TÉCNICO LISBOA

MedClick: Last Minute Medical Appointments No-Show Management

Ricardo Miguel Oliveira de Almeida

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. André Ferreira Ferrão Couto e Vasconcelos

Examination Committee

Chairperson: Prof. Paolo Romano
Supervisor: Prof. André Ferreira Ferrão Couto e Vasconcelos
Member of the Committee: Prof. Paulo Jorge Fernandes Carreira

September 2020

Acknowledgments

I would like to thank my family for supporting me during this stage, for all encouragement given and caring over all these years. This project would not be possible without this support.

I would also like to acknowledge my supervisor, Prof. André Vasconcelos for all the orientation given and the availability to discuss any problem. I also need to acknowledge Rui Cruz for all the productive discussions and insights given that shaped this work, to him and all MedClick colleagues a big thank you.

I would also like to express my gratitude to Nuno Silva from Grupo Luz Saúde for all the insights that he gave me and sharing of his knowledge that made this possible. Also for supporting to access to Hospital da Luz data, even after all the complications due to the current pandemic, it was a crucial part of this work.

Last but not least, to all my friends and colleagues that helped me throughout this work with times of leisure and fun, which allowed me to overcome the pressure of this work.

To each and every one of you – Thank you.

Abstract

A no-show is when a patient misses a previously scheduled appointment. No-shows cause an impact in the healthcare sector, decreasing their efficiency. When a patient misses an appointment it wastes the clinic resources, postpones his chance to get treated for a medical condition and denies medical service to another patient. Attenuating this problem is the focus of this thesis. Machine Learning is a branch of artificial intelligence that provides techniques to find patterns in data and make predictions. These techniques were used in healthcare data and to make no-show predictions. A no-show prediction model was created to integrate these machine learning techniques into a model that facilitates the testing of these predictions on different datasets. This prediction model was integrated into the MedClick platform to allow the models and predictions made to be saved and integrated into a real-time system. To figure out which machine learning techniques work better and how good the predictions can be, they were tested on three distinct datasets. With these tests, we were able to find that the best features for predicting no-shows were waiting time and distance. The results obtained from the prediction algorithms are still not ideal and a compromise between recall and precision needs to be found.

Keywords

No-show; Healthcare; Prediction Algorithms; Machine Learning; Pre-processing.

Resumo

Um no-show é quando um paciente falha uma consulta previamente marcada. No-shows causam impacto no sector da saúde, diminuindo a sua eficiência. Quando um paciente não comparece a uma consulta, os recursos da clinica são desperdiçados, o paciente atrasa a sua oportunidade de receber tratamento e nega serviço médico a outro paciente. Atenuar este problema é o foco desta tese. Aprendizagem automática é um ramo de inteligência artificial que providência técnicas para encontrar padrões nos dados e fazer previsões. Estas técnicas foram usadas nos dados providenciados por clínicas e hospitais para fazer previsões de no-show. Um modelo para previso de no-shows foi criado para integrar as técnicas de aprendizagem automática num modelo que facilite o teste destas previsões em datasets diferentes. O modelo de previsão foi integrado na plataforma da MedClick para permitir aos modelos criados e previsões feitas serem guardadas e integradas num sistema em tempo real. Para perceber quais técnicas de aprendizagem automática funcionam melhor e quo boas as previsões podem ser, estas foram testadas em três datasets distintos. Com estes testes fomos capazes de descobrir que os melhores atributos para prever no-shows, foram o tempo de espera e a distância. Os resultados obtidos pelo algoritmo de previão ainda não são os ideias, e um compromisso entre as métricas ”recall” e precisão precisa de ser encontrado.

Palavras Chave

No-show; Sector da Saúde; Algoritmos de Previsão; Aprendizagem automática; Pré-processamento.

Contents

List of Figures

List of Tables

Acronyms

ANN	Artificial Neural Network
GB	Gradient Boosting
LASSO	Least Absolute Shrinkage and Selection Operator
LR	Logistic Regression
RF	Random Forest
RFE	Recursive Feature Elimination
RUS	Random Under Sampler
SMOTE	Synthetic Minority Oversampling Technique

1

Introduction

Contents

A no-show is when a patient misses an appointment that was previously scheduled. This phenomenon happens in all sorts of areas, where there is the need to schedule patients or clients into a time slot. In this thesis, we are going to focus on the healthcare area. No-shows cause an impact in every hospital and clinic in the world, attenuating the effects of no-shows in the healthcare area is something that can provide many economic and social benefits.

When a no-show happens there are two consequences, the first happens to the patient who misses the appointment who postpones his chance to be treated for a medical condition. The second one affects the hospital and other patients because there are other patients who could have used that opportunity to be seen by the doctor. This lost opportunity also means a loss of revenue to the clinic and hospital. Given the current high demand for healthcare, wasting this percentage of available resources is unacceptable, as long as there is a list of patients waiting to receive assistance.

To attenuate some of these consequences it is important to figure out what makes patients miss their appointments and, whether or not there are identifiable patterns that allow us to know how likely are patients to miss their appointment.

In order to do this, the appointment data stored by hospitals and clinics around the world can be used. Using this data and combining it with some machine learning techniques we can begin to find some of these patterns and obtain a probability for how likely is a patient to no-show. If these probabilities are high then specific actions can be performed by the hospital, like scheduling another patient for that time slot or contact the patient to try to confirm the appointment.

1.1 Motivation

This thesis was proposed by MedClick and the work will be developed in collaboration with them and Grupo Luz Saúde.

MedClick is a company with the aim of creating a scheduling platform for patients and at the same time helping medical service providers increase the efficiency of their practices.

The work developed in this thesis will all be done in the context of this platform, to help MedClick reach their goals. This platform will have many features but the one this thesis will focus on is on the no-show prediction system. The goal of this system is to predict no-shows and with these predictions help medical service providers attenuate the negative effects of no-shows. The successful implementation of this system will significantly improve the healthcare system in Portugal, helping healthcare providers and patients.

Grupo Luz Saúde is one of the largest private healthcare providers in Portugal, providing care though over 30 hospitals and clinics from all over Portugal. With this collaboration, we were able to develop and apply the algorithms using real world data from an hospital.

This work was important because it allowed us to test the efficiency of machine learning techniques in a real-world problem that causes issues in the healthcare area and many others. Helping solve this no-show problem is something that can provide many economic and social benefits.

1.2 Objectives

The objective of this thesis is to improve and keep developing the no-show prediction system for the MedClick scheduling platform. The goal of this system is to help clinics and hospitals mitigate the negative effects of no-shows.

There are three main features of this system: The first one is to notify the patients of the appointments and to confirm their presence. The second one is a prediction algorithm that uses machine learning techniques and will return the probability of no-show for an appointment. Finally, if the system detects a high probability of no-show, it will automatically try to reschedule another patient to that time slot.

This thesis is focused on the second one by improving and testing different machine learning techniques. This thesis focused on three main objectives which are:

- Create a prediction model that automatically returns the no-show probability for the appointments in the MedClick database, and can efficiently test data from many clinics and hospitals without the need of an advanced user to tweak the system.
- To test many machine learning techniques and find out which provide the best and most consistent results across many datasets.
- To find out which features are more important to obtain better results, which prediction algorithm works best and how accurate can the predictions be in large datasets from real-world clinics and hospitals.

This work is expected to give a solid foundation to allow the continuation of tests in the no-show prediction system and give some answers to what results are possible to obtain from these machine learning methods to tackle the no-show problem.

1.3 Thesis Outline

The thesis was structured in the following way: Chapter 2 is a review of related work, starting with a brief description of what the problem is and what has been found. Then it focuses on the main causes for no-shows, what are the best features for predicting no-shows and comparing the results obtained by the prediction models in previous works. Chapter 3 describes how the prediction algorithms and the

machine learning techniques used work. Chapter 4 describes the previous work done in the context of the MedClick application, focusing on the no-show part. Chapter 5 provides information on how the prediction model created works and how it was adapted into the existing no-show module. Chapter 6 shows the characteristics of the datasets used to test the machine learning techniques and the prediction algorithms. Chapter 7 is where the results achieved by the different machine learning techniques and prediction algorithms are analyzed and discussed. Chapter 8 is the conclusion which summarizes the work done, the conclusions obtained and reveals the main limitations along with suggestions for future work.

2

Related Work

Contents

No-shows are estimated to have a big financial impact on hospitals and clinics [21] and as such many studies can already be found on analysing the impact of no-shows [7], how to best deal with them and using new ways to try to predict them. All of this to reduce the level of impact they have in hospitals and clinics worldwide.

Many of these studies try to pinpoint what are the major causes of a patient no-show [11], whether they are involuntary or not. They also focus on looking at what are the best practices to reduce the impact of no-shows, this is normally done by overbooking [14, 16], but has discussed in this articles this practice may have some impact on waiting time and client satisfaction. Machine learning is a technology that has been emerging and being used in several fields and, as such, some articles studied how to take advantage of this to reduce no-shows. This technology uses the appointments data, in a clinic or hospital, and using an algorithm it tries to find patterns in the data that can then be used to make predictions on no-shows. These predictions can be in form of a probability, which translates to how sure the algorithm is of that outcome. Using these probabilities a clinic or hospital could then decide whether or not it would be better to schedule a patient for that time slot.

In the context of MedClick, there has already been a thesis, made by Daniel Sousa [1], on trying to develop a method that can predict, with accuracy, whether a patient is going to no-show. There is also one thesis by Inês Ferreira [2] on the same subject, particularly on trying to find out which algorithm can make better predictions and on further developing the prediction system.

2.1 Causes of No-Shows

Finding causes for no-shows is a good starting point to check if these causes can be prevented from happening and, whether or not, they can be used as ways of predicting no-shows.

Many studies emphasize finding out what are the causes for a patient to not show to an appointment [7, 11]. Missing an appointment can be a voluntary or an involuntary act, this last one being when the patient did not intend to miss the appointment. There are many reasons for not showing to an appointment these include forgetting the appointment, other competing priorities or conflicts, and the patient's health status.

The most common reason is when a patient forgets the appointment [7], for this, many clinics have already implemented a phone or e-mail reminder, which is reported to reduce no-shows [8, 9, 15]. Other reported reasons for no-shows are the health of the patient which may feel better and not need the appointment anymore, other priorities like a work schedule change or having to take care of another family member can also lead to no-shows. Scheduling problems due to bad quality of the service can also cause problems in cancelling the appointment.

The weather can also be a factor if it is raining or snowing people prefer to stay at home and if the

health problem is not too severe, they can no-show [20]. Financial problems and lack of transportation were also some of the reported reasons.

2.2 Features for No-show Prediction

To be able to predict whether a patient is going to no-show to an appointment, it is required to have access to many factors about the appointment and the patient, which in conjunction leads to a prediction that can be stronger by having access to many factors and many similar cases.

Many studies tackle this problem in an attempt to make their prediction algorithms better [3–5, 10, 12, 13]. So there is already some information to help figure out which features in the appointment data of a clinic or hospital have more relevance to predict a no-show.

These features can be divided into two categories: some are relevant to the patient like gender, age, marital status and insurance status. The others are relevant to the appointment like a day of the scheduled appointment, the amount of time between the day the appointment was scheduled and the actual appointment day and the type of clinic.

The feature found, relevant to the patient, which most articles conclude as having the most predictive power is age, where younger patients seem to no show to more appointments than other age groups [18]. Other features in the patient category also have an impact. These are being unmarried, not having health insurance, the severity of the illness and the scholarship level. The gender of the patient was considered by most articles as having very little impact, in other words, there are no differences between men and women regarding their attendance to previously arranged appointments [3]. In the appointment features, it is found that the waiting time has a large impact [17, 20]. Other characteristics that also have an impact are the hour of the day, whether it is the patient first appointment [19], the chosen medical speciality, the hospital centre, whether it is a weekday or weekend, the type of appointment and the distance to the clinic. Other features are predicted to have an impact, but it is required to use other datasets than the ones that exist in most clinics like, for example, the weather.

Beyond all these features there is another one that has a huge impact in predicting accurately if a patient will no-show, which is the prior no-show history, whether the patient has missed the last scheduled appointments [17, 20]. This feature is the one that has the most predictive power.

To obtain some of these features the datasets provided by the clinics needs to be pre-processed. This allows the use of more specific information that can work as a better predictor, for example, having a feature that tells you if the previous appointment was a no-show and having the hours of the day and the age divided into categories. Creating these new features allows for an improvement of the results to as much as 10% better predictions [5].

2.3 Comparing Articles on No-show Prediction

Before starting this work it is important to look at the works that already exist for this no-show problem, what where the techniques used and the results obtained. The following table shows a comparison between the most important articles found on the no-show prediction that clearly stated which prediction algorithm was used, the features used, and the results. With this table, it is possible to easily see which features are used the most, and compare the results from each prediction algorithm.

It is possible to see, in the table 2.1, that four different algorithms seem to have achieved the best results in their respective articles, these are the Hybrid Model, the Logistic Regression, the Neural Network and the Gradient Boosting Machine. It is hard, based on these results, to say which algorithm performed the best since they were not tested with the same datasets, nor with the same conditions. Also, it is possible to verify that the metrics used to evaluate their performance are not always the same.

In the set of features chosen, we can see four features that are always used these are: Age, Gender, Prior no-show history (different variations of this feature are used), and Date (used to create other features). Some articles use more features than others, this might be due to the information present or lack of it, in each dataset. The datasets might also work better with some features than others which might justify the different choices between articles.

The Hybrid Model was tested against other 12 algorithms and achieved the best results, using the metric Mean Squared error to compare them. These results indicate that this algorithm performs better than most but it was tested in just one dataset with the same set of features, which is not enough to be sure of these results. The Logistic Regression was used in three of these articles, in the first one it achieves a c-statistic of 0.82 which normally means it makes strong predictions, it also performed well in the article [4], coming right after the Hybrid model. On the other hand, in the last article from Inês Ferreira, we can see it is one of the algorithms that performed worse. The Neural network is used in article [6] and achieved an accuracy of 90% which seems really promising, but as already noted the accuracy alone is not a strong metric. Finally, the Gradient Boosting achieved really good results in the last article from Inês Ferreira, but in article [5] it achieved only average results.

With this table, it is possible to see that some promising results were already obtained, but the algorithms were tested with just one dataset and the same features throughout. This is one of the issues this work will tackle by using different datasets, different features and different metrics to try to find out if there is an algorithm that actually has a generally better performance.

No-Show Modeling for Adult Ambulatory Clinics [3]	
Features used	Age, Race, Provider type, Insurance type, Learning site, Last appointment was no-show, Time and date of appointment, New visit, Received telephone reminder, Same day, Overbooked Days of lead time, Days since last visit
Algorithm	Logistic Regression
Results	c-statistic = 0.8236 Data: data obtained from 20 outpatient clinics of a Midwestern hospital. Includes information from 130,819 patients over a period of 3 years.
An integrated decision-support tool to forecast and schedule no-show appointments in healthcare [6]	
Features used	Age, Appointment day, Appointment time, Average Income, Distance, Education level, Events, Father's employment, Mother's employment, Poverty level, Precipitation, Primary payer, Proficiency scores, Provider name, Provider type, Race, Description, Sex, Temperature
Algorithm	Neural Network w/ conjugate gradient as a learning algorithm Decision Tree to discover rules in the Database
Results	Accuracy = 90% Data: children's hospital in Midwestern Ohio with 87,000 appointments
A hybrid prediction model for no-shows and cancellations of outpatient appointments [4]	
Features used	Sex, Marital Status, Insurance, Clinic Type, Age, Distance, Recency, Closeness to non-work days, Attendance record
Algorithm	Chosen Algorithm: Hybrid Model Beyond this algorithm another 12 algorithms were tested, after the hybrid model the 4 algorithms that achieved best results were: Multinomial Logistic Regression, Multinomial Bayesian update, UBM(Universal Background Model) and MLP(Multilayer Percepton Neural Net).
Results	MSE(Mean Squared Error) =0.072 (in the Hybrid Model) The other algorithms obtained an MSE of 0.111, 0.143, 0.197 and 0.286 respectively. Data: Veteran Affairs Medical Center in Detroit (appointments from 10/1/2009 to 2/1/2010 have been used for training and appointments after 2/1/2010 have been considered for testing)
Machine-Learning-Based No Show Prediction in Outpatient Visits [5]	
Features used	Age, Sex, First appointment, Last appointment, Number of appointments, Number of days, Days since last appointment (Speciality, Type of appointment, Consultation, Medical center, Days since last appointment, Appointment time interval, Day of the week, Month, Days since it was requested, Days since first appointment, Attended last appointment) from both the preceding appointment and the appointment we want to predict
Algorithm	Gradient Boosting Machine
Results	Accuracy = 70% Error of 38% in predicting non-attendance Error of 28% in predicting attendances Data: 322,979 appointments from university hospital in Madrid, The San Carlos Clinical Hospital (January 2015 to September 2016)

MedClick: Last Minute Medical Appointments No-Show, Daniel Sousa [1]	
Features used	Age, Appointment date, Sex, Distance, No-show history
Algorithm	Hybrid Model (simpler version)
Results	Accuracy = 70% using (70% for training and 30% for testing) Data: MD Clinica with 608,000 records
MedClick: Last Minute Medical Appointments No-Show Management, Inês Ferreira [2]	
Features used	MD Clinica: Age, Weekday, Postal code, Previous appointments, Previous no-shows, no-show ratio, Gender, Insurance, Physician Brazil Data: Age, Gender, Handicaps, Weekday, Waiting days, Number of previous appointments, Number of previous no-shows, No-show ratio, Number of diseases
Algorithm	Gradient Boosting, Random Forest, K-Nearest Neighbors, Logistic Regression
Results	MD Clinica Data with 10,123 records with 17% no-shows Gradient Boosting: Accuracy = 0.877, Precision = 0.958, Recall = 0.790, F1 Score = 0.865 Random Forest: Accuracy = 0.867, Precision = 0.94, Recall = 0.785, F1 Score = 0.855 K-Nearest Neighbors: Accuracy = 0.644, Precision = 0.725, Recall = 0.462, F1 Score = 0.563 Logistic Regression: Accuracy = 0.634, Precision = 0.72, Recall = 0.456, F1 Score = 0.556 Brazil Data with 110,527 records with 20% no-shows Gradient Boosting: Accuracy = 0.836, Precision = 0.888, Recall = 0.771, F1 Score = 0.825 Random Forest: Accuracy = 0.825, Precision = 0.889, Recall = 0.746, F1 Score = 0.809 K-Nearest Neighbors: Accuracy = 0.615, Precision = 0.626, Recall = 0.569, F1 Score = 0.596 Logistic Regression: Accuracy = 0.633, Precision = 0.642, Recall = 0.612, F1 Score = 0.617

Table 2.1: Comparison of articles that used prediction algorithms

3

Machine Learning

Contents

Machine learning is a branch of artificial intelligence that provides systems with the ability to automatically learn and improve from experience, without being explicitly programmed. The techniques in machine learning use different methods to identify patterns in data and then use these learned patterns to make accurate predictions.

