean	True
max_depth	int	None
min_samples_split	int	3
min_samples_leaf	int	2

Table 6.2: RFR's hyper parameters

GBRT		
Parameter	**Type**	**Value**
n_estimators	int	181
max_depth	int	4
min_samples_split	int	2
learning_rate	float	0.08
loss	Categorical	"ls"
criterion	Categorical	"squared error"

Table 6.3: GBRT's hyper parameters

Furthermore, besides these parameters, we've enabled scikit's multi-threading (n_jobs = -1, all processors) every time we could. The models were ran 5 times and their final success metrics were averaged.

After the regression models are completed, the metrics of Mean Squared Error (MSE), Root Mean Squared Error (RMSE) are used to calculate R^2, along side the accuracy metric described at 4.3 Lastly, the best performing model outputs will be put through our scheduling algorithm and we'll evaluate how much time could be saved or lost, when using this method compared what happened in the historical data.

6.3 Success Metrics

In this study, we would define success by basing ourselves on the work of Master et al. 2017. To be more specific, we considered as positive outcome, any model with a R^2 above 0.30 with 70% true positives of the accuracy metric defined at 4.3. MSE are RMSE used to calculate R^2 but still retain important information about the discrepancy between our model and reality. We also considered any time potentially saved, thanks to our scheduling algorithm to be an positive outcome.

On not so much academic standpoint, we would also like to monitor the training and running time for these models, as they will have an impact in future iterations, when more surgeries are being processed.

6.4 Results

After our tests we obtained the following results:

R^2				
Surgery	**AVG**	**MLR**	**RFR**	**GBRT**
02RF08Z	-0.03	-0.31	0.41	0.32
02100Z9	0.19	0.13	0.29	0.39
0GTK0ZZ	0.08	-0.01	0.38	0.32
0JQ80ZZ	0.30	0.23	0.34	0.49
0WUF0JZ	0.29	0.21	0.31	0.41
0HBT0ZZ	0.12	0.08	0.29	0.43
Result	**0.15**	**0.06**	**0.34**	**0.39**

Table 6.4: R^2 results

Prediction Accuracy				
Classification	**AVG**	**MLR**	**RFR**	**GBRT**
Correct	0.43	0.37	0.65	0.71
Overestimate	0.06	0.17	0.22	0.08
Underestimate	0.51	0.46	0.13	0.21

Table 6.5: Accuracy results

Time taken				
Task	**AVG**	**MLR**	**RFR**	**GBRT****
Modeling	0	0.12	0.23	0.96
Running	0.08	0.07	0.09	0.32
Total	0.08	0.19	0.32	1.28

Table 6.6: Time taken in seconds, to execute (** multi-threading not available)

6.5 Results Analysis

Going over our case study, on 6.4, we can ascertain that GBRT was the better performer on our metrics, in some cases by a large margin. With a R^2 of 0.39 and accuracy of 0.71 we surpassed our success metrics with some leeway. Still, there is a caveat, as we discussed in 4.3, what surgeons look to reduce is the need for rescheduling, meaning reducing underestimation in the duration of surgery; if we look at the table 6.5, GBRT did underestimate more then overestimate, being behind RFR for that metric. This implies that this model might be more sensible to outliers. When discussing this outcome with the surgeons responsible, we analyzed some cases where patients with very specific clinical situations had exceptionally long interventions, and in this case, the multitude of generated trees in RFR did come in handy.

Going over the other algorithms, RFR did had some good results, even though it didn't beat our metrics with a R^2 of 0.34 and accuracy of 0.648. In comparison, MLR has had very poor results, not managing to beat the arithmetic average with an R^2 of 0.06 and an accuracy of 0.510. This might be a result of poor hyper-parameter optimization, though we believe it just consists in a problem too complicated for this kind of implementation. In summation, GBRT is the better performing model, while RFR stands and a good alternative, less subjective to the impact of outliers; in contrast, MLR performed very poorly and might be removed in future iterations.

In case of the scheduling algorithm, assuming our schedule would fit perfectly and using the values discerned by GBRT, we observe a difference of -9:23 hours (which we most round down to -9:00, as scheduling happens in 30 minute blocks), meaning we saved 9 hours in the space of 11 months. While this might seem a minuscule time slot, we most remember we are talking of 6 surgeries in the entirety of the operating block. If this time save was to be increased to other surgeries, the impact would be vast.

Lastly, briefly mentioning the execution times for these models, there were no instances of a model taking more then 5 seconds. This is in part to be expected, as the datasets supplied is still growing. Still, in between each algorithm GBRT does stick out, as multi-threading via scikit's libraries was not available at the time of testing. As such, this value isn't comparable.

Chapter 7

Conclusion and Future Work

The present chapter serves as a way to present the final conclusions of the project, as well as some personal comments, not forgetting to mention the future work that there is to be done.

7.1 Conclusions

Regarding the appreciation of the developed work, I am personally very satisfied about the solution that was developed, as well as with the obtained results. The problem and the requirements presented at the beginning were successfully taken into account, being correctly development in the best standards and practices of Software Engineering, which resulted in a piece of software that meets the proposed objectives. Although the success metrics were a little bit below the expected results taken from various literature, they demonstrate the capabilities of the project to a great extent, in the possibility of having a much bigger dataset. Furthermore, the dataset in our case study was in many cases a difficult task, as it consisted of intervention with a great degree of variance.

7.2 Future Work

Although that this project's phase was a success, there is always room for improvement and a few action points were identified, specially considering this work will continue beyond the scope of the document presented.