It is already possible to find some studies on using machine learning techniques to predict no-shows [3–6, 12, 13], in these studies it is possible to see different algorithms being tested, in an attempt to get better predictions, in other words making these predictions as accurate as they can be.

3.1 Prediction Algorithms

There are many prediction algorithms and models that can be applied to a wide range of use cases. Determining what prediction algorithms are best for your problem is key to getting better results and leverage the data into making insightful decisions.

In this section, six algorithms are analysed that were shown to make better predictions, which are: Gradient Boosting, Random Forest, Hybrid model that combines two different algorithms, Bayesian Inference, Logistic Regression and Neural Networks.

It will be discussed how these algorithms work, their advantages and disadvantages.

3.1.1 Gradient Boosting

Gradient Boosting (GB) is a machine learning technique, which produces a prediction model in the form of an ensemble of weak prediction models. Boosting in machine learning is a method for creating an ensemble in which the predictors are not made independently, but sequentially. An ensemble is a combination of simple individual methods that work together to create a new model more powerful. Machine learning boosting fits an initial model into the data and afterwards the second model built will focus on predicting the cases where the other model performed worse. Combining these two models is expected to yield better results than any of the other models alone. This process of boosting is repeated many times, with each successive model attempting to solve the mistakes of the previous one. Finally, the models are all joined together into a more powerful version.

Gradient Boosting is a type of machine learning boosting. It assumes that the best model when combined with the previous ones will minimize the prediction error. To improve its predictions, gradient boosting looks at the difference between its current approximation and the known correct target vector, this is called the residual. It then trains a weak model that maps feature vector x to that residual vector. Adding a residual predicted by a weak model to an existing model's approximation pushes the model towards the correct target. Making lots of these pushes improves the overall model's approximation.

Gradient Boosting has shown considerable success in various practical applications. Also, it is extremely flexible and can easily be customized to different practical needs. The biggest drawback of this model is that it is sensible to over-fitting and they have high memory consumption, which can affect the speed of the model's evaluation [24].

3.1.2 Random Forest

Random Forests are an ensemble learning method, used in classification and regression tasks [31]. As previously mentioned in Gradient Boosting, the ensemble is a combination of simple and weak models that together produce a more powerful version. Random Forest does this by creating multiple decision trees and combining them together to obtain a model which outperforms any of the individual decision trees.

Random forest trains each of the decision trees with random subsets of data, and then searches for the best features amongst that subset of data. This causes the decisions trees to have a low correlation between each other, which is key because if the decision trees where all built in the same way, there would be no advantage to this method.

The result of a prediction will be the average of all the individual model's prediction. By doing this each tree protects one another from their individual errors, if some trees are wrong, many others will be right, so together they can move into the right direction.

The model performance can be optimized by increasing the number of trees trained, but this can lead to a slower computation.

3.1.3 Artificial Neural Network

Artificial Neural Network (ANN), are a set of algorithms modelled to try to imitate the human brain. They are designed to recognize patterns in data and are used for a variety of problems, such as classification, prediction, pattern recognition, or optimization [6].

An Artificial Neural Network used in prediction problems is composed of several layers. The layers are made of nodes. A node is a place where the computation happens, loosely patterned on a neuron in the human brain, which fires when it encounters sufficient stimuli. A node combines input from the data with a set of coefficients, or weights that either amplify or dampen that input, based on its significance in trying to make the prediction. Afterwards, the inputs and weight products are summed and then passed through an activation function, which determines if that signal should progress further through the network and affect the outcome. If it does then that neuron has been "activated".

A node layer is a row of these neuron switches that turn on and off as the input is fed through the net. The output layer is then a result of these subsequent layers, starting from an input layer with your

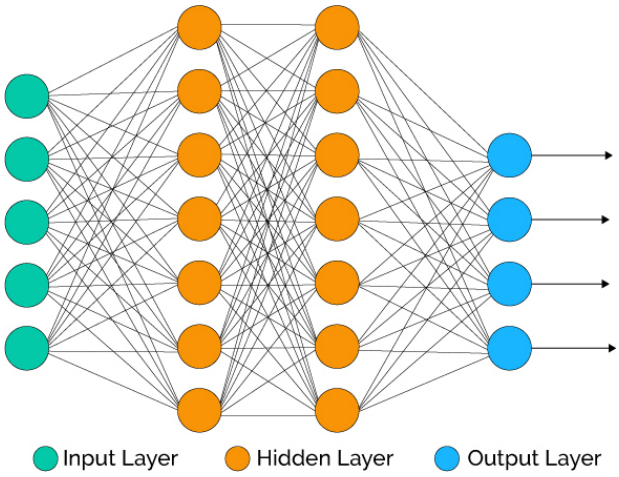

Figure 3.1: Multi-layer perceptron neural network.

initial data to a hidden layer where the computations are done 3.4.

Where:

- **Input layer** contains the Artificial Neurons which receive initial data for the neural network.
- **Output layer** contains units that produce the result for the given inputs.
- **Hidden layer** is the intermediate layer between the input and output layer and the place where all the computation is done. The job of the hidden layer is to transform the input into something that output unit can use in some way.

3.1.4 Logistic Regression

Logistic Regression (LR) is a method of regression analysis to estimate the parameters of a logistic model. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

There are different ways of using the logistic regression there is, for example, Multinomial Logistic Regression, which deals with situations where the response variable can have three or more possible values.

The central mathematical concept that underlies Logistic Regression is the Logit function [22], which is used in Logistic Regression like the equation described below 3.1.

$$P(x_1, ..., x_k) = \frac{1}{1 + e^{-}(\beta_0 + \beta_1 x_1 + ... + \beta_k x_k)} \tag{3.1}$$

In the equation, above:

- k represents the total number of features used in the model;
- x_k represents a respective feature;
- β_0 is the intercept term;
- β_k is a weight associated to its respective feature;

The output of this function is always between 1 and 0, which can be translated into the probability of that event occurring. The algorithm has two important phases: The first one is the model building, where the training data is used to estimate the best coefficients to solve the given problem. This phase also builds a cost function that quantifies the error by comparing the predicted probability with the real value. Then, using gradient descent, a global minimum can be found of that function and get better coefficients for that problem. The second phase is the prediction phase, which will use the estimated coefficients and the logit function to make a prediction, using a set of features. This will return the probability for the prediction.

Logistic Regression is widely used in solving problems similar to no-shows since it can make predictions very fast and it is easy to add and remove features, you only need to rebuild the model. The biggest disadvantages of this model are that it requires a large dataset to be precise, and the features need to have been previously studied since they can greatly affect the precision of the model.

3.1.5 Bayesian Inference

Bayesian Inference is a method of deducing properties about a population or probability distribution from data using Bayes' theorem (equation 3.9). This theorem is useful because it allows us to use some knowledge or belief that already exits have to help calculate the probability of a related event.

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{3.2}$$

Where:

- **P(H | E)** corresponds to the posterior probability;
- **P(E | H)** is the probability of observing E given H, it is called likelihood;
- **P(H)** is the prior probability;
- **P(E)** is the evidence;

The Bayesian Inference main goal is to provide a method for incorporating our prior beliefs, having evidence in hand, in order to produce an updated posterior belief. What makes it a valuable method is that posterior beliefs can then be used as prior beliefs, under new data, allowing us to continually adjust our beliefs by repeatedly applying Bayes 'rule.

This method was used in the hybrid model described in chapter **??** and in Daniel Sousa thesis [1] with a slight adaptation, which was the following:

$$E(a_k \mid y_1, y_2, ..., y_k) = \frac{y_k + a_k}{\sum_{k=1}^{k} y_k + \sum_{k=1}^{k} a_k} \tag{3.3}$$

Where:

- k corresponds to the posterior probability;
- a_k is the probability of observing E given H, it is called likelihood;
- y_k is the prior probability;

The advantages of the Bayesian Inference model is that it is fairly easy to compute when using a few features. The main disadvantage is that it requires us to perform integration over variables, and many of these computations are analytically intractable. As a result, much contemporary research in Bayesian approaches to machine learning relies on or is directly concerned with, approximation techniques [23].

3.1.6 Hybrid Model

A hybrid model is a combination of logistic regression, as a population-based method, with Bayesian inference, as an individual base method. Logistic regression is used to build a model that computes the initial estimation of the no-show probability for each patient, based on the general behaviour of the population. Afterwards, it uses Bayesian Inference to adapt the initial probability obtained with each patient, using their appointment records.

A complex version of this algorithm was created and tested in article [4].

3.1.7 Models Comparison

This table 3.1 provides a comparison of the prediction algorithms described above. The goal of this table is to provide an easy way to compare the strengths and weaknesses of each algorithm.

Algorithm	Strenghts	Weaknesses
Gradient Boosting	-Strong predictive accuracy -Not much data pre-processing required	-More sensitive to overfitting if the data is noisy -Not easy to understand predictions -High flexibility means lots of parameters need to be fine tuned
Random Forest	-Tends to result in high quality models -Fast to train	-Can be slow to output predictions -Not easy to understand predictions
Logistic Regression	-Highly interpretable -Does not require any tuning -Easy to implement and very efficient to train	-Can be outperformed by more complex methods -Needs to have the most important features identified before making strong predictions
Bayesian Inference	-Highly interpretable	-Computationally expensive with to many features -Needs a good prior distribution to be effective
Hybrid Model	-Comprises two different algorithms to improve the predictions	-Hard to implement -Features need to be divided to be used at different stages of the algorithm
Artificial Neural Network	-Outperforms other models on large datasets -Ability to learn and model non-linear and complex relationships -It can infer unseen relationships on unseen data	-Very slow to train -Requires large amounts of data to be effective -Almost impossible to understand predictions

Table 3.1: Comparison of prediction algorithms

3.2 Pre-Processing

Data pre-processing is a fundamental step in Machine Learning since the capability of our model to learn will depend on the quality of the data and the information that can be extracted out of it. Pre-processing refers to the transformations applied to our data before it is fed into the algorithm. The data needs to be formatted properly, to be applied to some Machine Learning models. Some examples of these are null values and categorical values which are not accepted in most prediction algorithms. Other techniques like normalization, data balancing and feature selection try to improve the results of the applied model by making the data cleaner and easier to learn from. In this chapter, some of these techniques are discussed.

3.2.1 Normalization

Normalization is a technique used in data pre-processing for machine learning. The goal of normalization is to change the values of numeric columns in a dataset to a common scale, without distorting

the differences in the range of values. The reason for applying normalization is that variables that are measured at different scales do not contribute equally to the model fitting and might end up creating bias. For example, if some feature in the dataset has values in the range of 0 to 10 and another feature has values in the range of 0 to 10000, then the result of prediction will be much more influenced by the second feature, then the first, even if the first one is more important.

By using normalization we counter this problem and prevent the outcome to be wrongly influenced by a feature with large numeric values.

The type of normalization used is MinMax Scaling and the formula is:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.4}$$

Where:

- X_{min} corresponds to the minimum value of that feature;
- X_{max} corresponds to the maximum value of that feature;

3.2.2 One-Hot Encoding

Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric. This means that categorical data must be converted to a numerical form. One-Hot encoding is one solution to this problem where each categorical value will be transformed into a new feature in the dataset of 1 and 0.

There are other solutions to this problem that transform each categorical value into a different number, the problem to this approach is that the numbers will have an order and the algorithm might give more importance to higher values or make calculations using those values and reach wrong conclusions. One-Hot encoding is a solution to this problem but has the disadvantage of increasing the size of the dataset which can become very computationally expensive, depending on the size of the dataset and the number of distinct categorical values.

3.3 Feature Selection Techniques

Most datasets used to obtain predictions have a high dimension, which means they have many features. This may lead to slow computation times and also to overfitting. Overfitting is when the model has high variance and low bias on the training set which leads to poor generalization on new testing data.

Feature selection techniques are used to counter this problem since not all features used in a dataset contribute to the prediction variable. These techniques remove the features that are not important for the prediction. By doing this we can improve accuracy and reduce the complexity of the model, which in

turn, reduces the time it takes to train a model.

In this chapter, we will discuss some of the techniques that were tested and their advantages and disadvantages.

3.3.1 Recursive Feature Elimination

Recursive Feature Elimination (RFE) is a technique that selects features by recursively considering smaller and smaller sets of features. This technique uses an external estimator, which can be logistic regression or random forests, to assign weights to features. First, the estimator will be trained on that set of features and the feature importance of each one will be obtained through a coefficient.

Afterwards, the least important features are pruned from the current set of features. This procedure is recursively repeated on the pruned set until the desired number of features is reached. This technique can be used with cross-validation to make sure the selected features are the best.

3.3.2 Boruta

The Boruta algorithm is a wrapper built around the random forest classification algorithm. It is a feature selection technique to choose the most important features in a dataset.

Boruta starts by duplicating the dataset and shuffles the values in each column to remove their correlation with the target variable. These values are called shadow features. Then it combines the original ones with the shuffled copies and trains a Random Forest Classifier on the dataset to obtain a score for the importance of each feature.

Afterwards, a Z score is computed, which is the mean of accuracy loss divided by the standard deviation of accuracy loss. The maximum Z score among the shadow features is found and it will start tagging the variables as unimportant, if they have a significantly lower value than the maximum, or has important if they have a significantly higher value than the maximum Z score. This process is then repeated for a predefined number of iterations, or until all the attributes are tagged as either unimportant or important.

3.3.3 LASSO

Least Absolute Shrinkage and Selection Operator (LASSO) is a method that performs two main tasks, regularization and feature selection.

The LASSO method works by putting a constraint on the sum of the absolute values of the model parameters, where the sum has to be less than a fixed value (upper bound). To do this a shrinking (regularization) process is applied that penalizes the coefficients of the regression variables, shrinking some to zero. During the feature selection process the variables that have a non-zero coefficient, after

the shrinking process, are selected to be part of the model. The goal of this process is to minimize the prediction error.

There is an important tuning parameter () that controls the strength of the penalty. The larger the parameter, the more number of coefficients will be shrunk to zero.

3.3.4 Comparison of Feature Selection Methods

The following table 3.2 provides a comparison of the feature selection techniques described above. The goal of this table is to provide an easy way to compare the strengths and weaknesses of each algorithm.

Algorithm	Advantages	Disadvantages
RFE	-For more complicated tasks it can significantly reduce memory consumption and speed up calculations	-Computational expensive when used together with cross validation
Boruta	-More precise and stable than feature importance values from single random forests	-May be quite long as several iterations need to be performed until the algorithm converges
LASSO	-Computationally efficient	-Might have worse performance than alternative methods -LASSO will only select one feature from a group of correlated features

Table 3.2: Comparison of feature selection methods

3.4 Sampling Techniques

Sampling techniques are used when there are imbalanced datasets, where there is a severe class imbalance. This is the case of the datasets used to predict no-shows because there are many more appointments with shows than no-shows. The challenge of working with imbalanced datasets is that most machine learning techniques tend to ignore the minority class, which is normally the most important to have good predictions on.

Sampling techniques help to solve this problem by balancing the data, either by increasing the minority class or by decreasing the majority class.

3.4.1 SMOTE

One way to solve the problem of imbalanced datasets is to oversample the minority class. This can be done by simply duplicating examples from the minority class. This can balance the class distribution but does not provide any additional information to the model.

Synthetic Minority Oversampling Technique (SMOTE) is an improvement on duplicating values by synthesizing new examples from the minority class. SMOTE works by selecting examples that are close

in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line.

Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbours for that example is found. A randomly selected neighbour is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.

This procedure can be used to create as many synthetic examples for the minority class as are required.

This approach is effective because new synthetic examples from the minority class are created that are relatively close in feature space to existing examples from the minority class.

A general downside of the approach is that synthetic examples are created without considering the majority class, possibly resulting in ambiguous examples if there is a strong overlap for the classes.

3.4.2 Random Under Sampler

Another way to solve the problem of imbalanced datasets is to under-sample the majority class. Random Under Sampler (RUS) is a technique that does just that. It removes random samples from the majority class until the class distribution is balanced. This technique can work in balancing the data, but it removes what might be important data from the dataset, it should only be used if you have a large amount of data and can afford the luxury of "slimming down".

Another solution that can be used is first using random undersampling to trim the number of examples in the majority class, and then use SMOTE to oversample the minority class to balance the class distribution. This way we can help mitigate the disadvantages of both strategies.

3.4.3 SMOTE with Tomek Links Undersampling

Tomek Links is an undersampling technique that identifies pairs of nearest neighbours near the border of two different classes and removes one or both of the samples in these pairs (such as the examples in the majority class). This affects the decision boundary of the training dataset making it less noisy and ambiguous.

The combination with SMOTE works by first applying the SMOTE method to oversample the minority class to balance the distribution, then examples in Tomek Links from the majority classes are identified and removed. This combination was shown to provide a reduction in false negatives at the cost of an increase in false positives for a binary classification task [28].

3.4.4 SMOTE with Edited Nearest Neighbours Undersampling

Edited Nearest Neighbours is another undersampling method and involves using k Nearest Neighbours to locate those samples in a dataset that are misclassified by its neighbours and then removes them. This can be applied to the whole dataset or just the majority class.

ENN is more aggressive at downsampling the majority class than Tomek Links and provides more in-depth cleaning [29]. The combination with SMOTE works the same way by first applying SMOTE to oversample and then using ENN to clean the dataset.

3.5 Evaluation Methods

Evaluation methods are the methods that are used to evaluate how well your model performs, after it has been trained by a prediction model and, as such, it is an essential part of any project. There are methods like cross-validation which train the data and simultaneously test it several times to obtain the performance of the prediction model. This performance will be an average score of each metric in all the iterations performed by the cross-validation. There are many metrics some of these might be accuracy or precision. The choice of these metrics depends on the problem at hand since some metrics might be more important in some cases than others.

3.5.1 Cross Validation

Cross-validation is a widely used technique to estimate how accurately a predictive model will perform. This method prevents overfitting and ensures that the model has got most of the patterns from the data correct, and it is not picking up too much noise. There are different types of Cross-Validation Techniques but the overall concept remains the same.

- To partition the data into a number of subsets.
- Hold out a set at a time and train the model on remaining set.
- Test model on hold out set.

One of the most common techniques and the one used in this project is k-Fold Cross-validation with the stratified variation.

K-Fold Cross Validation works by dividing the data into k subsets, one of which will be used for testing the model, and the remaining k-1 subsets will be used for training the model. Then, the process will be repeated k times so that each data point gets to be used once as a test set. This significantly reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in the test set. Interchanging the training and test sets also adds to the effectiveness of this method. As a general rule and empirical evidence, K = 5 or 10 is generally

preferred, but nothing's fixed and it can take any value. The error estimation is averaged over all k trials to get total effectiveness of our model.

There is another variation of this technique called Stratified K-Fold Cross-Validation. This variation works well on imbalanced datasets. The reason for this is that the splitting of the data into folds is governed by criteria, which ensures that each fold contains approximately the same percentage of samples of each target class as the complete set.

3.5.2 Metrics

Several metrics can be used to check how well a model is performing. the choice of these metrics is an important factor in evaluating how well our model is performing to a given problem. There might be cases where it is important to be really precise to avoid a wrong diagnosis, others where we can accept more mistakes because the benefits of having it right outweigh the drawbacks of having it wrong. This must all be taken into consideration when opting for a model over another one.

3.5.2.A Confusion Matrix

Confusion Matrix is a matrix that describes the complete performance of the model. It shows how many classes were correctly classified, and how many were not.

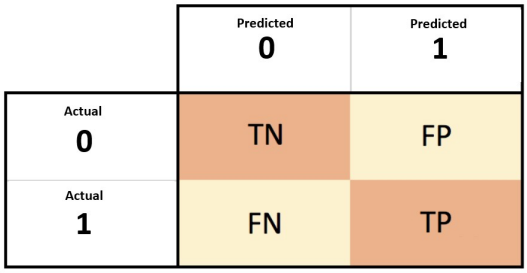

Figure 3.2: Confusion matrix for binary classification.

Where:

- True Positives (TP): Number of instances that were correctly assigned to the positive class.
- True Negatives (TN): Number of instances that were correctly assigned to the negative class.
- False Positives (FP): Number of instances that were incorrectly assigned to the positive class.
- False Negatives (FN): Number of instances that were incorrectly assigned to the negative class.

3.5.2.B Accuracy

Accuracy is the fraction of correct predictions made by the model. It is a good measure when there is an equal number of samples belonging to each class. However, in case of imbalanced datasets, for

example, considering a dataset that has 95% of samples of class A and 5% of class B, it is easy to obtain an accuracy of 95% simple by creating a classification model that predicts all classes as A. For this reason, accuracy alone is not a good measure when dealing with these types of datasets.

$$Accuracy = \frac{NumberofCorrectPredictions}{TotalNumberofPredictions} \tag{3.5}$$

3.5.2.C Recall

Recall it can be called sensitivity or True Positive Rate as well. Recall corresponds to the proportion of positive data points that are correctly considered as positive, among all actually positive data points. This metric is good for imbalanced datasets since it focuses on the positive values, which typically belong to the minority class.

$$Recall = \frac{TP}{FN + TP} \tag{3.6}$$

3.5.2.D Specificity

Specificity or False Positive Rate corresponds to the proportion of negative data points that are mistakenly considered as positive, among all negative data points.

$$Specificity = \frac{FP}{FP + TN} \tag{3.7}$$

3.5.2.E Precision

Precision corresponds to the number of correct positive results divided by the number of positive results predicted by the classifier. Like recall, this metric is also good for imbalanced datasets since it focuses on the positive values, which typically belong to the minority class.

$$Precision = \frac{TP}{FP + TP} \tag{3.8}$$

3.5.2.F ROC-AUC

ROC-AUC is one of the most widely used metrics for evaluation and is used for binary classification problems. AUC means Area Under the ROC Curve and measures the area underneath the entire ROC curve. ROC stands for Receiver Operating Characteristic and it is a graph that plots False Positive Rate against True Positive Rate. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. AUC ranges from 0 to 1 and the greater the value, the better is the performance of the model.

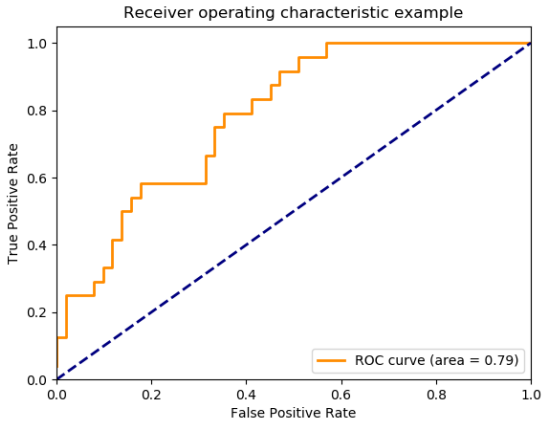

Figure 3.3: Example of a ROC graph.

3.5.2.G F1-score

F1-score is the Harmonic Mean between precision and recall. It represents how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances). High precision but lower recall means that the model is very accurate on identifying positive values, but at the same time classifies many positive instances as negative. On the other hand, when it has low precision and high recall, it means the model identifies correctly many of the positive instances, but at the same time classifies several Negative instances as positive.

The greater the F1-score, the better is the performance of our model. Mathematically, it can be expressed as:

$$F1 - score = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{3.9}$$

F1-score tries to find the balance between precision and recall.

3.5.2.H Precision-Recall-AUC

Precision-Recall-AUC is the area under the precision-recall curve. It is a graph that plots the precision against the recall for different probability thresholds. A skilful model is represented by a curve that bows towards a coordinate of (1,1). A no-skill classifier will be a horizontal line on the plot with a precision that is proportional to the number of positive examples in the dataset. For a balanced dataset, this will be 0.5.

The focus of the precision-recall curve on the minority class makes it an effective diagnostic for imbalanced binary classification models. Precision-recall curves (PR curves) are recommended for imbalanced datasets where ROC curves may provide an excessively optimistic view of the performance.

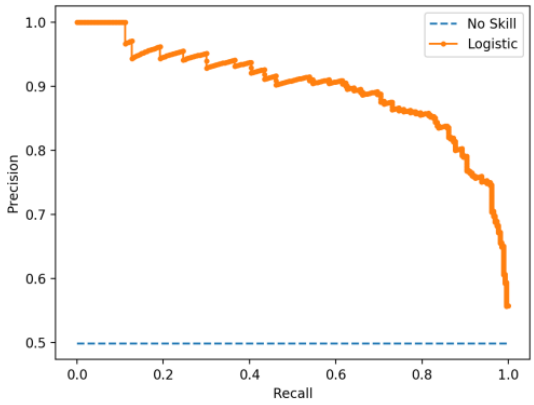

Figure 3.4: Example of a Precision-Recall graph.

3.5.3 LIME

LIME provides local model interpretability. It is a technique that tries to understand a prediction model by modifying a single data sample and tweaking the feature values to understand how the predictions change. This is also what most people are interested in observing to be able to answer why was the prediction made and what features caused the prediction. Other model interpretability techniques can only answer this from the perspective of the entire dataset. For example, feature importance explains which features are important for the whole dataset. It is possible to verify whether the model is overfitting to noise but it will be hard to diagnose why a specific prediction was made.

LIME solves this by providing a list of explanations, reflecting the contribution of each feature to the prediction of a data sample. This provides local interpretability, and it also allows us to determine which feature changes will have the most impact on the prediction.

An explanation is obtained by approximating the underlying model locally by an interpretable one. Interpretable models are linear models with strong regularisation like decision trees. These models are trained on small perturbations of the original instance and should only provide a good local approximation.

An example of an explanation applied to the no-show problem can be seen in figure 3.5, where the red means that feature contributes to the prediction being a show and the green to a no-show.

Lime is model-agnostic, which means it can be applied to any machine learning model.

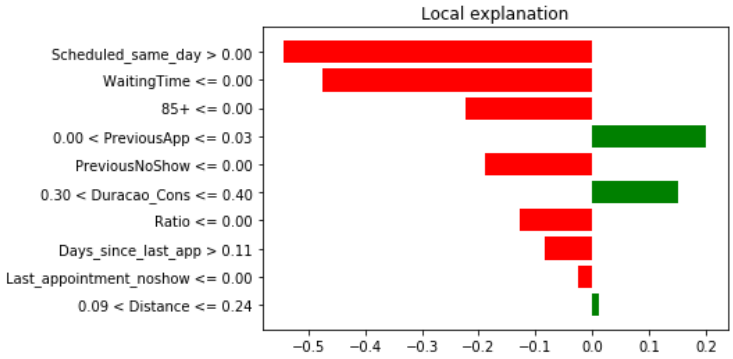

Figure 3.5: Example of an explanation obtained for an appointment in the MD Clínica dataset.

4

MedClick

Contents

This thesis is done in partnership with MedClick, which is a start-up that is creating a platform for hospitals and clinics that allows patients, to easily book appointments, choosing the clinic and medic most suitable for their case. In the case of the no-shows, the platform will have a functional no-show prediction algorithm that will allow clinics, to predict if a patient will miss an appointment and in the case, he is likely to miss, then the platform will automatically try to replace the patients with another one that has less likelihood of no-showing.

In MedClick there is already some work developed on this subject of no-shows. The first work is a thesis by Daniel Sousa which focused on developing the algorithm to predict the no-shows and creating a model to replace patients that have a higher chance of not showing. The last work is another thesis by Inês Ferreira, which focuses on testing other prediction algorithms and also updated how the platform behaved.

4.1 Daniel Sousa's Thesis

The thesis that was done by Daniel Sousa, [1] was mainly focused on developing a no-show prediction algorithm and a model for replacing patients.

The main objective of the model created for replacing the patients is to find patients interested in filling the "last-minute" vacancy slots. It starts with receiving a no-show notification, this can happen in three different situations: when the patient cancels the appointment, when it fails to respond to the appointment confirmation, or when it detects the patient will not arrive on time based on his location. Afterwards, a list is obtained with the patients that have an appointment that day or are close to the health centre, computing their respective no-show probability. Then it will loop until a candidate is found, sending notifications to the candidate with the lowest probability of no-show, giving him a short window of time to answer before skipping to the next candidate. If a candidate answered yes it replaces it, otherwise, it notifies the health centre it did not find a replacement.

The model used to make the predictions and obtain the probability of no-show was the Hybrid Model. This solution is a simple version of the one developed in article [4], using both population and individual approaches.

This model was tested with a dataset from MD Clínica. The features chosen to train this model were the age and sex of the patient, the appointment date and distance to the healthcare centre. First, the logistic regression part (population approach) was tested alone, and it achieved an accuracy of 70%. Only the accuracy was shown as a metric which, and since this dataset is unbalanced it is not a reliable metric. The patient history was then added using the last 10 appointments of each patient, and using the Bayesian Inference equation 3.9, a probability for a no-show for each of the appointments was calculated. The results achieved while adding the last appointments were not the best since this method

was not able to predict with much accuracy if the patient was going to miss his next appointment. By doing this a lot of emphases was given to the patient no-show history, making the other features not have as much impact. This is not the best approach because human behaviour is unpredictable and many other factors need to be taken into account.

4.2 Inês Ferreira's Thesis

The thesis done by Inês Ferreira [2,27], was focused on improving the models for detecting and replacing patients and in testing different prediction algorithms.

The work started by creating a model to detect no-shows for the prediction system. This model increases or decreases the probability of no-show, depending on if the patient answered or not to the appointment confirmation. In case of a negative answer, or if the probability of no-show to the appointment is higher than a defined threshold for that clinic, three days before the appointment, it will automatically try to find another patient interested in scheduling an appointment for that time slot.

The model for replacing patients developed by Daniel Sousa [1] was also improved in this thesis. The main improvements were: In the initial phase of the algorithm, instead of starting to find a replacement only when the system receives a cancellation notification, it also starts when it detects that a patient has a very high probability of no-show. The lists obtained were also altered, one is patients that added themselves to the waiting list for that day and another is the patients that are scheduled for an appointment later in the day. The computed no-show probability is now done by using the model to detect no-shows. Beyond these two new improvements, some features were added: the first one is to notify n candidates instead of just one and having to wait for the patient to confirm before going to the next one, this n can be altered by the clinics themselves. The second is to consider the answers of the patients in the second list so that, in the future, the algorithm has that into consideration and allows other patients to have a chance.

After this part, the emphasis of the thesis is on testing different prediction algorithms and comparing the results with the ones achieved by Daniel Sousa.

The prediction algorithms tested were Gradient Boosting Machine, Logistic Regression, Random Forest and k-Nearest Neighbours. These algorithms were tested with two databases one provided by MedClick, which is from MD Clínica and another obtained online that represents data from medical appointments in Brazil. Before testing the algorithms different pre-processing techniques were used, beyond handling the datasets and removing missing values a technique for balancing the datasets was used. The technique chosen was SMOTE and then the data was split into training and testing using k-Fold Cross-Validation with k = 10.

The results obtained and the features used are shown in more detail in table 2.1. The algorithms

that performed better were Gradient Boosting and Random Forest with the first one performing slightly better. An analysis of feature importance was also done for each dataset. In MD Clínica dataset the most important feature is gender but all features got similar values. On the opposite hand, the data from Brazil has two features that have much more important than the others, these are waiting time and age. In this case, it is already possible to see how much feature importance can change in different datasets and different locations, though, it is also important to note that this MD Clínica dataset had a much smaller quantity of records available than the Brazil data, which might have had an impact on this analysis.

This is the basis of the work developed in this thesis, there was also some work developed in incorporating the prediction system in the MedClick platform that is not discussed here. The work that will be developed with MedClick is a continuation of the work developed in these two theses.

4.3 Limitations

A lot of work has been done in these two theses, but there is still much to be done to improve the prediction system.

In Daniel Sousa thesis there was a limitation in the model for replacing patients, where he only focused on replacing patients at the last minute, this was solved in Inês Ferreira thesis, by using the no-show's predictions to replace the patients that have a high probability of not showing before the day of the appointment. The prediction algorithm developed by Daniel Sousa was also not thoroughly tested, only four features were used and only the accuracy was used to describe the results. It is required many more tests to validate any prediction algorithm.

In Inês Ferreira thesis there are also some limitations. The model developed for detecting no-shows cannot be tested yet, only at further development stages of the scheduling platform. The prediction algorithms developed and tested by Inês Ferreira seemed to yield great results, but one mistake was done because the SMOTE technique was used in the test data at every split made by the cross-validation. By doing this you are generating fake samples and then getting results on those, which does not translate to the real world. The only way to validate these results would be to use the trained model on a dataset that has not yet been seen by the training model and no sampling technique was used on the test data.

There are still other limitations found, one is in the fact that most datasets are different from each other with different features. This requires the code for pre-processing the datasets to have to be changed each time the prediction algorithm is trained. This is not scalable to the needs of the prediction system that MedClick wants to develop, as it will have to work with many clinics and hospitals from all over Portugal.

Prediction algorithms can also work better with some datasets than others, being able to know how to

get the best no-show predictions out of any dataset is really important for the scalability of the prediction system and the main focus of the solution that will be developed.

5

Prediction Model

Contents

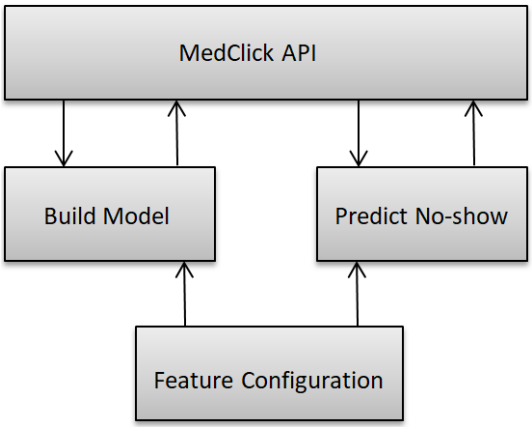

Figure 5.1: Prediction Model components.

This Prediction Model was created to automate the process of constructing prediction models and obtaining predictions from different datasets, without requiring an advanced user to tweak the model every time we are faced with a different dataset. Another advantage of this model is that the prediction model is running in short periods of time, this means that any new appointments that arrive on the MedClick API will be given a no-show probability value on a short period of time.

The prediction model has two main models and one model that supports the other two and provides a configuration file for the user. The first main model is a Build Model that, as the name suggests, builds the model from the given data and saves the model in a file. The name of the file and the final set of features that were used to save the model is stored in the MedClick API.

The second model is Predict Model. This model receives appointments to predict, from a CSV or the API, it then loads the model and the features that the model was constructed on, using the same set of features and the loaded model it returns for each appointment the respective probability of no-show.

Feature configuration is a supporting file that the user must configure to build the model. This supporting file is basically a set of variables that must be assigned a value corresponding to the name of the feature in the original dataset. This is done so that the Build Model can automatically know which feature it corresponds to, and then pre-process it into a new feature that can be used by the prediction algorithm.

5.1 Feature Configuration

Feature configuration is a support file to be used in the build model and predict model phases. The goal of this configuration file is to make it easier to use any type of dataset, without having to change the way it was programmed every time a different dataset is used. This configuration file is basically a simple python file with a set of variables, that need to be filled with the names of the features of the specific

dataset we are using. These variables are then used by the Build Model to automatically pre-process the dataset features into the ones that will be used to train the dataset.

This list of variables contains one to specify whether the data comes from the MedClick API or a CSV file. It has also two variables that are specific to the CSV file, where the path to the CSV should be placed. The remaining features must be matched with the corresponding feature name of the dataset that will be used. A table with the variables used in the configuration file, along with a short description, can be seen in the following table 5.1.

Variable	**Description**
TypeOfInput	Should be filled with 'csv' if dataset is from a CSV or 'json' if it comes from MedClick API.
CSV_Path	The path to the CSV file with the appointments dataset.
CSV_Path2	The path to the CSV file with the patients dataset. Should be left empty in case the patients are already merged with the appointments.
ID	The name of the patient id feature in the dataset used.
Age	The name of the age feature in the dataset used.
Birthdate	The name of the birthdate feature in the dataset used. Only required if there is no age feature.
Gender	The name of the gender feature in the dataset used.
District	The name of the district feature in the dataset used.
Postal_Code	The name of the postal code feature in the dataset used.
Insurance	The name of the insurance feature in the dataset used.
Specialty	The name of the specialty feature in the dataset used.
ID_Medic	The name of the medic id feature in the dataset used.
Appointment _Date	The name of the appointment date feature in the dataset used.
Appointment _Hour	The name of the appointment hour feature in the dataset used. Only required if the hour comes separated from the appointment date.
Scheduled _Date	The name of the scheduled date feature in the dataset used.
NoShow	The name of the no-show feature in the dataset used.
NoShow PositiveValues	The value used to signal a no-show in the no-show feature, for example, 'Yes'.
Other _Features	The remaining features that will not be used to test the dataset.

Table 5.1: Variables in the feature configuration file.

The variables used in the configuration file match the most common features present in most health-care provider's datasets and to the features that possess stronger predictive power. Other features that prove to have a strong predictive power can be added in a future stage so that they can be easily pre-processed.

A dataset does not need to have correspondence to all the variables that are in the configuration file, variables that do not have a correspondent feature should be left empty. This way any dataset, as long

as it has the essential features, can be tested. The variables that need to be filled for the dataset to be tested are patient id, appointment date and no-show. Of course, the more features that are filled, the more information the prediction algorithms will have to learn from, which will lead to better predictions.

In some of the tested datasets, some features were used that were not in the configuration file. This was done because these features were unique to that dataset and they were already binary, so no pre-processing was required. To use these features they should not be placed in the variable Other_features. A dataset where this happened was Brazil dataset which had a lot of binary features that were not in the other datasets, an example of the configuration file used can be seen in figure 5.2.

The pre-processing done will not look at every possible scenario of input, as this would be almost impossible to do. This means that some features will be required to have a certain format to work. One of these cases is postal code, which requires the values to come in the format -. Other features are more intuitive, age will require numeric values and appointment date will require a date, as expected. If the datasets do not have these formats they should be manually pre-processed beforehand.

Examples of the features configuration files used for the other datasets can be seen in figure 5.3 and figure 5.4.