In our immediate future, we want to focus on reducing underestimations on our models as these constitute the biggest threat for a orderly schedule. There is also the possibility of implementing new machine learning models for the surgery time regression, which would make the results more accurate.

Then, if our solution is approved by the CHUSJ's medical community, there is an opportunity to develop it as a standalone application, outside the reporting services, as well as outside the Microsoft 365 ecosystem. This will bring not only more ease of use for the end user, but maybe new metrics might be exploitable, for example, integrating in our models as a feature, the expert opinion of the surgeon building his or hers schedule.

Through the following months we would care to explore in further detail the impact that certain combinations of surgeons could have in the surgery time, for example surgeon A and B being exemplar when working alone, but when together, the surgery would take longer.

Last but not least, as described in literature, there are many methodologies to make these kind of predictions, with the support of clinic metrics from the patient. Although we tried to test an implementation along those lines, the difficulty of access to the data and the small size of the datasets, ultimately led to abandoning this approach rather soon in the development cycle, as the models were greatly overfitted. It would be very interesting to pick up this methodology again in the future, when access to historical data is more abundant; alas we are at the mercy of data.

Bibliography

Cardoen, Brecht, Erik Demeulemeester, and Jeroen Beliën (2010). “Operating room planning and scheduling: A literature review”. In: *European Journal of Operational Research* 201.3, pp. 921–932. issn: 03772217. doi: 10.1016/j.ejor.2009.04.011. url: https://www.sciencedirect.com/science/article/pii/S0377221709002616.

Carlisle, Stephanie (2018). “Software: Tableau and Microsoft Power BI”. In: *Technology|Architecture + Design* 2.2, pp. 256–259. doi: 10.1080/24751448.2018.1497381. eprint: https://doi.org/10.1080/24751448.2018.1497381. url: https://doi.org/10.1080/24751448.2018.1497381.

Devi, S Prasanna, K Suryaprakasa Rao, and S Sai Sangeetha (2012). “Prediction of Surgery Times and Scheduling of Operation Theaters in Optholmology Department”. In: *Journal of Medical Systems* 36.2, pp. 415–430. issn: 1573-689X. doi: 10.1007/s10916-010-9486-z. url: https://doi.org/10.1007/s10916-010-9486-z.

Dey, Ayon (2016). “Machine learning algorithms: a review”. In: *International Journal of Computer Science and Information Technologies* 7.3, pp. 1174–1179.

Erekat, Asala et al. (Jan. 2020). “Efficient operating room planning using an ensemble learning approach to predict surgery cancellations”. In: *IISE Transactions on Healthcare Systems Engineering* 10.1, pp. 18–32. issn: 24725587. doi: 10.1080/24725579.2019.1641576. url: https://www.tandfonline.com/doi/abs/10.1080/24725579.2019.1641576.

Fehlberg, Trafford et al. (May 2019). “The surgical burden of disease and perioperative mortality in patients admitted to hospitals in Victoria, Australia: a population-level observational study”. eng. In: *BMJ open* 9.5, e028671–e028671. issn: 2044-6055. doi: 10.1136/bmjopen-2018-028671. url: https://pubmed.ncbi.nlm.nih.gov/31118179%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6549668/.

Fernández, Raquel and M. Bonillo (Dec. 2007). “The concept of perceived value: A systematic review of the research”. In: *Marketing Theory - MARK THEORY* 7, pp. 427–451. doi: 10.1177/1470593107083165.

Hoel, Michael and Erik Magnus Sæther (2003). “Public health care with waiting time: the role of supplementary private health care”. In: *Journal of health economics* 22.4, pp. 599–616.

Holmes, Geoffrey, Andrew Donkin, and Ian H Witten (1994). *WEKA: A Machine Learning Workbench*. Tech. rep.

Huang, Ching-Chieh et al. (2021). “A Machine Learning Study to Improve Surgical Case Duration Prediction”. In: doi: 10.1101/2020.06.10.20127910. url: https://doi.org/10.1101/2020.06.10.20127910.

Ihaka, Ross (2020). *R project*. url: https://cran.r-project.org/doc/FAQ/R-FAQ.html#What-is-R_003f (visited on 02/28/2021).

Isaksen, B. and V. Bertacco (2006). “Veri cation Through the Principle of Least Astonishment”. In: *2006 IEEE/ACM International Conference on Computer Aided Design*, pp. 860–867. doi: 10.1109/ICCAD.2006.320090.

JetBrains (2020). *Devecosystem 2020 Study*. url: https://www.jetbrains.com/lp/devecosystem-2020/ (visited on 02/28/2021).

Koen, Peter et al. (2001). “Providing clarity and a common language to the “fuzzy front end””. In: *Research-Technology Management* 44.2, pp. 46–55.

Kumar, Divakar (Sept. 2021). *How did The Random Forest Algorithm work in Machine Learning?* url: https://medium.com/@divakarkpandey/how-did-the-random-forest-algorithm-work-in-machine-learning-9e044573898b.

Lambora, Annu, Kunal Gupta, and Kriti Chopra (2019). “Genetic Algorithm- A Literature Review”. In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 380–384. doi: 10.1109/COMITCon.2019.8862255.

Master, Neal et al. (Aug. 2017). “Improving predictions of pediatric surgical durations with supervised learning”. In: *International Journal of Data Science and Analytics* 4.1, pp. 35–52. issn: 23644168. doi: 10.1007/s41060-017-0055-0.

Microsoft (2020). *Microsoft 365 Ecosystem*. url: https://github.com/ealtili/Blog/blob/master/Microsoft365/Cloud_based_collaboration.md (visited on 03/05/2021).