For the case where the dataset comes from the MedClick API, there is also a feature configuration file, that can be seen in figure 5.5. This feature configuration only needs to be updated if there are changes in the MedClick API that add or remove features. The MedClick API data will still require an update since many important features are missing like scheduling date. In later stages, these configuration files are not going to be required and the pre-processing should be optimized to the MedClick API, since here is where all the data will come from.

The main advantage of the feature configuration is that it permits many datasets to be worked without the constant need to change things in the main code. This is great for someone who is not familiar with the code to be able to easily test the prediction model on a dataset. Also in these early stages, Medclick will work with many datasets from different healthcare providers, which makes a configuration file like this essential.

```
'''
Match these variables to the ones used in the dataset
if dataset has no match to the variable put ''
'''

##Json or csv?
TypeOfInput = 'json'
CSV_path = 'BrazilianKaggle.csv'
CSV_path2 = ''#if you need to merge appointments and patients #use this one has the patients

ID = 'PatientId' ##Required
Age = 'Age'
Birthdate = '' #if there is no age feature
Gender = 'Gender'
District = 'Neighbourhood'
Postal_Code = ''
Insurance = ''
Specialty = ''
ID_Medic = ''
Appointment_Date = 'AppointmentDay' ##Required
Appointment_Hour = '' #if the hour is in a separate feature
Scheduled_Date = 'ScheduledDay'
NoShow = 'No-show' ##Required
NoShowPositiveValue = 'Yes' #value used to signal a no-show #if it is 1 leave empty ('')

Other_Features = ['AppointmentID']
#list of remaining features that will not be used
```

Figure 5.2: Feature configuration file used for Brazil dataset.

```
'''
Match these variables to the ones used in the dataset
if dataset has no match to the variable put ''
'''

##Json or csv?
TypeOfInput = 'csv'
CSV_path = 'MDclinica_2019_finalCut.csv'
CSV_path2 = 'MD_patients_2.0.csv' #if you need to merge appointments and patients #use this one has the patients

ID = 'patient_id' ##Required
Age = ''
Birthdate = 'nascimento' #if there is no age feature
Gender = 'sexo'
District = ''
Postal_Code = 'Codigo_Postal'
Insurance = 'convencao'
Specialty = ''
ID_Medic = 'Medico'
Appointment_Date = 'Data' ##Required
Appointment_Hour = 'Hora_Consulta' #if the hour is in a separate feature
Scheduled_Date = 'Data_Marcacao'
NoShow = 'NoShow' ##Required
NoShowPositiveValue = '' #value used to signal a no-show #if it is 1 leave empty ('')

Other_Features = ['freguesia', 'Ultima_Data_Alteracao', 'horachg', 'horaentr', 'estado_consulta', 'est_civil', 'pr
#list of remaining features that will not be used
```

Figure 5.3: Feature configuration file used for the dataset from MD Clínica.

```
'''
Match these variables to the ones used in the dataset
if dataset has no match to the variable put ''
'''

##Json or csv?
TypeOfInput = 'csv'
CSV_path = 'Agendamentos.csv'
CSV_path2 = '' #if you need to merge appointments and patients #use this one has the patients

ID = 'ID' ##Required
Age = 'Idade'
Birthdate = '' #if there is no age feature
Gender = 'Genero'
District = 'Distrito'
Postal_Code = 'Codigo Postal'
Insurance = 'Entidades Financiadoras'
Specialty = 'Codigo do Ato'
ID_Medic = 'Staff ID'
Appointment_Date = 'Data e Hora de Agendamento' ##Required
Appointment_Hour = '' #if the hour is in a separate feature
Scheduled_Date = 'Data e Hora de Marcacao'
NoShow = 'No Show (S/N)' ##Required
NoShowPositiveValue = 'S' #value used to signal a no-show #if it is 1 leave empty ('')

Other_Features = ['Descricao do Ato', 'Concelho']
#list of remaining features that will not be used
```

Figure 5.4: Feature configuration file used for the dataset from Hospital da Luz.

```
'''
Match these variables to the ones used in the dataset
if dataset has no match to the variable put ''
'''

##Json or csv?
TypeOfInput = 'json'
CSV_path = ''
CSV_path2 = '' #if you need to merge appointments and patients #use this one has the patients

ID = 'patientId'##Required
Age = ''
Birthdate = 'birthDate' #if there is no age feature
Gender = 'gender'
District = ''
Postal_Code = ''
Insurance = ''
Specialty = ''
ID_Medic = 'healthProfessionalId'
Appointment_Date = 'expectedBeginning' ##Required
Appointment_Hour = '' #if the hour is in a separate feature
Scheduled_Date = ''
NoShow = 'showedUp' ##Required
NoShowPositiveValue = True #value used to signal a no-show #if it is 1 leave empty ('')

Other_Features = ['expectedEnd', 'medClickValue', 'patientValue', 'reminderSentAt', 'userAnsweredReminderAt',
#list of remaining features that will not be used
```

Figure 5.5: Feature configuration file used for the data coming from MedClick API.

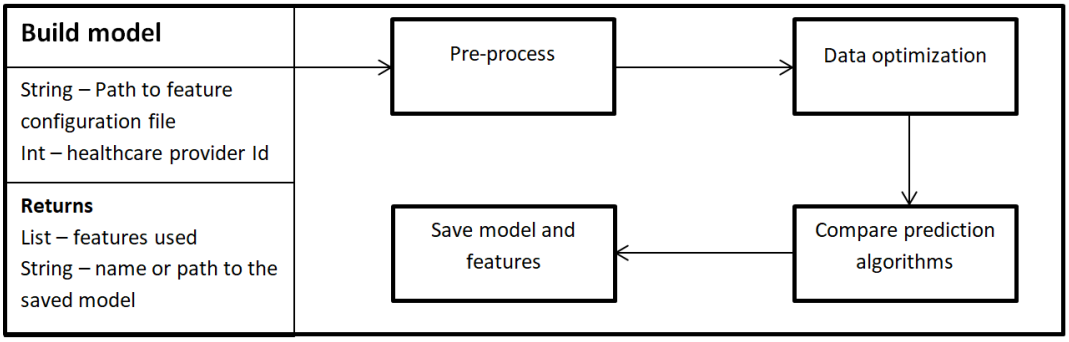

Figure 5.6: Build Model components.

5.2 Build Model

Build Model is the model that builds the prediction model, which is then used to obtain no-show predictions on the data. This data can come from a CSV file or directly from the MedClick API. The reason to have the CSV option is that most datasets, from the majority of clinics and hospitals, come in this format. Also, it will be important to have a way to quickly test these datasets without integrating them in the API, since due to data protection measures this will not happen right away.

Build Model receives as arguments the path to the feature configuration file to be used and the healthcare Id of the associated healthcare provider, this can be seen in figure 5.6. The model is then divided into four phases:

- Pre-process: Pre-processing is where the data from a specific healthcare provider is transformed into data that can be used by the prediction algorithms. This transformation adds new features from the existing ones to give the algorithms more information to learn from. It also removes or replaces missing values and transforms categorical variables with One-Hot encoding into dummy variables with values of 1 if true and 0 for false. Beyond this, all the numeric features are normalized into values in the range of 1 to 0, since the rest of the features are all binary this puts all features in the same range of values. This allows the algorithms to learn better without giving too much weight to features with high numerical values. Examples of these transformations can be seen in 6).

- Data optimization: In this phase, the data is ready to be used by the prediction algorithms, but first it should be optimized to give the best predictions possible. The first step is choosing only the features that possess stronger predictive power and removing the others. First variance threshold is used to remove features that are almost constant since these features will not contribute to the predictions. The next step is using a feature selection algorithm, the one chosen is Boruta. To validate the features chosen by this algorithm, it is used alongside a 10 fold cross-validation and for each cross-validation fold, the features chosen are registered and only the ones that appear more than

80% of the time are chosen. The final stage is to balance the data since most of the data come unbalanced with many more shows than no-shows. This can cause the prediction algorithms to prefer to classify appointments as shows in favour of having more accuracy. To mitigate this problem, SMOTE with Edited Nearest Neighbours is used which balances the data by generating data samples with SMOTE and then using k Nearest Neighbours it removes those samples that are misclassified by its neighbours. After the data is balanced, it is now ready to be fed to the prediction algorithms and be able to find which give better predictions.

• Compare Prediction Algorithms: It is impossible to find a prediction model that will be better for every scenario and every dataset, things like the size of the dataset and the number of features can affect the quality of the predictions for some algorithms. To solve this problem in this phase four different prediction algorithms are run on the dataset on cross-validation with 3 folds only, to prevent it from being very computationally expensive. The four prediction algorithms are Artificial Neural Network, Gradient Boosting, Logistic Regression and Random Forest. These algorithms were tweaked to provide better results, but the type of optimizations required for one dataset is not the same for other datasets. Because of this, general optimization was used in all datasets. After running the prediction algorithms, the one with the best overall score in the metric f1-score is the one chosen. This metric was chosen because, for these results, it will be more important to have the right balance between recall and precision than having good accuracy. At a later stage, this phase of comparing prediction algorithms can be removed once the data comes solely from the API since at this stage there will be more control on the features used and the prediction algorithm that generally performs better can be chosen. This will save computation time which will be more important at that stage.

• Save Model and Features: After we have chosen the model it must be saved, this is done using a pickle which is a python module that allows us to save the model in a file .dat. This file can then be easily loaded to make predictions for that healthcare provider. The name of the saved file will be unique having the id of the healthcare provider, the name of the model used and the accuracy obtained. This name along with the features chosen in the data optimization phase will be saved in the MedClick API to be used in the prediction phase. The reason to save the features, as well, is that the predictions need to be made with the same features the model was trained on, otherwise it will not work.

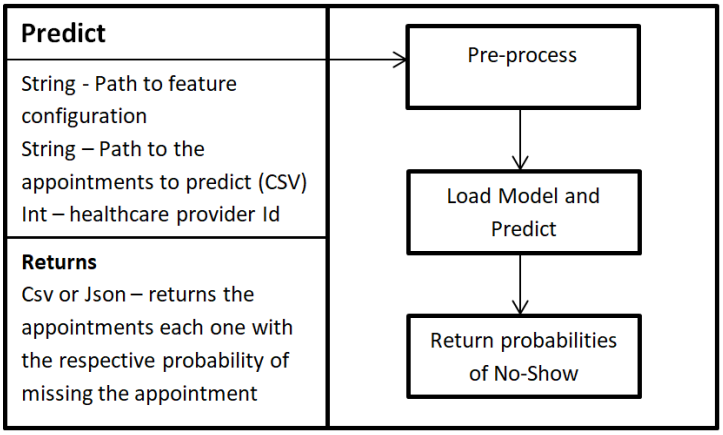

Figure 5.7: Predict Model components.

5.3 Predict Model

The Predict Model is the model used to obtain the probability of a patient missing his appointment. This model has two functions, one to obtain the probability of no-show for all the appointments scheduled, in every healthcare, and another to obtain the probability of no-show for a specific healthcare provider. The first one does not receive any argument and only works for the appointments in the API. This function is scheduled to be executed every hour or less so that the probabilities can be regularly updated and in the case, new appointments are added, we can quickly figure out what is the probability of no-show. This is especially important when appointments are scheduled for the same day or the next day. The other function is to predict for single healthcare, this function receives as arguments the path to the feature configuration and the healthcare id. This function also works for data in CSV, and in this case, it should receive an extra argument with the appointments we want to make predictions on.

The Predict Model has 3 phases, which are:

- Pre-process: In the pre-processing phase, the appointments which we want to predict are joined with the original dataset. This is done so that it is possible to pre-process the new appointments, in order for them to have all the features. Also, it is required to be able to put the right values in features like the number of appointments and the number of no-shows, since this needs to be calculated for the whole data. After this, the list of features saved in the MedClick API and associated with this healthcare provider is retrieved. The features that are not in this list of features are removed from the appointments to predict, this is done so they match the ones where the model was trained on, otherwise, it would not work. The numeric values are also normalized using MinMax Scaler so that everything is on the same scale.

- Load model and predict: In this phase, the name of the model used to train the data is retrieved

from the API and using python module pickle, the model is loaded. Using the loaded model the predictions for the probability of no-show are obtained for all of the appointments.

• Return probability of no-show: If the appointments come from the API, the list of probabilities must be uploaded to the API. This is done by using the appointment identifier and for each one uploading the respective probability of no-show to the API. Beyond this, an explanation of how the algorithm obtained that prediction is updated with the probability of no-show. This explanation is obtained using Lime, which gives a value for how relevant the features were to the prediction and returns a list with the nine most relevant features for that prediction. An example of these explanations plotted can be seen in figure 5.8 and figure 5.9. With this, it is possible to have a better idea of how the prediction algorithms are obtaining that probability and, in these early stages, will allow a better understanding of which features have the most impact and which tweaks can be made to improve it. In case the appointments come from a CSV file, the main process is the same but the results are saved to a CSV file, instead of uploading them to MedClick API. This file will have the appointments predicted with the original dataset features and an extra feature with the probability of no-show. An example of this file calculated for the MD Clínica dataset can be seen in figure 5.10.

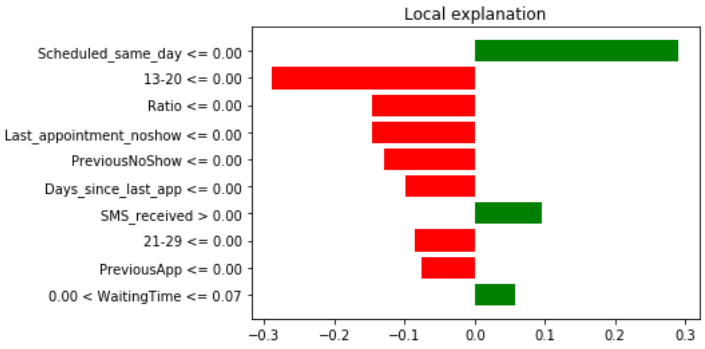

Figure 5.8: Plotted explanation obtained using Lime for the Brazil dataset.

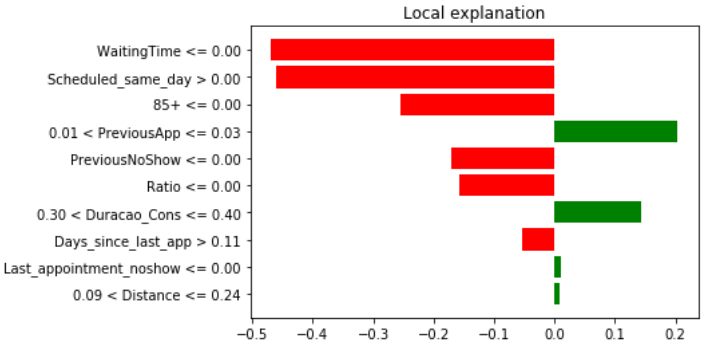

Figure 5.9: Plotted explanation obtained using Lime for the MD Clínica dataset.

patient_id	Codigo_Pc	freguesia	Medico	Data	Hora_Consulta	Duracao_Cons	Data_Marcacao	Ultima_Data_Alteracao	horachg	horaentr	estado_consulta	NoShow_prob
97909	0000-000		97	11/10/2019	15:40	0.4	05/06/2019	49:21.2			Paciente Remarc	0.181855
200384	4920-040	COVAS	49	11/10/2019	10:14	0.4	10/10/2019	41:57.2		10:15:45	Realizada	0.465617
200396	1200-738		55	11/10/2019	16:20	0.4	09/10/2019	23:09.2	16:22:13		Realizada	0.473088
49971	1070-048	CAMPOLID	30	11/10/2019	11:00	0.3	04/10/2019	18:27.7	10:38:32		Realizada	0.761971
99241	0000-000		81	11/10/2019	10:00	0.6	07/02/2019	28:21.0			Paciente Faltou	0.832626
200141	1170-289	ARROIOS	49	11/10/2019	08:40	0.4	11/03/2019	41:56.4			Paciente Desistiu	0.872578
87801	0000-000		65	11/10/2019	13:00	0.3	09/10/2019	12:20.0			Realizada	0.085584

Figure 5.10: Example of the calculated probabilities for a part of data from MD Clínica dataset.

5.4 No-show Module Integration

To integrate the prediction model with the current version of the no-show module in the MedClick system, some changes to the algorithms and the way the no-show module behaved were made. These changes were mainly in the prediction part of the no-show module.

The No-show module has two distinct tasks, the first one corresponds to training the model on the available data, which will now translate to running the build model on the desired data. This task will run once a month or until there has been a significant addition of new data to the dataset. The retraining of the model is important to ensure the model does not become unstable and that it is updated with fresh data since new events or newly adopted behaviours from the patients can eventually affect the precision of the predictions. Since this is a computationally expensive task and it takes time for a real change to occur in the data and affect predictions, this retraining can be done in a larger window of time. An example of an event that can happen to affect predictions is the current pandemic, since using trained data before the pandemic to predict appointments during the pandemic will probably not lead to good predictions. The opposite is also true using data trained during the pandemic will not lead to good predictions in the future. This means that the time to rebuild the model and the amount of data to build it must take into consideration the current events.

The second task is one that should be performed daily and in smaller windows of time. This task is comprised of two sub-tasks, one of them runs daily and is responsible to send reminders to the patients, with appointments five days from now. No changes were made to this sub-task but these reminders and patient confirmations will be an important step to prevent no-shows and eliminate false positives. The other sub-task used to run each day at 8:00 am and was responsible to compute the probability of no-shows on the appointments three days from now, using the trained model. This sub-task was changed, it runs the Predict Model instead and runs every hour of the day. The reason it runs every hour instead of just at the beginning of the day is that appointments are constantly being updated and rescheduled so having these predictions just at the beginning of the day would be very limited. Appointments can also be scheduled for the same day or the next day, and even though the risk of a no-show in this cases is lower, it is important to have the no-show probabilities so healthcare providers have time to act. Another change made was on the appointments that will have their probability computed, instead of just computing for the next three days, it will compute the probability for all the appointments and it will be up to each clinic and hospital, to define the window of time to replace the patients that have high probabilities of no-show. This way smaller clinics and hospitals that would struggle to find a last-minute replacement can have more time to find a replacement. While larger healthcare providers with many interested patients in their waiting lists can decide to replace the patients latter when they are more certain that the high no-show probability will translate to a no-show.

5.5 Project Structure

The prediction model was developed in python and using NodeJS. In this section, the most relevant files used to create the model and connect it to the API are shown. The structure and the way they are connected can be seen in figure 5.11.

- app.js: this file is used to call the functions to run the prediction model. It is used to run the Build Model and the Predict Model. It is also responsible for running the tasks included in the scheduler.js file.
- api.js: this file contains the methods that are responsible for fetching and posting data on the MedClick API
- scheduler.js: this file contains tasks to be performed repeatedly over time, it includes daily tasks such as sending reminders to the patients and hourly tasks to calculate the probability of no-show for each appointment.
- noShow.js: this file contains the main methods of the prediction problem and is used to call the methods in the python connector.
- pythonConnector.js: this file is one of the most important files in the prediction model since it is responsible for connecting the model built in python with the rest of the NodeJs files. It also calls the methods from the api.js file to be able to get the healthcare data and pass it to the python algorithm. It is also responsible for receiving the models and information from the python classifier and push it into the MedClick API.
- predModel.py: this is the python script where the model is built and the appointments predicted based on the data received. It contains every machine learning technique used to obtain the models and calculate the no-show probability.
- feature_configuration.py: this is the configuration file that must be tweaked by the user for the algorithm to work with the given dataset.

Besides the above-mentioned files and the typical NodeJS configuration files, there is a statistics.py which was the file used to obtain the results and graphs shown in chapter 7. Other less relevant files contain helper methods.

To implement the prediction model it was required to make changes to all of the files in figure 5.11. The files that suffered most alterations were api.js and pythonConnector.js, other files like predModel.py and feature_configuration.py were created almost from scratch. In the api.js file new functions and methods had to be created so it was possible to retrieve the required data from the MedClick API and

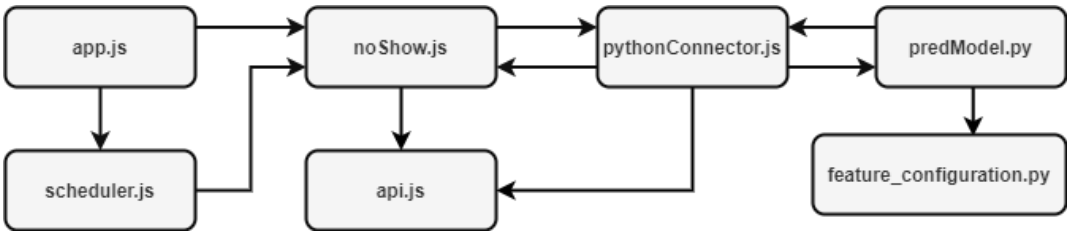

Figure 5.11: Structure of project files.

to filter it by healthcare. It was also required to create functions to post the model name, features and probabilities in the MedClick API. The pythonConnector.js file is where the most important functions used by the prediction model were built, to run the build the model and make the predictions.

6

Dataset Analysis

Contents

In this chapter, the characteristics and features of each dataset used are analysed. Three datasets from three different healthcare providers were used to test the prediction algorithms and machine learning techniques. The first one from a dental clinic (MD Clínica), the second one was obtained online from Kaggle and comprises data from a Brazilian dataset and the last one is from Hospital da Luz which is the largest dataset of the three. Here the main differences in the features of each one of the datasets can be seen, it is also shown the transformations that were made to the features in order to test the datasets with the prediction models.

6.1 MD Clínica Dataset

This dataset was provided by a Portuguese clinic, MD Clínica, this clinic had already provided data that was used in the thesis by Daniel Sousa and the thesis by Inês Ferreira to test the algorithms they implemented. The dataset provided is more recent and provides more features than those used before, which allows us to extract more information than before and further test the implemented algorithms and new prediction techniques.

6.1.1 Characteristics

The initial dataset had 152 393 records each corresponding to an appointment from a dental clinic in Lisbon and comprises data from 1 January 2019 to 31 October 2019. It has a feature comprised of several states that describe various stages of the appointment, being the relevant ones: the appointments that were completed, the ones were the patient did not show up and the ones were the patient rescheduled. After removing the states that were not relevant and some important values that were missing of important features, 90 419 records remain in the dataset. Of this records 58 872 were shows and 31 547 no-shows giving a distribution in percentage of 65% shows and 35% no-shows. The patient's dataset was separated from the original dataset and had to be merged to obtain some features like gender, postal code and birth date. The combined features of the original dataset merged with the patient's dataset can be seen in the following table 6.1, along with a brief description.

Not all of the features represented in this table 6.1 were used since many of them had too many missing values or were not useful to calculate the prediction probability values. The original dataset has 18 features but after removing these feature only 11 features remain.

One issue with this dataset is the use of the variable the patient rescheduled as a no-show. The reason is if a patient rescheduled one month before the appointment it probably should not be considered a no-show. This variable is used as a no-show because this clinic always calls patients before their appointment to check if they are coming to the appointment or not if they cannot the clinic tries to

Feature	Description
Patient_id	Patient's identifier.
Postal_code	Patient's postal code.
County	Patient's county.
Date	Appointment's date.
Medic Id	Medic's identifier.
Appointment_hour	Hour of the appointment.
Appointment_duration	Duration of the appointment.
Scheduled_date	Appointment's scheduled date.
Last_alteration_date	Date last status change.
Hour_of_arrival	Hour the patient arrived.
Hour_get_in	Hour the patient got in.
Appointment_status	Appointment's status (accomplished, rescheduled, cancelled, patient missed, etc.).
Birthdate	Patient's birthdate.
Marital_status	Patient's marital status.
Gender	Patient's gender.
Profession	Patient's profession.
Convention	Patient's insurance identifier.
Name_convention	Patient's insurance name.

Table 6.1: Features in the MD Clínica's original dataset.

reschedule them. This means that the majority of these rescheduled appointments would be no-shows, but not all of them and since we cannot figure out which, to maintain all the data all of them were used.

To add more information to this dataset another one was obtained, which contains the weather information. This dataset was obtained from the national information system of water resources [30] from the Lisbon region and contains four features, which can be seen in the next table 6.2.

Feature	Description
Average_humidity	Average humidity value for that day
Average_temperature	Average temperature value for that day
Precipitation	Average precipitation value for that day
Wind_speed	Average wind speed value for that day

Table 6.2: Features in the weather dataset.

6.1.2 Transformed Data

To be able to use the above features in the prediction algorithms and to give more information for the prediction algorithms to learn from, the data needs to be transformed. Before using the python algorithm to transform the data, it was necessary to transform the Appointment_status into the binary feature of No-show, where appointments completed are shows and appointments cancelled and rescheduled are no-shows. Afterwards and using the feature configuration file in figure 5.3, some features were added out of the existing ones and some were changed to provide the algorithms with more information, the

final set of features with the added features and the updated ones can be seen in table 6.3.

Feature	Description
Distance	Obtained from postal code.
XX_Medic	A new feature binary feature was added for each one of the medics in the Medic feature.
XX_Month	A new feature binary feature was added for each one of the months.
XX_Weekday	A new feature binary feature was added for each one of the weekdays.
XX_Age	The age was divided into 10 different age groups with a binary feature for each one of them. The age groups are ((0-4), (5-12), (13-20), (21-29), (30-39), (40-49), (50-59), (60-69), (70-84), (>85)).
XX_Insurance	A new feature binary feature was added for each one of the insurances in the Insurance feature.
Part_of_the_day	A binary feature for each of the parts of the day (early morning, morning, lunch, afternoon, night).
Gender	Binary feature (1 if male, 0 if female).
WaitingTime	Time between the day the appointment was scheduled and the appointment itself in days.
PreviousApp	Number of previous appointments.
PreviousNoShow	Number of previous no-shows.
Ratio	Number of previous no-shows divided by the number of previous appointments.
Scheduled_during_last_app	Whether the patient scheduled his appointment in the same day of his last appointment.
Scheduled_same_day	Whether the appointment was scheduled in the same day of the appointment.
Last_appointment_noshow	Whether the patient last appointment was a no-show.
Days_since_last_app	Days since the patient had his last appointment

Table 6.3: Features used to test the prediction algorithms in the MD Clínica dataset.

Some features like age, insurance, weekday, month and medic were divided into binary features using one-hot encoding, this was done because many of this features are categorical and to be used in the prediction algorithms must be altered to numeric. In the case of the age feature it was transformed into age groups because age can be considered to be a categorical feature and after testing both strategies, this one provided better information to the prediction algorithm and thus better results. For the feature distance, it had to be calculated from the postal code, this was done using python library pgeocode. Since it took a lot of time to calculate the distance using pgeocode, all the unique postal codes were put on a list and then the distance was calculated. Afterwards, a dictionary with the distance and postal code was used, this saved a lot of computation time in further tests.

The total number of features in the dataset is 114, this is a significant increase from the original number of features and provides much more information. Some of these features are removed during

the feature selection phase, this is done to remove some of the noise and leave the algorithms with only the features that have more predictive power. Beyond these transformations, all the numeric features had to be normalized to be on the same scale with MinMax scaler.

6.2 Brazil Dataset

This dataset was obtained online from Kaggle, it has data from clinics in Brazil. It had already been previously tested in the thesis by Inês Ferreira [2]. Since new machine learning techniques are being used and new features were added it makes sense to have this dataset as an extra validation tool. It also provides distinct features which can be analyzed to find out how strong they are. This dataset is not from Portugal, which is not ideal since MedClick will operate only in Portugal, but it also allows us to analyze whether these results are similar in other parts of the world.

6.2.1 Characteristics

The dataset has 110528 records where 22319 are no-shows, which means the dataset has a distribution of approximately 80% shows and 20% no-shows. This dataset comprises data from 29 April 2016 to 8 June 2016. There are 12 features in the original dataset, which are shown in table 6.4.

Feature	**Description**
Date	Appointment's date.
Scheduled_date	Appointment's scheduled date.
Age	Patient's age.
Scholarship	Whether the patient has a scholarship, which is a type of aid given by the Brazil government to poor families.
Hipertension	Whether the patient suffers from hipertension.
Diabetes	Whether the patient suffers from diabetes.
Alcoholism	Whether the patient is alcoholic.
Handcap	Whether the patient has any disability.
SMS_Received	Whether the patient received a SMS.
Gender	Binary feature (1 if male, 0 if female).
Neighborhood	Neighborhood of the patient.
NoShow	Number of previous appointments.

Table 6.4: Features in the original dataset from Brazil.

All the features in this table 6.4 were used, even though some were adapted into new ones. The reason we were able to use all the features in this dataset is that it did not have almost any missing values. The only problems found were some negative values in age and some scheduling dates earlier than appointment dates. Since it, only a affected a very few numbers of rows they were removed.

6.2.2 Transformed Data

To extract value from the above features they must be transformed. To do this the feature configuration file in figure 5.2 was used to automatically transform the features into the ones required. Some features were left out of the configuration file because the features are already binary and do not need any pre-processing. These features were Scholarship, Hypertension, Diabetes, Alcoholism, Handicap and SMS_Received. The features with categorical values were handled using one-hot encoding and the rest of the features were obtained using calculations with the rest of the features. The final set of features used in this dataset, along with a short description, can be seen in table 6.5.

Feature	Description
XX_Month	A new binary feature was added for each one of the months.
XX_Weekday	A new binary feature was added for each one of the weekdays.
XX_Neighborhood	A new binary feature was added for each one of the neighborhoods.
XX_Age	The age was divided into 10 different age groups with a binary feature for each one of them. The age groups are ((0-4), (5-12), (13-20), (21-29), (30-39), (40-49), (50-59), (60-69), (70-84), (>85)).
Scholarship	Whether the patient has a scholarship, which is a type of aid given by the Brazil government to poor families.
Hipertension	Whether the patient suffers from hipertension.
Diabetes	Whether the patient suffers from diabetes.
Alcoholism	Whether the patient is alcoholic.
Handcap	Whether the patient has any disability.
SMS_Received	Whether the patient received a SMS.
Gender	Binary feature (1 if male, 0 if female).
WaitingTime	Time between the day the appointment was scheduled and the appointment itself in days.
PreviousApp	Number of previous appointments.
PreviousNoShow	Number of previous no-shows.
Ratio	Number of previous no-shows divided by the number of previous appointments.
Scheduled_during_last_app	Whether the patient scheduled his appointment in the same day of his last appointment.
Scheduled_same_day	Whether the appointment was scheduled in the same day of the appointment.
Last_appointment_noshow	Whether the patient last appointment was a no-show.
Days_since_last_app	Days since the patient had his last appointment.

Table 6.5: Features obtained after pre-processing the Brazil dataset.

After the pre-processing phase, the dataset had 116 features more 104 than the original dataset. The increase in the number of features can be justified by the creation of a new binary feature for each

one of the categorical values. These features will still pass through a feature selection phase where the number of features used is greatly decreased. All the numerical features in this dataset were also normalized using MinMax Scaler.

6.3 Hospital da Luz Dataset

This dataset was obtained from Hospital da Luz and contains the most records of the three datasets used with 494 627 records.

6.3.1 Characteristics

The original dataset has 494 627 records and comprises data from 2 January 2017 to 31 December 2017. This dataset had some missing values but not enough to discard any of the features. Other small issue found, was that some of the scheduling dates happened after the appointment date, since this is not possible and it was a very small amount of rows they were removed. The distribution of no-show for this dataset is approximately 90% shows and 10% no-shows. The original dataset has 15 features with some unique features not found in the above datasets like speciality, scheduled way and first or follow through. The features of the original dataset can be seen in the following table 6.6, along with a brief description. All the data was fully anonymised prior to the work.

Feature	**Description**
ID	Patient's identifier.
Age	Patient's age.
Postal_code	Patient's postal code.
Gender	Patient's gender.
County	Patient's county.
District	Patient's district.
Insurance	Patient's insurance identifier.
Speciality code	Speciality of the appointment's identifier.
Speciality description	Speciality of the appointment's description.
First or follow through	Whether it is the first appointment or a follow through.
Medic Id	Physician ID.
Scheduled_date	Appointment's scheduled date.
Date & Hour	Appointment's date and hour.
Scheduled_way	Whether the appointment was scheduled in person.
No Show	Whether this appointment was a no-show.

Table 6.6: Features of the original dataset belonging to Hospital da Luz.

All the features of this dataset were used except for speciality description and county. Speciality description provides the same information as the speciality code so there is no need to use it. County has a lot of categorical values, which would translate to a lot of binary features since the distance can be obtained from the postal code and there is also district there is no need to have this extra layer.

6.3.2 Transformed Data

To transform the dataset the feature configuration file in figure 5.4 was used. We can see that first or follow through and scheduling way are not in the configuration file, this happens because these features have values of yes or no which can be easily transformed into binary. Since these two features are unique to this dataset there was no need to put them into the configuration file. The final set of features used in the prediction algorithms can be seen in the following table 6.7, along with a short description.

Feature	**Description**
Distance	Obtained from postal code.
XX_Medic	A new binary feature was added for each one of the medics in the Medic feature.
XX_Month	A new binary feature was added for each one of the months.
XX_Weekday	A new binary feature was added for each one of the weekdays.
XX_Age	The age was divided into 10 different age groups with a binary feature for each one of them. The age groups are ((0-4), (5-12), (13-20), (21-29), (30-39), (40-49), (50-59), (60-69), (70-84), (>85)).
XX_Insurance	A new binary feature was added for each one of the insurances in the Insurance feature.
XX_District	A new binary feature was added for each one of the districts in the District feature.
XX_Speciality	A new binary feature was added for each one of the specialities in the Speciality feature.
Part_of_the_day	A binary feature for each of the parts of the day (early morning, morning, lunch, afternoon, night).
Gender	Binary feature (1 if male, 0 if female).
WaitingTime	Time between the day the appointment was scheduled and the appointment itself in days.
PreviousApp	Number of previous appointments.
PreviousNoShow	Number of previous no-shows.
Ratio	Number of previous no-shows divided by the number of previous appointments.
Scheduled_during_last_app	Whether the patient scheduled his appointment in the same day of his last appointment.
Scheduled_same_day	Whether the appointment was scheduled in the same day of the appointment.
Last_appointment_noshow	Whether the patient last appointment was a no-show.
Days_since_last_app	Days since the patient had his last appointment.
First_or_follow_through	Whether it is the first appointment or a follow through.
Scheduling_way	Whether the appointment was scheduled in person.

Table 6.7: Features obtained after pre-processing the dataset from Hospital da luz.

After the pre-processing, the dataset was transformed into 819 amount of features which is a large

number. This happened because there are a lot of categorical variables that were transformed with one-hot encoding into new features. Since many of these features are almost constant with a very few positive values they are removed in the feature selection phase. For the distance variable, the same process of creating a dictionary with the unique postal codes and respective distance was used, which saved a lot of computation time. The features chosen to be used for the prediction algorithms can be seen in the following chapter, where we can see that only a small amount of features end up being used.

7

Results and Discussion

Contents

In this chapter, the results obtained in each of the different datasets are analysed. The prediction algorithms and machine learning techniques were tested in datasets from three different healthcare providers. The datasets used and the features used in each of them are discussed in the previous chapter. With this analysis, it is possible to see which machine learning techniques and what conducts are most efficient at predicting no-shows. It is also discussed in the respective conclusions if the results obtained were good enough to help attenuate the negative effects of no-shows.

7.1 Feature Selection Analysis

In this section, the most important features and the feature selection techniques used in each one of the datasets are analyzed. Feature selection is done by first using a variance threshold to remove features that have a variance inferior to 0.01. This is done because features with a very low variance are basically constant and will not improve the performance of the model. After this, a 10 fold cross-validation is used and the feature selection algorithm is tested in each one of the folds, returning for each one of the folds the selected features. Only the features that are selected in eight or more folds are chosen as the features to be used. By doing this, with 10 fold cross-validation, we make sure that the features chosen are not overfitting to the data. The feature selection algorithms tested are Recursive Feature Elimination, Boruta and LASSO. To be able to compare the results a prediction algorithm is run alongside the features chosen in each one of the folds. The prediction algorithm used is Gradient Boosting, since the goal is to compare which feature selection technique provides the best results only one prediction algorithm is required. The normalization is also done independently in each fold, using MinMax scaler to prevent overfitting.

This comparison was done in each one of the datasets to understand whether there is one feature selection technique that performs better in all the datasets.

7.1.1 MD Clínica Dataset

This subsection is about how the feature selection algorithms performed in the MD Clínica dataset. First variance threshold was used removing 35 features, leaving the dataset with 79 features. Afterwards, each one the feature selection algorithms was used. With Recursive Feature Elimination only 39 features remained, which can be seen in figure 7.1. The Boruta algorithm removed the most features, leaving the dataset with 10 features, figure 7.2. Lasso does not remove any feature, which means 79 features remain since this number is too large it could not be represented in a figure.

The results obtained by each one of the feature selection techniques for the MD Clínica dataset can be seen in the following table 7.1 and figure 7.3.

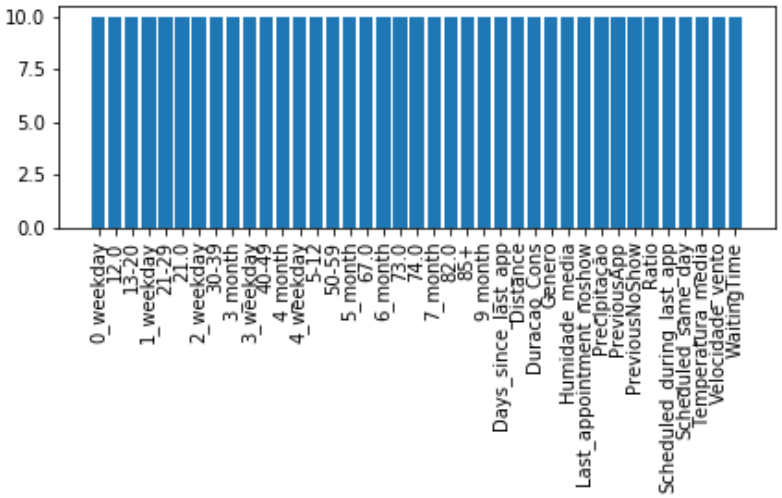

Figure 7.1: Plot with features chosen by RFE in each one of the folds.