– (Oct. 2021). *What is automated ML? AutoML - Azure Machine Learning*. url: https://docs.microsoft.com/en-us/azure/machine-learning/concept-automated-ml.

Miles, Eleanor R. (1989). “Lawrence Delos Miles’ "Techniques of value analysis and engineering"”. In: *Ekistics* 56.336/337, pp. 119–121. issn: 00132942. url: http://www.jstor.org/stable/43620667.

Milicchio, Franco (2007). “The Unix KISS: A Case Study”. In: *arXiv preprint cs/0701021*.

Nygren, Jonas (2006). “The metabolic effects of fasting and surgery”. In: *Best practice & research Clinical anaesthesiology* 20.3, pp. 429–438.

OdM (2020). *Ordem dos Médicos - Relatório de intervenções*. url: https://ordemdosmedicos.pt/wp-content/uploads/2021/02/Por_especialidade_abi-copy.pdf (visited on 02/04/2021).

Osterwalder, Alexander (2005). *What is a business model*. url: https://web.archive.org/web/20061213141941/http://business-model-design.blogspot.com/2005/11/what-is-business-model.html (visited on 02/19/2021).

– (2008). *What is a business model, revisited*. url: https://web.archive.org/web/20080906034734/http://business-model-design.blogspot.com/2008/07/what-is-business-model.html (visited on 02/19/2021).

Osterwalder, Alexander et al. (2014). *Value proposition design: How to create products and services customers want*. John Wiley & Sons.

Osterwalder Alexander; PIGNEUR, Yves. (2011). “Lawrence Delos Miles’ "Business Model Generation: Inovação em Modelos de Negócios"”. In:

Prieto González, Lisardo, Gerrit Tamm, and Vladimir Stantchev (July 2016). “Towards a Software Engineering Approach for Cloud and IoT Services in Healthcare”. In: vol. 9789, pp. 439–452. isbn: 978-3-319-42088-2. doi: 10.1007/978-3-319-42089-9_31.

Rouhani, Saeed, Sara Asgari, and Vahid Mirhosseini (Jan. 2012). *Review Study: Business Intelligence Concepts and Approaches*.

Saaty, Thomas L (2008). “Decision making with the analytic hierarchy process”. In: *International journal of services sciences* 1.1, pp. 83–98.

– (1988). “What is the Analytic Hierarchy Process?” In: *Mathematical Models for Decision Support*. Ed. by Gautam Mitra et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 109–121. isbn: 978-3-642-83555-1.

Sánchez-Fernández, Raquel, M Ángeles Iniesta-Bonillo, and Morris B Holbrook (2009). “The conceptualisation and measurement of consumer value in services”. In: *International Journal of Market Research* 51.1, pp. 1–17.

SNS (2021). *SNS - Fluxos financeiros no SNS*. url: `http://www.cns.min-saude.pt/wp-content/uploads/2017/09/Fluxos_Financeiros_SNS_3.11.2017_final.pdf` (visited on 02/05/2021).

Thirumagal, R et al. (2014). "ETL tools in data mining—a review". In: *International Journal of Research in Computer Applications & Robotics* 2.1, pp. 62–69.

WHO (1978). *International classification of diseases : [9th] ninth revision, basic tabulation list with alphabetic index*. WHO, 331 p.

– (2011). *International Statistical Classification of Diseases and Related Health Problems 10th Revision*. WHO. isbn: 9789241548342. url: `www.who.int`.

– (2016). *WHO's Surgery Definition*. url: `https://gateway.euro.who.int/en/indicators/hfa_539-6031-total-number-of-inpatient-surgical-procedures-per-year/visualizations/#id=19634&tab=table` (visited on 02/04/2021).

– (2021). *WHO's Surgery Definition*. url: `https://www.who.int/surgery/SurgeryDebasworldbank.pdf?ua=1` (visited on 02/02/2021).

Wicaksono, Ananto and Ahmad Afif (Jan. 2018). "Hyper Parameter Optimization using Genetic Algorithm on Machine Learning Methods for Online News Popularity Prediction". In: *International Journal of Advanced Computer Science and Applications* 9. doi: `10.14569/IJACSA.2018.091238`.

Woodall, Tony (Jan. 2003). "Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis". In: *Academy of Marketing Science Review* 12.

Zuck, Jonathan and J. Peter DONLON (1997). In: *Developing to Distributed N-Tier Client/Server Applications*. society of actuaries, pp. 1–27.

Appendix A

Microsoft 365 Ecosystem

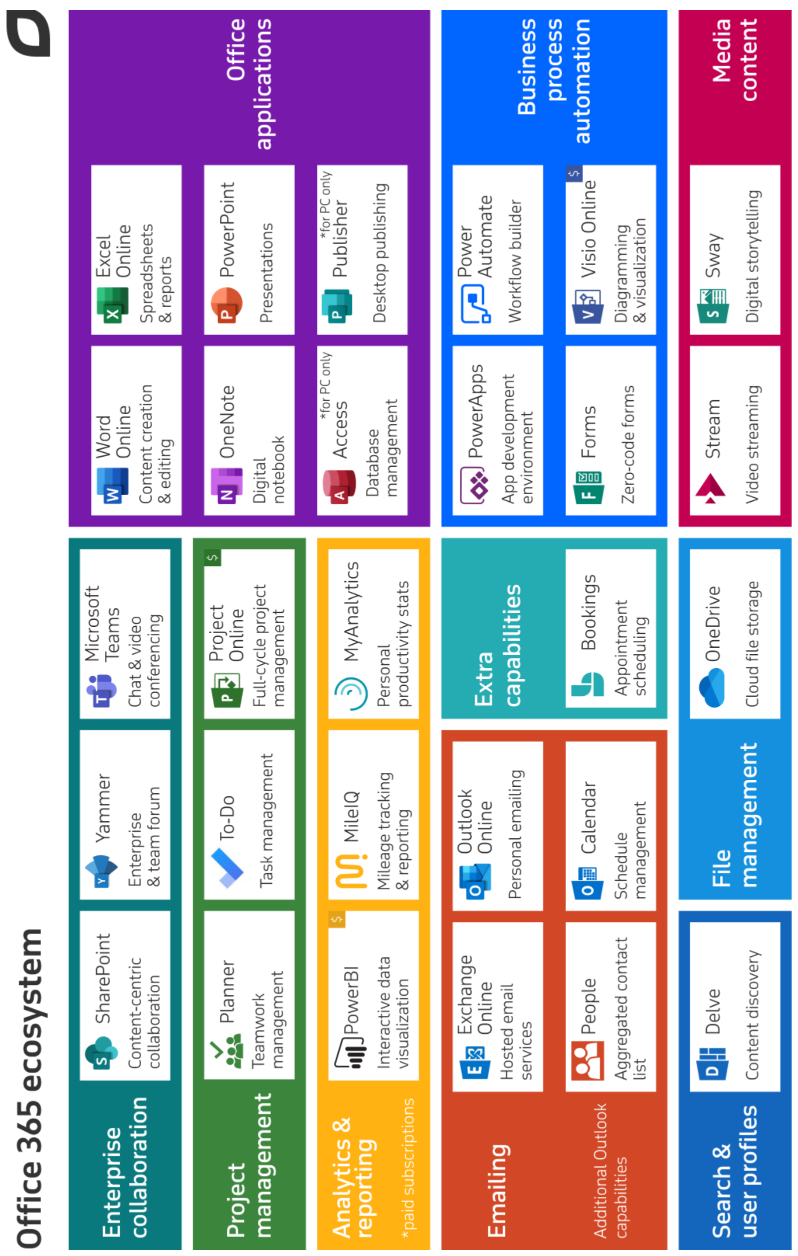


Figure A.1: Microsoft 365 Ecosystem (source Microsoft 2020)

Appendix B

Case study 21-03-2021

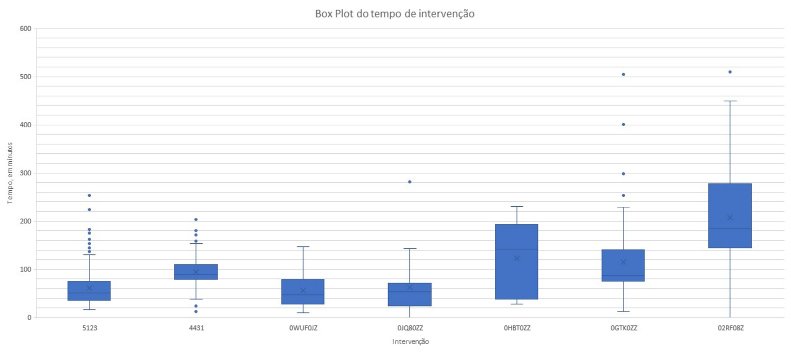
Metodologia

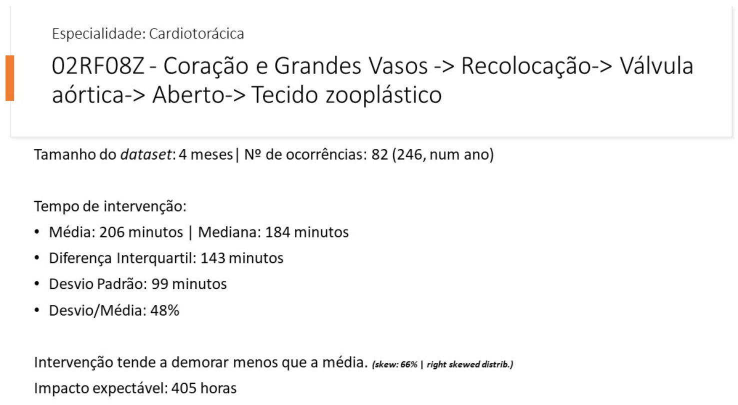
A escolha de potenciais intervenções para o projeto, baseou-se numa amostra de 14 meses, da base de dados BSIMPLE, fornecido pelo Bloco Central. Com este, foram:

1. Criados dois *datasets* de intervenções para o ICD9 e ICD10, baseado na base de dados BSIMPLE.
2. Excluídas as intervenções com menos de 50 ocorrências, por ano.
3. Excluídas as intervenções com menos de 45 minutos de distância interquartil e menos de 60 minutos de média*.
4. Calculada a relação desvio padrão/mediana e o impacto expectável do desvio (desvio padrão **x** nº intervenções num ano).