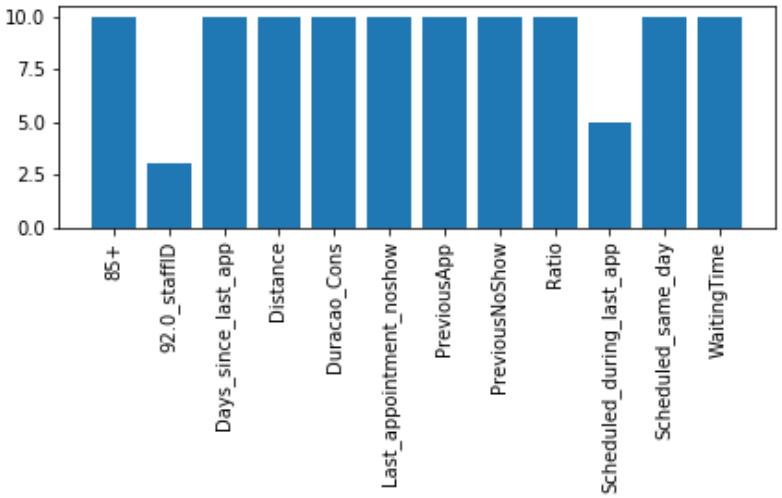

Figure 7.2: Plot with features chosen by Boruta in each one of the folds.

Algorithm	Accuracy	Recall	Specificity	Precision	ROC(AUC)	F1-score	PR(AUC)
RFE	0.687	0.24	0.919	0.604	0.579	0.343	0.547
Boruta	0.688	0.236	0.922	0.609	0.579	0.34	0.545
Lasso	0.691	0.261	0.914	0.61	0.587	0.365	0.553

Table 7.1: Comparison of the results achieved by the feature selection algorithms on the dataset from Hospital da Luz.

Looking at the results, it is not possible to see a significant difference between the different feature selection techniques. Even though each one of the algorithms chose a different number of features, the prediction algorithm, in this case, Gradient Boosting, is able to properly weight each one of the features during computation, making it work well with many features or just the important ones. This

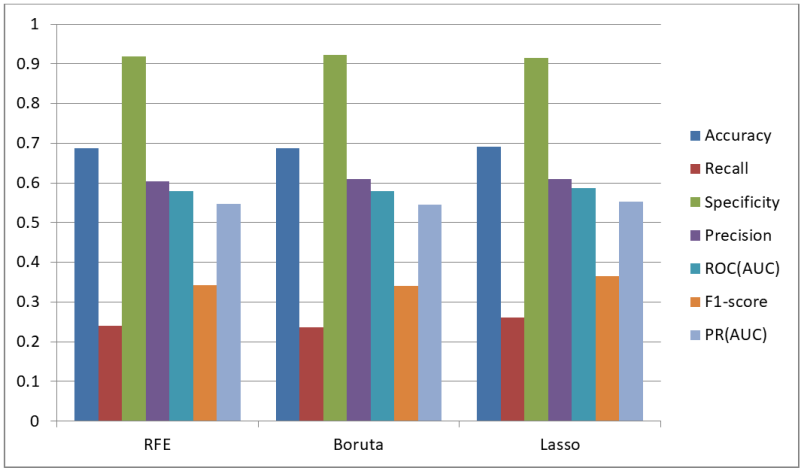

Figure 7.3: Chart that compares the feature selection techniques in the dataset from MD Clínica.

means that the noise in the other features does not affect the predictions and many of those features are unnecessary. The feature selection algorithm that performs slightly better is LASSO, which uses the most features. But since this difference is so small the other algorithms, mainly Boruta, has to be considered since it achieves equivalent results with a very small number of features. Using fewer features in very large datasets will be important since it will save lots of computation time, which can increase exponentially with a high number of features.

7.1.2 Brazil Dataset

For the Brazil dataset, 120 features were removed using variance threshold out of 188 total. The Recursive Feature Elimination selected 31 features, figure 7.4, while Boruta selected 12 features, figure 7.5. LASSO removes 3 features leaving the dataset with 65 features. Boruta chose the least features, while LASSO the most.

The results obtained by each one of the feature selection techniques for the Brazil dataset can be seen in the following table 7.2 and figure 7.6.

Algorithm	Accuracy	Recall	Specificity	Precision	ROC(AUC)	F1-score	PR(AUC)
RFE	0.806	0.07	0.992	0.7	0.531	0.127	0.421
Boruta	0.806	0.069	0.993	0.704	0.531	0.125	0.415
Lasso	0.806	0.074	0.991	0.688	0.533	0.134	0.428

Table 7.2: Comparison of the results achieved by the feature selection algorithms on the Brazil dataset.

It can be seen in the table and chart that all the feature selection algorithms achieved very similar results, so in this example, there is no difference between the feature selection algorithms, beyond the fact that fewer features lead to much faster computation times.

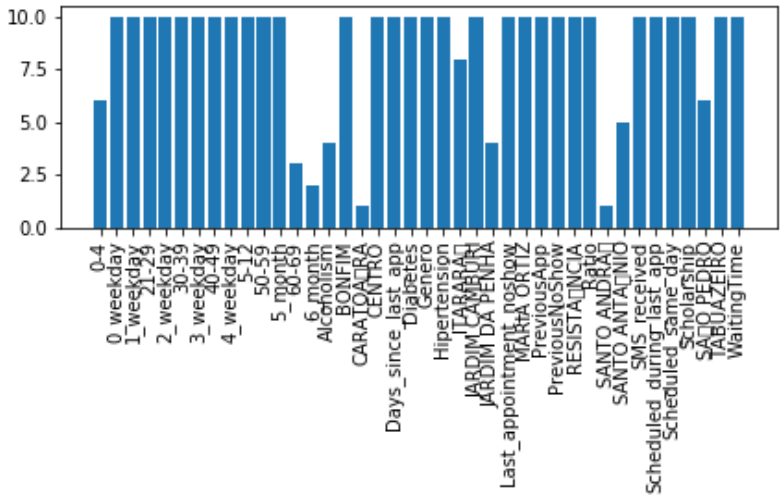

Figure 7.4: Plot with features chosen by RFE in each one of the folds.

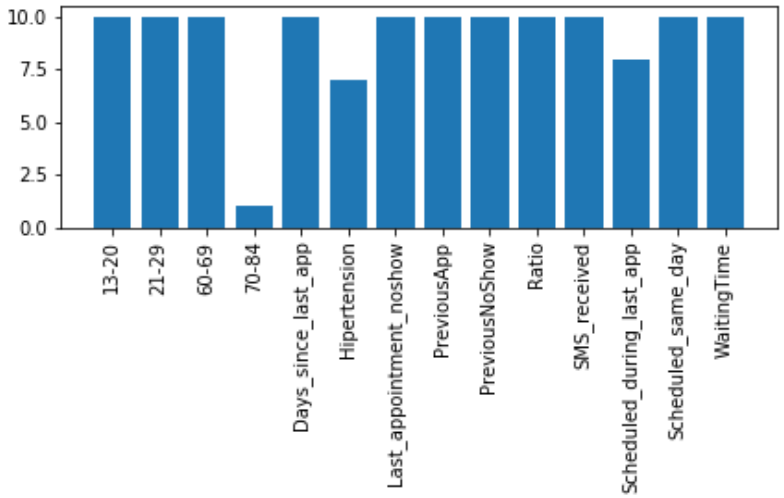

Figure 7.5: Plot with features chosen by Boruta in each one of the folds.

7.1.3 Hospital da Luz Dataset

After being pre-processed this dataset had 819 features, most of them being binary features from the categorical features in the original dataset. After using variance threshold to remove features with a variation smaller than 0.01, only 83 features remain. Afterwards, the feature selection algorithms were applied. With LASSO 80 features were chosen out of the 83. As for Recursive Feature Elimination 39 features were selected, which can be seen in figure 7.8. Boruta chose the least features with 21 being chosen, which can be seen in figure 7.7.

It was also tried using all the features directly on the feature selection algorithm without variance threshold, the problem with this approach was that for Recursive Feature Elimination and Boruta there

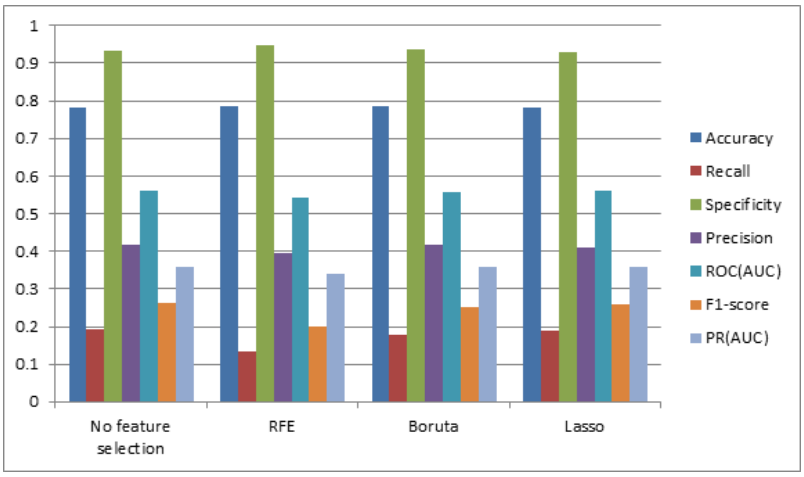

Figure 7.6: Chart that compares the feature selection techniques in the Brazil dataset.

were memory problems in the machine. It worked for LASSO which used on average 600 features, but the results achieved by Gradient Boosting algorithm were very similar and slightly worst than the ones obtained here. Since the results were not different, using variance threshold was concluded to be the best approach, as it makes the whole process less computationally expensive.

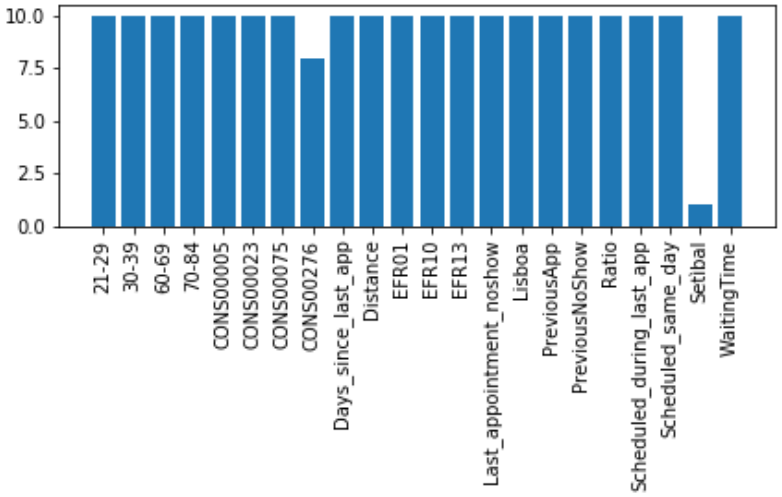

Figure 7.7: Features chosen by boruta for Hospital da Luz dataset for all the 10 folds.

The results obtained by each one of the feature selection techniques for the dataset from Hospital da Luz can be seen in the following table 7.3 and figure 7.9.

As we can see in the tables the results are almost equal. The high accuracy stands out but the dataset is imbalanced and the recall is very low, which means it is not finding the many no-shows. Like the above datasets, there does not seem to be any advantage of using feature selection algorithms, beyond the fact that for very large datasets, like this one, using fewer features allows for faster computation

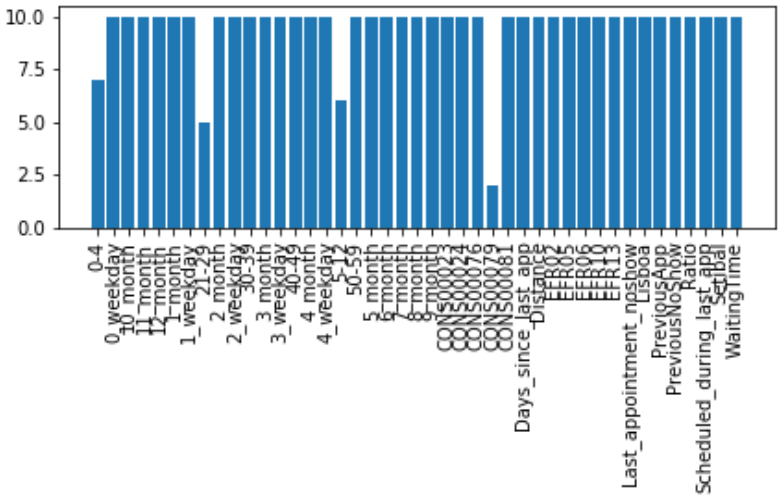

Figure 7.8: Features chosen by recursive feature selection for Hospital da Luz dataset for all the 10 folds.

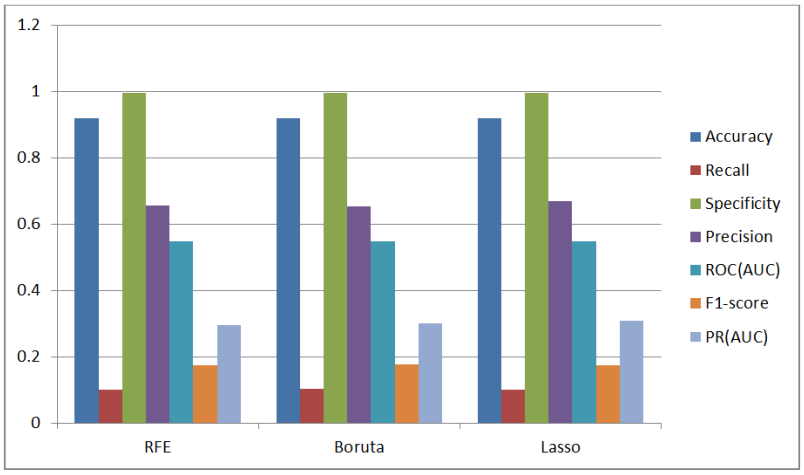

Figure 7.9: Chart that compares the feature selection techniques in the dataset from Hospital da Luz.

Algorithm	Accuracy	Recall	Specificity	Precision	ROC(AUC)	F1-score	PR(AUC)
RFE	0.918	0.099	0.995	0.656	0.547	0.173	0.296
Boruta	0.918	0.102	0.995	0.654	0.549	0.177	0.301
Lasso	0.919	0.099	0.995	0.669	0.547	0.173	0.307

Table 7.3: Comparison of the results achieved by the feature selection algorithms on the dataset from Hospital da Luz.

times and less computational resources are required.

7.1.4 Results Comparison

After analysing how each feature selection algorithm performed in all the datasets, we can conclude that all feature selection algorithms perform almost the same. Since some of these algorithms use lots of

features while some use a much smaller number, it can be inferred that the prediction algorithm used to obtain these results, Gradient Boosting, is automatically giving lots of weight to certain features and disregarding the rest. This means that there are a lot of irrelevant features that do not need to be used by the prediction algorithms, which will help reduce noise and improve computation times. The feature selection algorithm that uses the least features, while simultaneously obtaining equivalent results is Boruta. This algorithm should be used for the prediction model since it allows us to know which features are relevant and reduces computation time, which will be important as data increases and is used by MedClick.

7.1.5 Feature Importance Comparison

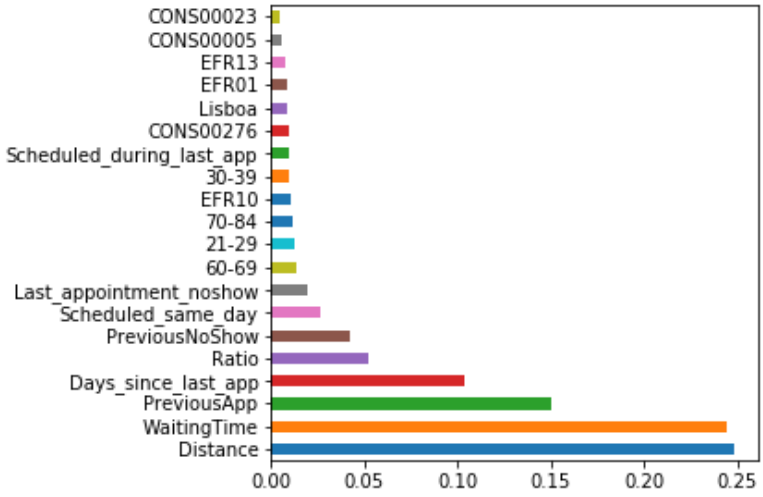

Figure 7.10: Feature importance graph for the dataset from Hospital da Luz.

In the following figures 7.10,7.11 and 7.12 we can see the most relevant features for each one of the datasets. We can see that most of the chosen features are similar between the different datasets, which means some constant features are better in predicting no-shows. The most relevant feature is waiting time, it seems the time from when the appointment was scheduled to the time of the appointment is crucial to find no-shows. One reason for this is appointments with a long waiting time are easily forgotten, which is one of the most reported reasons for no-shows. Another reason is it is hard to predict how your life will be set up in a long time from now leading to the patients having to reschedule or no-show. Another feature which is also correlated to waiting time is scheduled the same day. This happens when the waiting time is zero and means that the appointment was scheduled for the very day. Patients with appointments scheduled like this will most likely show to the appointment since they committed for that time means that they need it urgently, or at least, they are sure they can make it on time. Since there

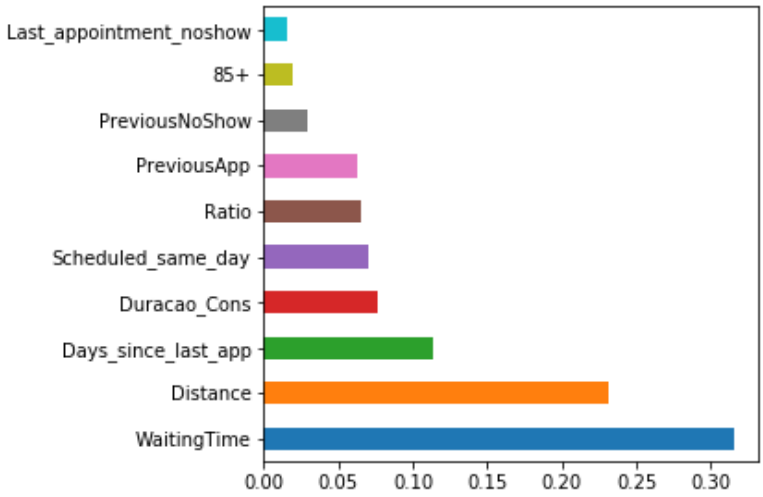

Figure 7.11: Feature importance graph for the dataset from MD Clínica.