* Ambas as condições tem que ser verdade para a exclusão

Gráfico Box and Whisker



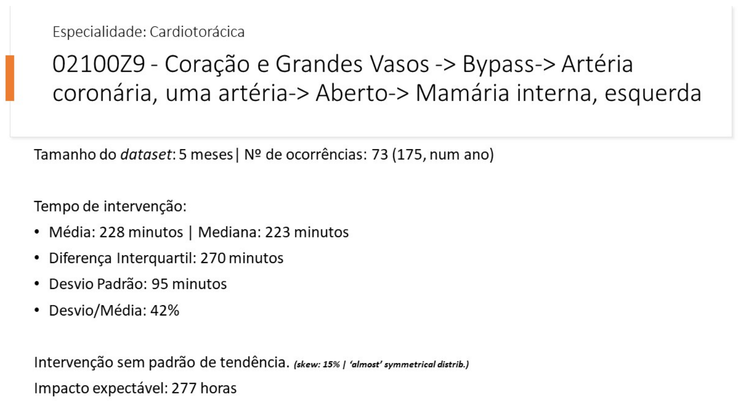

Especialidade: Cardiotorácica

02RF08Z - Coração e Grandes Vasos -> Recolocação-> Válvula aórtica-> Aberto-> Tecido zooplástico

Tamanho do *dataset*: 4 meses | Nº de ocorrências: 82 (246, num ano)

Tempo de intervenção:
- Média: 206 minutos | Mediana: 184 minutos
- Diferença Interquartil: 143 minutos
- Desvio Padrão: 99 minutos
- Desvio/Média: 48%

Intervenção tende a demorar menos que a média. ***(skew: 66% | right skewed distrib.)***
Impacto expectável: 405 horas

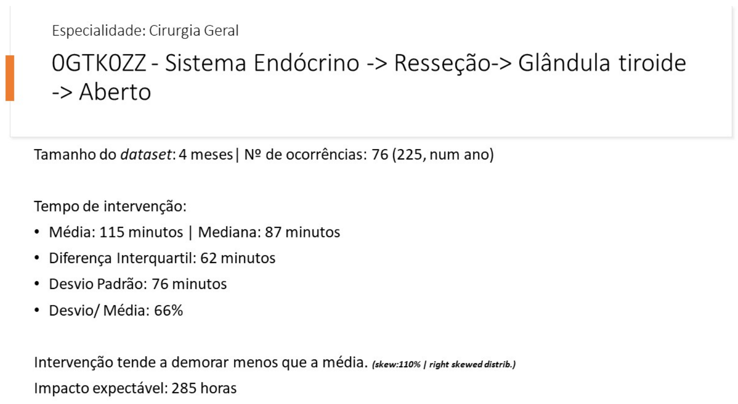

Especialidade: Cardiotorácica

02100Z9 - Coração e Grandes Vasos -> Bypass-> Artéria coronária, uma artéria-> Aberto-> Mamária interna, esquerda

Tamanho do *dataset*: 5 meses | Nº de ocorrências: 73 (175, num ano)

Tempo de intervenção:
- Média: 228 minutos | Mediana: 223 minutos
- Diferença Interquartil: 270 minutos
- Desvio Padrão: 95 minutos
- Desvio/Média: 42%

Intervenção sem padrão de tendência. ***(skew: 15% | 'almost' symmetrical distrib.)***
Impacto expectável: 277 horas


Especialidade: Cirurgia Geral

0GTK0ZZ - Sistema Endócrino -> Resseção-> Glândula tiroide -> Aberto

Tamanho do *dataset*: 4 meses | Nº de ocorrências: 76 (225, num ano)

Tempo de intervenção:
- Média: 115 minutos | Mediana: 87 minutos
- Diferença Interquartil: 62 minutos
- Desvio Padrão: 76 minutos
- Desvio/ Média: 66%

Intervenção tende a demorar menos que a média. ***(skew:110% | right skewed distrib.)***
Impacto expectável: 285 horas

Especialidade: Cirurgia Geral

0JQ80ZZ - Tecido subcutâneo e Fáscia -> Reparação-> Fáscia e tecido subcutâneo, abdómen-> Aberto

Tamanho do *dataset*: 5 meses | Nº de ocorrências: 29 (70, num ano)

Tempo de intervenção:
- Média: 62 minutos | Mediana: 54 minutos
- Diferença Interquartil: 50 minutos
- Desvio Padrão: 56 minutos
- Desvio/ Média: 90%

Intervenção tende a demorar menos que a média. *(skew: 42% | right skewed distrib.)*
Impacto expectável: 65 horas

Especialidade: Cirurgia Geral

0WUF0JZ - Regiões Anatómicas, Gerais-> Suplementação-> Parede abdominal-> Aberto -> Substituto Sintético

Tamanho do *dataset*: 4 meses | Nº de ocorrências: 38 (114, num ano)

Tempo de intervenção:
- Média: 55 minutos | Mediana: 47 minutos
- Diferença Interquartil: 48 minutos
- Desvio Padrão: 33 minutos
- Desvio/ Média: 60%

Intervenção tende a demorar menos que a média. *(skew: 72% | right skewed distrib.)*
Impacto expectável: 63 horas

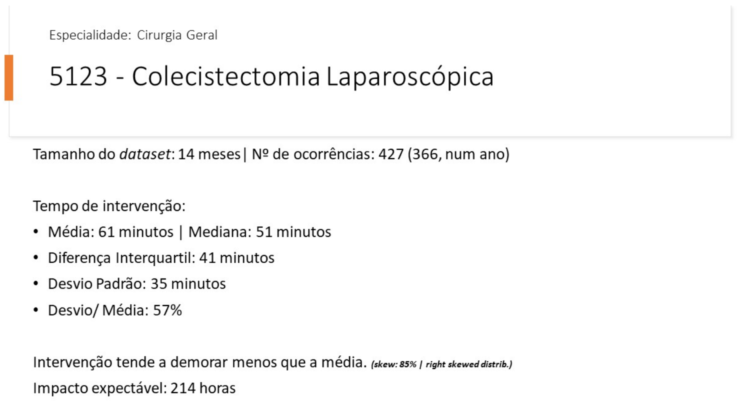
Especialidade: Cirurgia Geral

0HBT0ZZ - Pele e Mama -> Excisão-> Mama, direita-> Aberto

Tamanho do *dataset*: 4 meses | Nº de ocorrências: 12 (36, num ano)

Tempo de intervenção:
- Média: 122 minutos | Mediana: 142 minutos
- Diferença Interquartil: 150 minutos
- Desvio Padrão: 78 minutos
- Desvio/ Média: 63%

Intervenção tende a demorar mais que a média. *(skew: -77% | left skewed distrib.)*
Impacto expectável: 47 horas


Especialidade: Cirurgia Geral

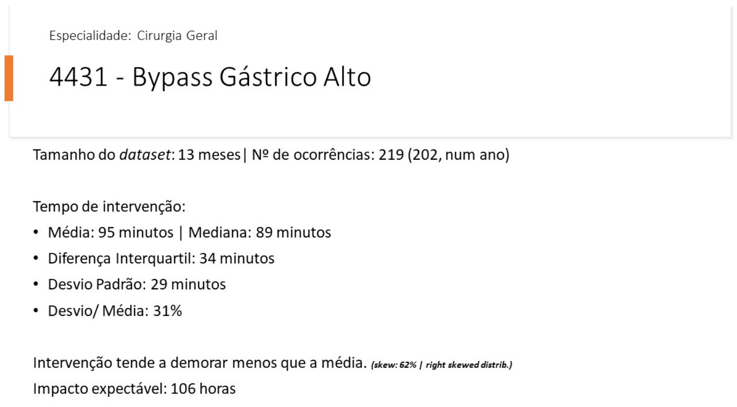
5123 - Colecistectomia Laparoscópica

Tamanho do *dataset*: 14 meses | Nº de ocorrências: 427 (366, num ano)

Tempo de intervenção:

- Média: 61 minutos | Mediana: 51 minutos
- Diferença Interquartil: 41 minutos
- Desvio Padrão: 35 minutos
- Desvio/ Média: 57%

Intervenção tende a demorar menos que a média. ***(skew: 85% | right skewed distrib.)***
Impacto expectável: 214 horas


Especialidade: Cirurgia Geral

4431 - Bypass Gástrico Alto

Tamanho do *dataset*: 13 meses | Nº de ocorrências: 219 (202, num ano)

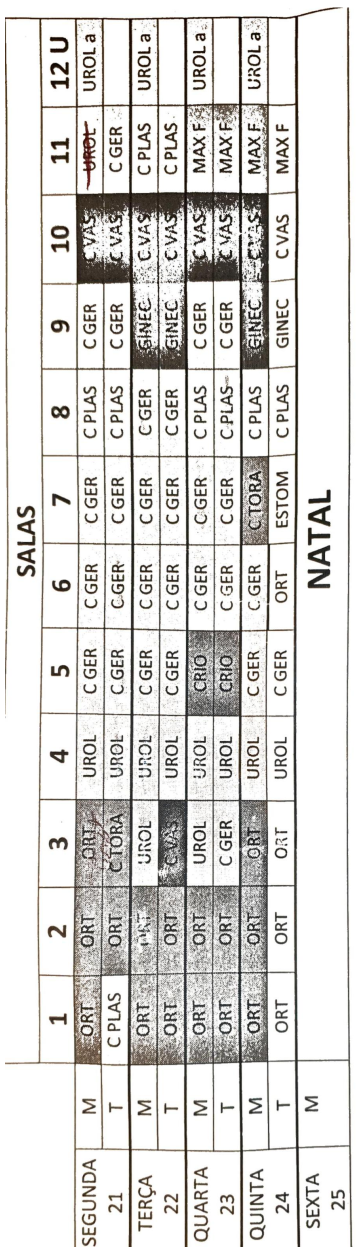
Tempo de intervenção:

- Média: 95 minutos | Mediana: 89 minutos
- Diferença Interquartil: 34 minutos
- Desvio Padrão: 29 minutos
- Desvio/ Média: 31%

Intervenção tende a demorar menos que a média. ***(skew: 62% | right skewed distrib.)***
Impacto expectável: 106 horas

Appendix C

Master Surgery Speciality Schedule



		SALAS											
		1	2	3	4	5	6	7	8	9	10	11	12 U
SEGUNDA 21	M	ORT	ORT	~~ORT~~	UROL	C GER	C GER	C GER	C PLAS	C GER	C VAS	~~UROL~~	UROL a
	T	C PLAS	ORT	C TORA	UROL	C GER	C GER	C GER	C PLAS	C GER	C VAS	C GER	
TERÇA 22	M	ORT	ORT	UROL	UROL	C GER	C GER	C GER	C GER	GINEC	C VAS	C PLAS	UROL a
	T	ORT	ORT	C VAS	UROL	C GER	C GER	C GER	C GER	GINEC	C VAS	C PLAS	
QUARTA 23	M	ORT	ORT	UROL	UROL	CRIO	C GER	C GER	C PLAS	C GER	C VAS	MAX F	UROL a
	T	ORT	ORT	C GER	UROL	CRIO	C GER	C GER	C PLAS	C GER	C VAS	MAX F	
QUINTA 24	M	ORT	ORT	ORT	UROL	C GER	C GER	C TORA	C PLAS	GINEC	C VAS	MAX F	UROL a
	T	ORT	ORT	ORT	UROL	C GER	ORT	ESTOM	C PLAS	GINEC	C VAS	MAX F	
SEXTA 25	M	NATAL											