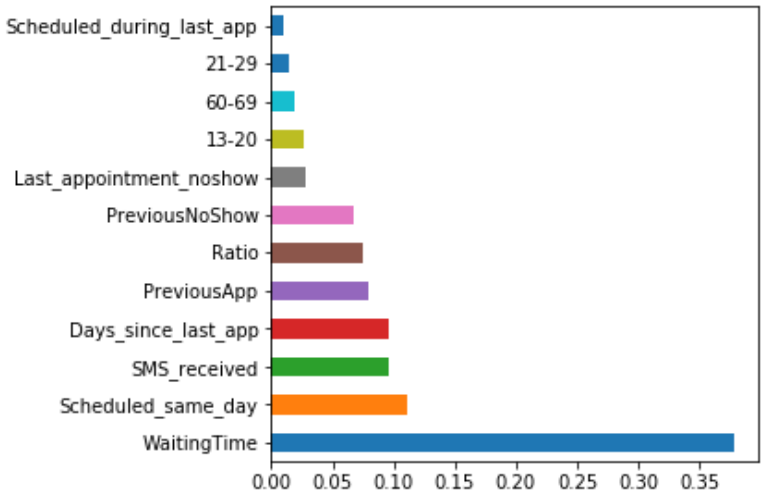

Figure 7.12: Feature importance graph for the dataset from Brazil.

are many appointments like this in all datasets it helps increase the importance of waiting time.

Another very important feature is the distance, which has even slightly more importance than waiting time in the dataset from Hospital da Luz. This feature is the distance between the postal code of residence and hospital postal code. Reasons of distance being important can be obvious, if a patient lives really close it is not hard for him to show up to the appointment, while a patient that lives far away, any delay at work or small unpredictable event can lead to a no-show. The opposite can also be true a patient who lives very far away might have scheduled there because it is important for him so he might be more committed to showing up and have already prepared his scheduled to go. The combination with other features can help the algorithm answer this question. It was not possible to obtain this feature

for Brazil dataset, because the available features did not allow this to be calculated.

Other relevant features that follow these two but with a large difference, with no specific order, are the days since the last appointment, number of previous appointments, number of previous no-shows, ratio and whether the last appointment was a no-show. These features have different values of importance depending on the dataset but they are chosen in all the datasets, which means these features are also very important to accurately predict a no-show.

Other unique features of some datasets that got a considerable value of importance are appointment duration (Duracao_Con), which is specific to the appointments from MD Clínica and whether a message was received (SMS_Received), which is specific to the appointments from Brazil. These features can lead to better results in the predictions and, as such, an attempt should be made to make this available on other obtained datasets and, more importantly, on the data from the MedClick API.

7.2 Sampling Techniques Analysis

Sampling techniques are important to balance the datasets when the dataset is imbalanced. It brings some advantages, like increasing recall and subsequently the number of no-shows found. In this section, we can see how every sampling technique performed for each dataset. The sampling techniques used are SMOTE, a combination of SMOTE and Random Under Sampler, SMOTE with Tomek Links and SMOTE with Edited Nearest Neighbors.

The sampling techniques were tested using the features chosen by the feature selection algorithm Boruta, this algorithm was chosen since it used only the relevant features and provided better computation times. These sampling techniques were also tested with a 10 fold cross-validation and Gradient Boosting was chosen as the prediction algorithm to compare which sampling technique yields the best results. It was tested on all three datasets and the results obtained can be seen in the next subsections.

7.2.1 MD Clínica Dataset

For MD Clínica the average results obtained for all the 10 folds can be seen in the following figure 7.13 and table 7.4).

Algorithm	Accuracy	Recall	Specificity	Precision	ROC(AUC)	F1-score	PR(AUC)
NoSampling	0.688	0.236	0.922	0.609	0.579	0.34	0.545
SMOTE	0.675	0.424	0.806	0.53	0.615	0.471	0.544
SMOTE&RUS	0.644	0.627	0.653	0.484	0.64	0.546	0.551
SMOTETomek	0.665	0.496	0.753	0.51	0.625	0.503	0.544
SMOTEENN	0.602	0.715	0.544	0.448	0.63	0.551	0.561

Table 7.4: Comparison of the results achieved by the sampling techniques on the dataset from MD Clínica.

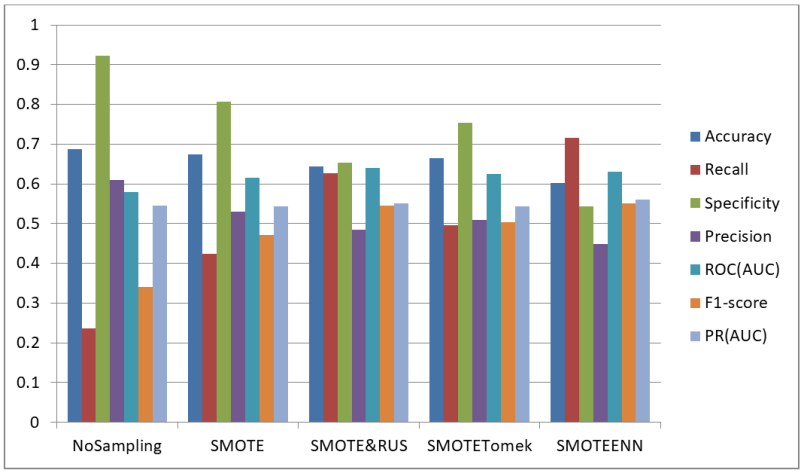

Figure 7.13: Chart that compares the sampling techniques in the dataset from MD Clínica.

We can see there is the loss of accuracy, specificity and precision from using these sampling techniques but there is a significant gain in the recall. Which also leads to improvements in the ROC AUC score and f1-score. The difference between the different sampling techniques is basically on how steep this loss and gain is. The sampling technique with the largest gain and lost is SMOTE with Edited Nearest Neighbour. This technique achieves a recall of 0.715, which is much larger than the original value of 0.236. This comes with a consequence of lowest value for accuracy, specificity and precision, but still, lead to a better f1-score. The sampling technique with the lowest difference is SMOTE, which only achieves a recall score of 0.424 but maintains high accuracy. SMOTE with RUS also achieves good results with the best value for ROC AUC. The best sampling technique could be the one with a higher f1-score, in this case, SMOTE with Edited Nearest Neighbours, but this will depend from case to case.

7.2.2 Brazil Dataset

We can be seen in the following figure 7.14 and table 7.5 the results obtained for the Brazil dataset.

Algorithm	Accuracy	Recall	Specificity	Precision	ROC(AUC)	F1-score	PR(AUC)
NoSampling	0.806	0.069	0.993	0.704	0.531	0.125	0.415
SMOTE	0.721	0.508	0.774	0.363	0.641	0.423	0.401
SMOTE&RUS	0.698	0.592	0.725	0.353	0.659	0.442	0.409
SMOTETomek	0.708	0.551	0.747	0.356	0.649	0.432	0.398
SMOTEENN	0.677	0.663	0.68	0.344	0.672	0.453	0.415

Table 7.5: Comparison of the results achieved by the sampling techniques on the dataset from Brazil.

In these results we can see there is a trade-off between precision and recall as in the previous dataset. The accuracy and specificity also decrease with the increase in recall. The sampling technique that achieves the higher recall is SMOTE with Edited Nearest Neighbors. It achieves 0.663 a

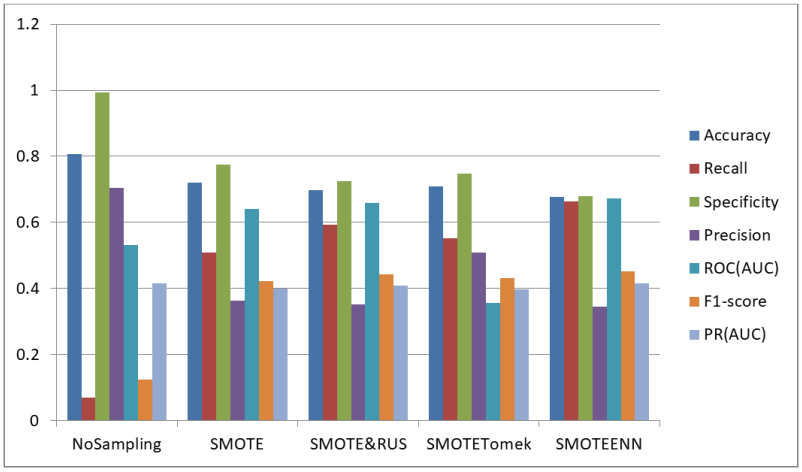

Figure 7.14: Chart with the results achieved by the sampling techniques on the dataset from Brazil.

large increase from the original 0.069. This technique has the lowest precision as a consequence but still manages to have better ROC AUC and f1-score. The other sampling techniques achieved not so different results but have slightly higher values for accuracy and precision with a lower recall.

7.2.3 Hospital da Luz Dataset

In Hospital da Luz dataset the results obtained using the sampling techniques are shown in the following figure 7.15 and table 7.6).

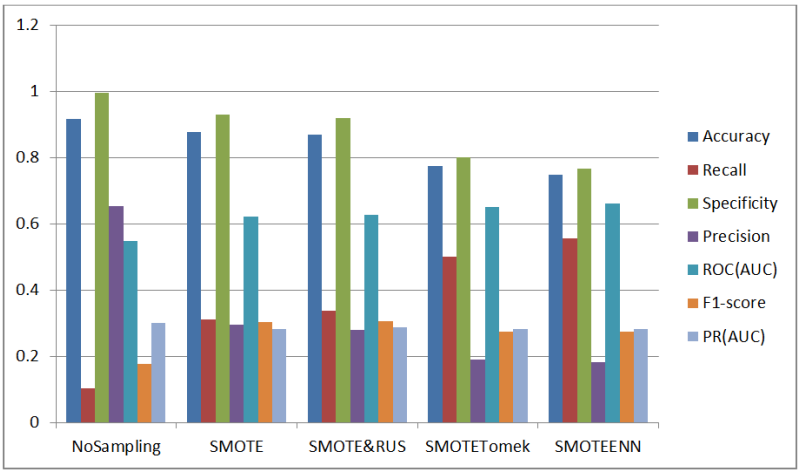

Figure 7.15: Chart that compares the sampling techniques in the dataset from Hospital da Luz.

In these results, SMOTE with Edited Nearest Neighbours is again the one with the highest increase in recall with 0.557, but it comes at the cost of a very low precision, which leads to an f1-score lower than other sampling techniques. The sampling technique with the highest f1-score, even though this score is very similar to the others is SMOTE combined with RUS. The recall achieved is 0.311, from the original

Algorithm	Accuracy	Recall	Specificity	Precision	ROC(AUC)	F1-score	PR(AUC)
No Sampling	0.918	0.102	0.995	0.654	0.549	0.177	0.301
SMOTE	0.877	0.311	0.93	0.296	0.621	0.303	0.283
SMOTE&RUS	0.869	0.337	0.919	0.279	0.628	0.305	0.286
SMOTETomek	0.774	0.5	0.8	0.19	0.65	0.275	0.281
SMOTEENN	0.748	0.557	0.766	0.182	0.661	0.274	0.283

Table 7.6: Comparison of the results achieved by the sampling techniques on the dataset from Hospital da Luz.

0.102, but precision suffers a large drop from 0.654 to 0.279.

7.2.4 Conclusion

The results obtained using the sampling techniques are still not ideal. Increasing recall and decreasing precision means more no-shows are found but at the cost of making more mistakes. How worth this manoeuvre is depends on how much the f1-score improves at the same time. It also depends on the problem at hand if it is required more precision or to find as many positive classes as possible. For the no-show problem, it will depend a lot on how each hospital and clinic operates. For some to have a strategy where they would find as many no-shows as possible and then using a patient confirmation strategy on those patients only could save the clinic many resources, for example in the MD Clínica they would not need to call every patient. Other clinics or hospitals might be very strict and like to maintain patient waiting times to a minimum so they might only want to replace patients that they are 100% sure they will miss. Because of this, no sampling technique can be made final and for some not using a sampling technique can be more beneficial. For the MedClick case, a sampling technique is used in the prediction model, since it is beneficial at this stage so the initial tests can find more no-shows. When combined with the no-show module the sampling technique will also be beneficial, since the no-show module will use a confirmation strategy that can eliminate a lot of the false positives.

7.3 Prediction Algorithms Comparison

This section compares four prediction algorithms to find out, which ones can provide more reliable predictions. The four algorithms used are Artificial Neural Network, Gradient Boosting, Logistic Regression and Random Forest.

To compare the prediction algorithms, Boruta was chosen as the feature selection algorithm and SMOTE with Edited Nearest Neighbours was chosen as the sampling technique. Boruta was chosen because it uses the least amount of features and has a faster computation time. It is also a good practise using the same techniques used before throughout this analysis to make things easier to compare. SMOTE with Edited Nearest Neighbours was chosen because it achieved the most f1-score in most of

the tested datasets and has the highest recall in all them.

This algorithm was executed in a 10 fold cross-validation and the average scores for each one of the prediction algorithms were obtained. This was tested on three datasets to find out if there is a prediction algorithm that performs well on all the datasets.

7.3.1 MD Clínica Dataset

Algorithm	Accuracy	Recall	Specificity	Precision	ROC(AUC)	F1-score	PR(AUC)
ANN	0.557	0.729	0.468	0.416	0.599	0.529	0.546
GB	0.602	0.715	0.544	0.448	0.63	0.551	0.561
LR	0.516	0.853	0.341	0.401	0.597	0.546	0.541
RF	0.607	0.709	0.554	0.452	0.632	0.552	0.56

Table 7.7: Comparison of the results achieved by the prediction algorithms on the dataset from MD Clinica.

The table 7.7 describes the average scores obtained by the prediction algorithms. Logistic Regression achieves the best recall but with an accuracy of only 51.6%, which is almost the same as guessing. Gradient Boosting and Random Forest achieve very similar results to each other, with the best results for all the metrics, except recall. The result obtained for f1-score is good but the accuracy achieved is still not enough as it should be at least 70% to be useful in a real-world scenario.

In the following figures, we can see boxplots that show how the prediction algorithms performed in each iteration for a specific metric. The most noticeable thing is how random forest and gradient boosting outperform the other algorithms on all metrics except for recall. Another thing we can see is that the results of the Artificial Neural Network are all spread out, while the results from the other prediction algorithms are in a very close range of values. This means the Artificial Neural Network is not a very trustworthy prediction algorithm, since it can have a very different range of results.

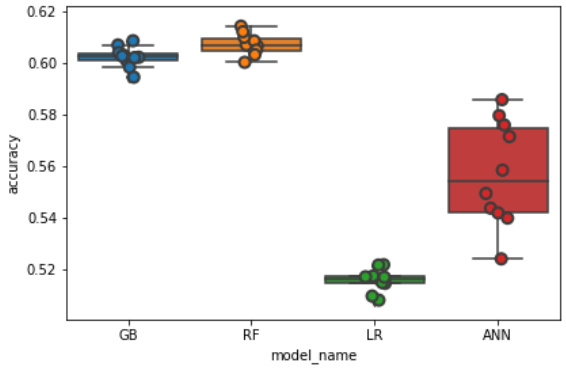

Figure 7.16: Boxplot that compares the accuracy of the prediction algorithms.

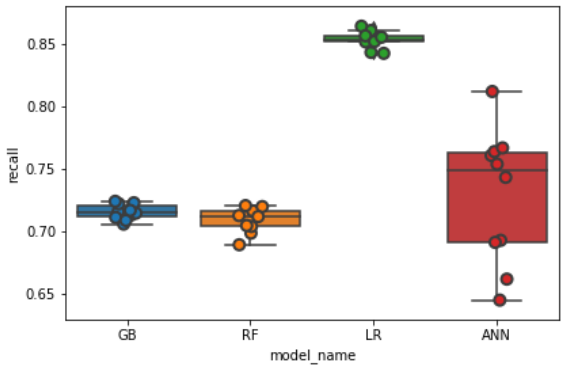

Figure 7.17: Boxplot that compares the recall of the prediction algorithms.

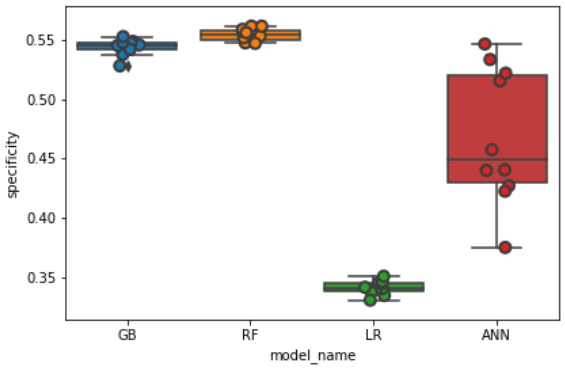

Figure 7.18: Boxplot that compares the specificity of the prediction algorithms.

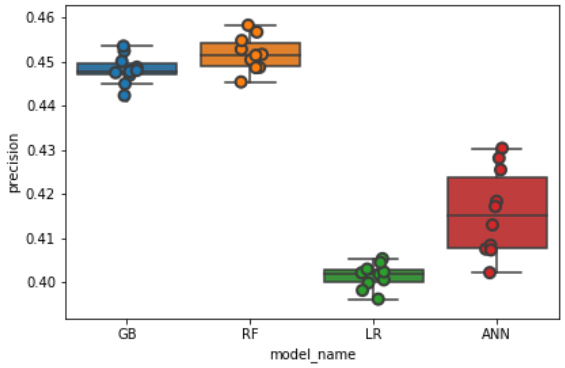

Figure 7.19: Boxplot that compares the precision of the prediction algorithms.

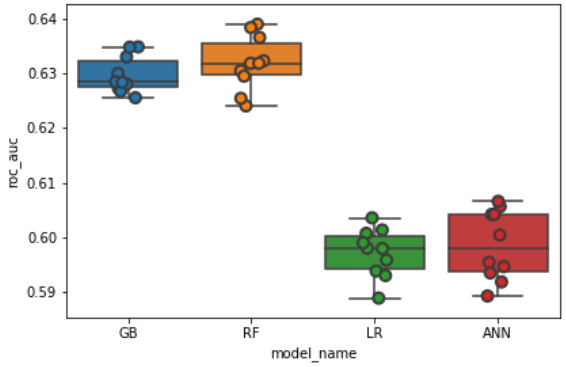

Figure 7.20: Boxplot that compares the ROC AUC of the prediction algorithms.

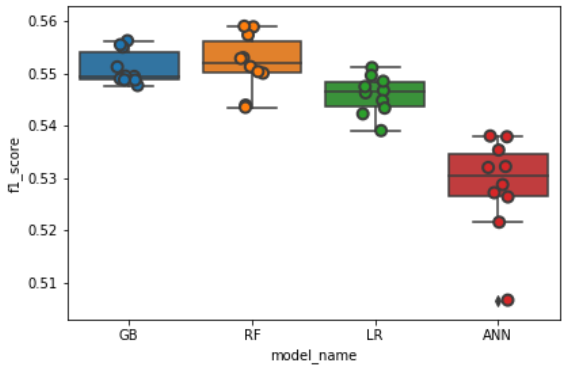

Figure 7.21: Boxplot that compares the f1-score of the prediction algorithms.