Figure C.1: Master Surgery Speciality Schedule

Appendix D

Requirements Gathering

This was part of the process of Requirements Engineering, throughout our project

Curso do paciente: • Entrada no Bloco • Primeira abordagem(triagem) • Movido para antecâmara • Movido para Sala de Operação • Anestesia • Intervenção • Movido para recobro (UPA) • Saída do Bloco

Fases: Primeira Abordagem: O paciente é encaminhado de internamento para o bloco, à entrada são feitas um conjunto de questões e análise por 1 auxiliar e por 1 enfermeiro. • Validação da identidade do paciente e verificação da lista para intervenções • Verificar testes, consentimentos • Verificar jejum, injeções, medicamentos tomados e alergias a medicamentos (tudo isto nas últimos 24 horas)

É nesta fase colocada uma pulseira eletrónica que contem os dados da localização do paciente, identificado qual das fazes em que este se encontra, até a saída do UPA, e como tal, saída do bloco. A falha em qualquer destas fases implica que o paciente seja retornado para internamento

Antecâmara: Local de espera à frente de cada sala de operação. O paciente espera aqui pela preparação da sala. A preparação da sala é segmentada em dois eventos, a limpeza e a preparação. A limpeza depende de um auxiliar, que trabalha volante entre duas salas (1-2,3-4,5-6, ...). Após esta limpeza, a sala necessita arejar, demorando uns desejáveis 15 minutos. A preparação da sala depende da equipa da operação e do auxiliar responsável por trazer o paciente, e equipamentos preparados previamente. Este equipamento depende da intervenção.

Anestesia: Primeira fase dentro da sala de operação. É aplicada anestesia geral ou local por profissionais especializados. É necessário a presença. Esta demora espectavelmente 30 minutos a fazer efeito. A equipa para a realização da anestesia é um médico anestesista (especialista em anestesia local ou geral) e um enfermeiro anestesista.

Intervenção: Realização da intervenção programada. Isto requer uma equipa com vários membros a efetuar tarefas distintas. A equipa consiste em: • 1 a 3 Cirurgiões (da especialidade), havendo o cirurgião principal, 1º ajudante e 2º ajudante • Médico anestesista (presente desde a administração da anestesia) • 3 enfermeiros: Enfermeiro instrumentista (age na área de operação, fornecendo as ferramentas necessárias à intervenção, é um trabalhador especializado); Enfermeiro circulante (faz a ligação entre interior e o exterior da sala de operação e chega material ao instrumentista) e enfermeiro anestesista • Auxiliar (que está encarregue de duas salas) Estas intervenções têm tempos planeados, estruturados

UPA: Após a intervenção o paciente é movido para recobro. Existem 2 tipos de recobro: rápido e lento. O rápido é dedicado a intervenções mais leves e demora por volta de 2 horas.

Aqui é realizado a monitorização da recuperação do paciente, procurando encontrar qualquer fator que implique algum tipo de intervenção. O lento é uma espécie de internamento de nível 2/3 em cirurgias mais complicadas/invasivas que necessite um recobro mais demorado e/ou ventilado. Este recobro pode levar até 48 horas, no qual, no final destas, o paciente será movido para outro serviço de cuidados.

Saída do bloco: Na saída do bloco é removida a pulseira localizadora.

Bottlenecking (14/12/20)

Intervenção atrasada por falta de equipamentos não preparados por auxiliar: Na sala 8(salvo erro) do bloco, a intervenção agendada para as 08:00 só começou as 09:00 pois a auxiliar encarregue de preparar(trazer) os equipamentos necessários para a intervenção, não o fez previamente. Salas não sinalizadas corretamente sobre o estado de "Em limpeza" Agendamento do UPA excessivo, excedendo em 6 camas.

Bottlenecking (15/12/20)

De manhã, um grande número de salas atrasa, devido à necessidade de preparação de equipamentos. Em teoria este equipamento pode começar a ser preparado quando o agendamento é libertado, no dia anterior pelas 14:00 da tarde, contudo, isto não acontece. Devido as intervenções começarem atrasadas, as equipas acabam mais tarde, acabando mais tarde não tem tempo para preparar equipamentos, sem equipamentos, as intervenções do dia seguinte atrasam-se, acabando por reiniciar este ciclo de atrasos.

A presença de doentes COVID obriga à alteração de salas agendadas, pois a sala para COVID's é a 11... sendo preciso passar as atividades programadas aqui, para a sala de cirurgia geral. Esta situação também obriga à presença de um quarto enfermeiro na sala de intervenção. Este serve de supervisor das atividades durante a intervenção, garantido que não sejam quebrados protocolos de segurança. Um paciente foi chamado indevidamente para a operação, isto causou uma espera indevida de 1 hora e pouco.

Bottlenecking (16/12/20)

O dia começou com novos atrasos devido e equipamentos, contudo, em vez do equipamento normalmente circulante (título não oficial para bisturis, ferramentas, ...), tratou-se de camas por inteiro, por estarem "danificadas". Isto era informação já conhecida desde o dia anterior e deveria ter sido preparado aí. Numa questão de DQP's, a limpeza das salas continua a estar errada, com auxiliares esquecendo-se de marcar os momentos de ação.

Reunião com o Dr. António Pereira (16/12/20) - Inscrição de pacientes - Esof.Gast.Duod.

Contexto:

Cirurgião, médico ligado à neoplasia do esófago, realiza consultas externas e é professor convidado. Neste âmbito, é responsável por inscrever pacientes internados, sendo estes pacientes decididos em grupo com outros médicos. Trabalho realizado no SClínico com apoio registos maioritariamente digitais (exames/relatórios internos), mas não todos. (exames/relatórios externos).

Resumo:

O cliente demonstrou-se recetivo a criação de uma ferramenta, contudo a sua proposta afastou-se da área e das tecnologias tencionadas usar no projeto.

Diálogo:

Foi abordado como se realiza a inscrição de um paciente para intervenção cirúrgica. Os pacientes que foram inscritos já tinham sido decididos antes da reunião. Quando perguntado de onde esta lista tinha surgido, o cliente falou de que o grupo de cirurgiões escolhem os seus pacientes da LIC baseado num conjunto de fatores (talvez para sistema pericial ou SAD).

Foram realizadas 4 inscrições, no qual o processo foi demonstrado. Este é relativamente simples e rápido (ou aconteceu de ser simples e rápido nesta situação), baseado no preenchimento de uma ficha com informação identificativas e clínicas do paciente. O processo foi em suma, o seguinte: O paciente é verificado processo, identidade confirmada, é introduzido na ficha os seus dados identificativos, os dados identificativos do Dr. são automaticamente introduzidos, é validado (em teoria) que o paciente realizou as consultas de anestesia, se deu e quando deu consentimento, diagnostico reportado e prioridade deste (baseado em vários fatores). Depois disto o diagnostico e intervenção planeada são introduzidos, gerando o código ICD10.

Ao longo deste processo o dr. António baseou as suas escolhas nem relatórios em texto livre. Nem todos os relatórios a que recorreu se encontravam em formato digitalizado por originarem de fontes externas, quando abordado este assunto, o Dr. garantiu que ultimamente o número de relatórios digitais tem vido a aumentar. Todo este processo foi bastante rápido, não demorando mais que 15 segundos a encontrar a documentação e menos que 1 minuto para validar as decisões tomadas na ficha de inscrição. Depois destes registos, estes têm de ser aprovados e agendados pela administrativa no bloco, consoante a tabela de alocação das salas às especialidades. Uma versão física, impressa, da ficha é assinada e dá-se por terminado o processo.

O diálogo prossegui para a fase de bottlenecking do ponto de vista do cliente. Este referiu a falta de espaço na UPA e as falhas das consultas de anestesia (é comum médicos colocarem como realizada uma consulta que esteja marcada para antes da cirurgia, contudo, devido a atrasos, esta pode acabar por não acontecer a tempo, causando o cancelamento/adiamento da intervenção). Aproveito para elaborar que as datas do consentimento aparentaram, ao longo da reunião, depender do entendimento do clínico, ou seja, a data de consentimento foi elaborada com dia 16/12 para todos os pacientes, contudo, no único caso em que foi demonstrada prova de consentimento, esta datava de há 2 dias atras, dia 14. O cliente quando perguntado em que ferramenta gostaria de ter acesso falou sobre um sistema de aviso quando uma consulta de anestesia se encontrava em falta.

O cliente usou durante toda a interação o serviço de SClínico pelo navegador Internet Explorer, em janela maximizada com um só monitor, sem recurso a shortcuts. O seu grau de utilizador seria considerado de básico. O cliente demonstrou a facilidade de recolha da informação necessária a validação da ficha, procurando esta informação enquanto falava. Esta pesquisa foi maioritariamente breve, fora um caso no qual a devida subsecção da pesquisa não era conhecida.

Conclusões:

A falta de entusiasmo quando a produção de uma solução de apoio ao planeamento aliado ao facto de todo e qualquer atrito previsível com introduzir uma nova ferramenta num ambiente que já "funciona" faz difícil a decisão da abordagem. Isto é magnificado quando a ferramenta em questão não se encontra embutida no standard atual. A usabilidade da ferramenta também se demonstra diminuta pelos padrões de utilização do PC do cliente junto ao facto da ferramenta atual ser um serviço web completamente fora do escopo deste projeto. No outro caso, um sistema de monitorização da ficha tal como o cliente sugeriu, no momento da introdução dos dados enquanto possível, pode deter de DQP devido aos registos em papel. Por último, a solução parece ter um valor final decrescido, pelo facto que os elementos impeditivos (validação de documentos) são resolvidos "à mão" de modo extremamente rápido, numa tarefa também esta rápida, que não acontece todos os dias.

Reunião das Especialidades - 17/12/2020

Reunião breve em que se validou apenas a segmentação do bloco... depois disto, foi apresentado o meu trabalho e proposta mais "obvia". Para minha surpresa, e da Dra. Susana, aparentemente o Professor Silvestre, antigo chefe do bloco central e agora na urgência, criou uma ferramenta que monitoriza a prontidão de um paciente ser intervencionado.... (foi aplicado noutro bloco e com pouco sucesso) Falado com o Dr. Afonso, ele acredita esta situação ser peculiar, e o produto possivelmente inferior ao que nós podemos propor.

Notas Random Muita da informação em questão parece existir digitalizada, contudo, aparenta ser de vários serviços diferentes... ou seja não existe uma fonte de dados centralizada.

Muitos dos problemas que acontecem no bloco não dependem do agendamento, mas de fator humano. A solução talvez tenha de ser segmentada em estrutura de monitorização (no bloco) e estrutura de agendamento inteligente (na especialidade). Existem muitos fatores fora da autoridade da gestão dos blocos.

Os cirurgiões pareceram recetivos à criação de um mecanismo para monitorizar a preparação de um paciente para intervenção. Dia 16/12 irei ter um diálogo mais longo. Dia 18 terei um novo diálogo. Falar com o Dr. Vítor Lopes, dia 18/12 A Dra. Susana falou de um problema de informação que podia ser um trabalhito engraçado. Criar formalizações ICD10 em texto comum.