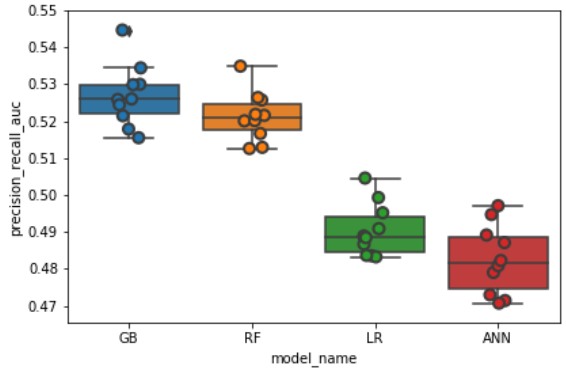

Figure 7.22: Boxplot that compares the precision-recall AUC of the prediction algorithms.

7.3.2 Brazil Dataset

Algorithm	Accuracy	Recall	Specificity	Precision	ROC(AUC)	F1-score	PR(AUC)
ANN	0.64	0.69	0.627	0.319	0.659	0.436	0.403
GB	0.677	0.663	0.68	0.344	0.672	0.453	0.415
LR	0.584	0.836	0.521	0.306	0.678	0.448	0.423
RF	0.716	0.521	0.766	0.36	0.643	0.426	0.409

Table 7.8: Comparison of the results achieved by the prediction algorithms on the dataset from Brazil.

In this table 7.8 we can see how the prediction algorithms performed on the Brazil dataset. The two algorithms with the higher f1-score are Logistic Regression and Gradient Boosting. Logistic Regression also achieves a very high recall of 0.836 without losing too much precision but the accuracy obtained of approximately 60% is not enough. For Gradient Boosting the recall is not as high but manages a much better accuracy. The algorithm that offers the most conservative approach is Random Forest, with the highest accuracy and precision.

In the next figures, there are the boxplots from the results obtained over the 10 fold cross-validation iterations. We can see each prediction algorithm chosen behaves a little different, some provide a more balanced approach with intermediate results in the metrics, while others have really good scores in one metric but always with a trade-off in other metrics. In this case, the results of Artificial Neural Network are not as spread out as observed in the previously analysed dataset.

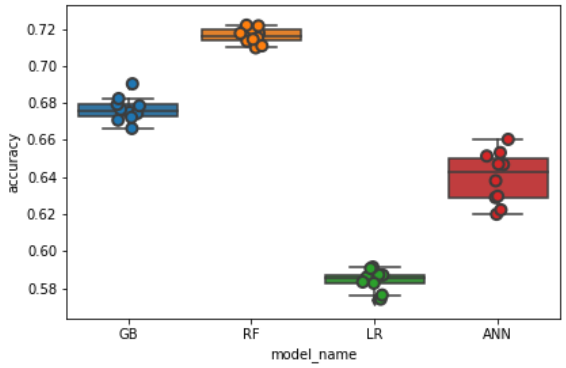

Figure 7.23: Boxplot that compares the accuracy of the prediction algorithms.

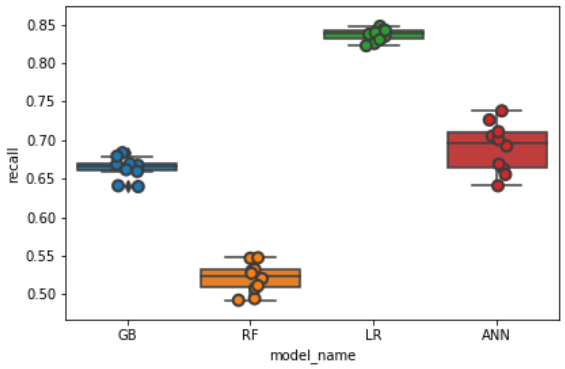

Figure 7.24: Boxplot that compares the recall of the prediction algorithms.

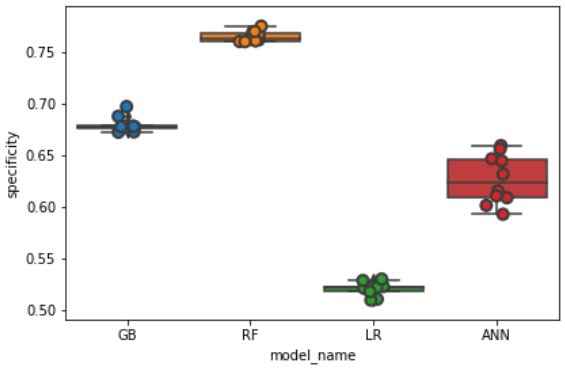

Figure 7.25: Boxplot that compares the specificity of the prediction algorithms.

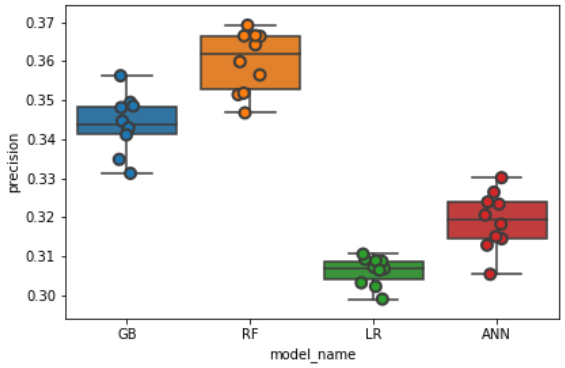

Figure 7.26: Boxplot that compares the precision of the prediction algorithms.

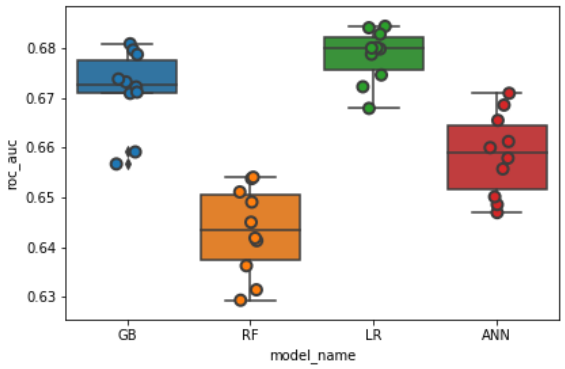

Figure 7.27: Boxplot that compares the ROC AUC of the prediction algorithms.

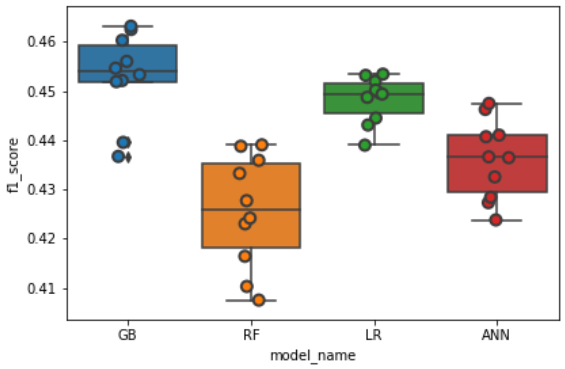

Figure 7.28: Boxplot that compares the f1-score of the prediction algorithms.

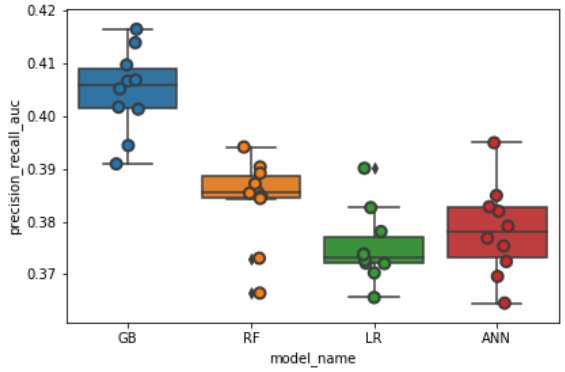

Figure 7.29: Boxplot that compares the precision-recall AUC of the prediction algorithms.

7.3.3 Hospital da Luz Dataset

Algorithm	Accuracy	Recall	Specificity	Precision	ROC(AUC)	F1-score	PR(AUC)
ANN	0.639	0.684	0.635	0.15	0.659	0.246	0.272
GB	0.748	0.557	0.766	0.182	0.661	0.274	0.283
LR	0.614	0.7	0.61	0.143	0.653	0.237	0.22
RF	0.803	0.406	0.84	0.192	0.623	0.261	0.234

Table 7.9: Comparison of the results achieved by the prediction algorithms on the dataset from Hospital da Luz.

In the above table 7.9, we can see the results obtained in Hospital da Luz dataset. One thing that can be noticeable right away is that the scores for precision and f1-score are much lower than in previous examples. This happens because the original dataset was very imbalanced and after applying a sampling technique it sacrificed precision for more recall. No prediction algorithm performed really well, and even though Artificial Neural Network and Random Forest achieved a really good accuracy, the recall and precision need to be higher and more balanced to be beneficial in a real-world scenario.

In the boxplots shown in the following figures, we can compare the prediction algorithms in each iteration. It is possible to see that no prediction algorithm outperforms the others on all the metrics, there is always one metric each algorithm performs better on. Gradient Boosting is the algorithm the more balanced algorithm with respectable results in all metrics and with the highest f1-score.

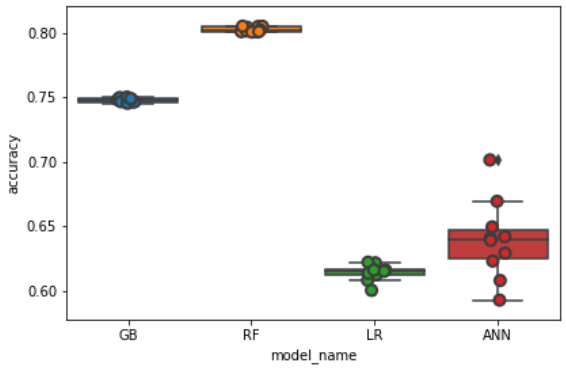

Figure 7.30: Boxplot that compares the accuracy of the prediction algorithms.

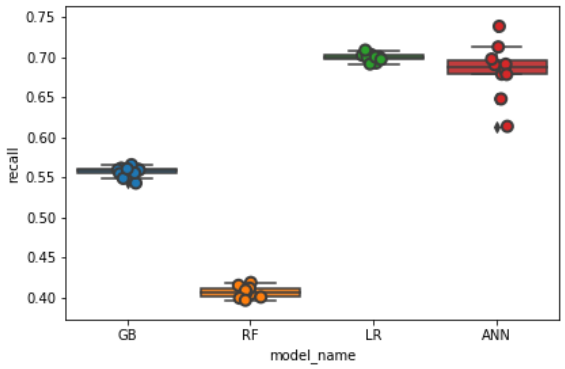

Figure 7.31: Boxplot that compares the recall of the prediction algorithms.

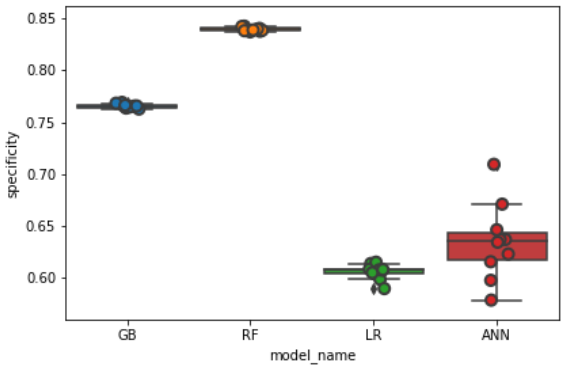

Figure 7.32: Boxplot that compares the specificity of the prediction algorithms.

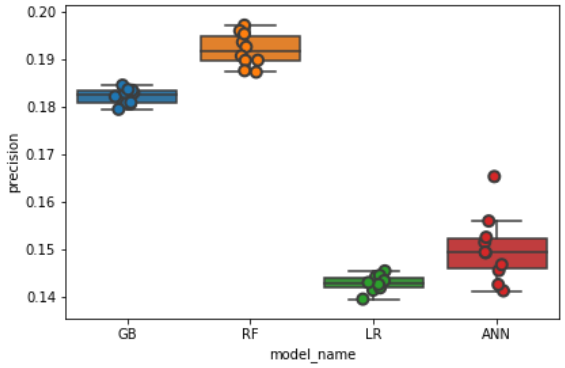

Figure 7.33: Boxplot that compares the precision of the prediction algorithms.

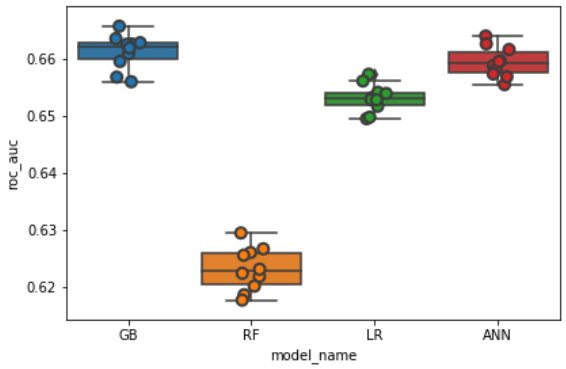

Figure 7.34: Boxplot that compares the ROC AUC of the prediction algorithms.

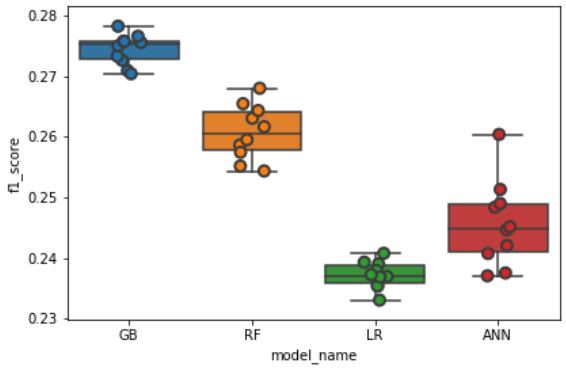

Figure 7.35: Boxplot that compares the f1-score of the prediction algorithms.

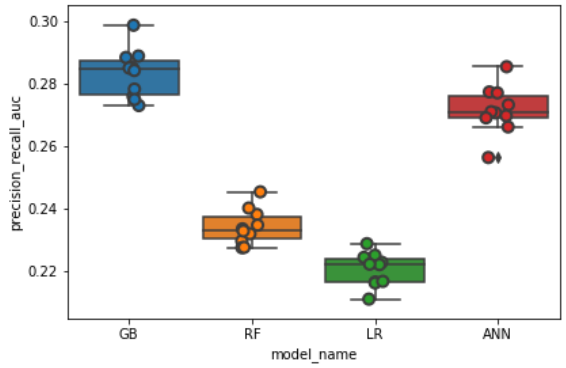

Figure 7.36: Boxplot that compares the precision-recall AUC of the prediction algorithms.

7.3.4 Real-life scenario simulation

This is an attempt to mimic a real-life scenario and find out how many no-shows and misclassifications happen. This simulation was done using the last week of each dataset for testing and the rest for training. A threshold of 70% was also used, what this means is that only no-shows with a probability of 70% or more are considered no-shows. This threshold was used because in many hospitals and clinics only no-shows with a high probability will be considered to avoid overbooking. To make this tests the sampling technique used was SMOTE with Edited Nearest Neighbours and the feature selection was Boruta.

In the next table 7.10, we can see the comparison between the confusion matrices for all datasets and prediction algorithms. The amount of appointments for that week varies for each dataset, Brazil is the one with more appointments compressed into that week with a much higher number than the others. In all of the datasets, it is possible to see that on average 50% of no-shows are found by the prediction algorithms. The prediction algorithm that finds more no-shows in all datasets is Artificial Neural Network but it also has the highest number of false positives. On the other hand, we have Random Forest with the least number of false positives but fewer no-shows found, which translates to a more conservative and precise approach.

In MD Clínica dataset we can see that on average for every no-show found there is a false positive. The prediction algorithm with the best results here is Gradient Boosting since it finds almost as many no-shows as Artificial Neural Network but at a much smaller cost of false positives. The Random Forest algorithm could also be used for a more conservative approach, as it has the least amount of misclassifications, making it the most precise of the four.

In Brazil dataset, for every no-show found there is slightly more than the double of false positives. The prediction algorithms with the best results here are Logistic Regression and Gradient Boosting with similar results and more balance approaches than the other two algorithms.

In the dataset from Hospital da Luz, there is almost the triple of false positives compared to no-shows found. Here it is clear that the more the original dataset is imbalanced the more false positives are to be expected. The best prediction algorithm here is Gradient Boosting since it is even more precise than Random Forest and finds more no-shows. Also, the number of no-shows found is not that distant from Artificial Neural Network but with less false positives.

In the next figures 7.37 and 7.38, we can see the plot of the ROC curve and Precision-Recall curve in all the datasets. This was also obtained using only the last week for training.

In the ROC curve, we can see that is not a big difference between the results obtained by each prediction algorithm. The ones that stand out more are Gradient Boosting and Artificial Neural Network in the dataset from MD Clinica and the dataset from Hospital da Luz. These curves need to be much closer to the upper left edge for the results to be considered good.

In the Precision-Recall curves, we can see that no prediction algorithm really achieves very good

Algorithm	MDClinica	Brazil	Hospital da Luz
ANN	[873 385] [292 373]	[12808 5109] [1785 2285]	[4335 704] [250 222]
GB	[931 327] [300 365]	[13651 4266] [2089 1981]	[4617 422] [289 183]
LR	[903 355] [359 309]	[13469 4469] [1982 2088]	[4342 697] [268 204]
RF	[993 265] [356 306]	[13866 4051] [2315 1755]	[4611 428] [329 143]

Table 7.10: Comparison of confusion matrices for all datasets and prediction algorithms.

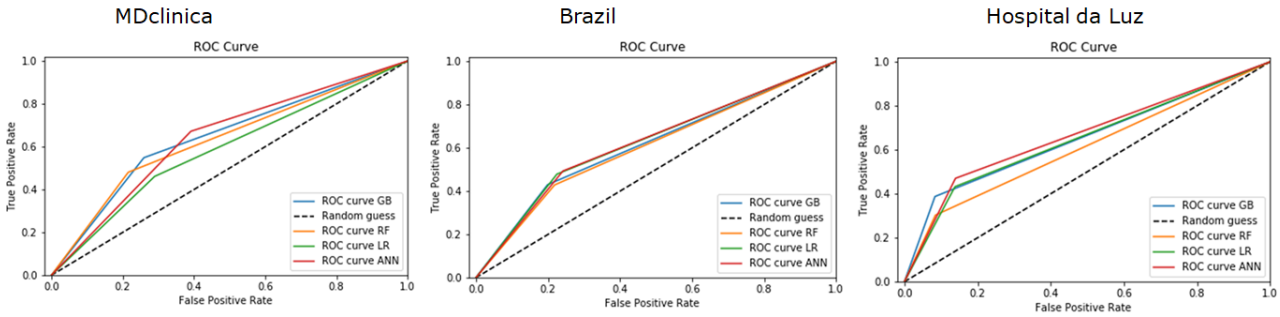

Figure 7.37: Graphs that compares the ROC curves in all datasets.

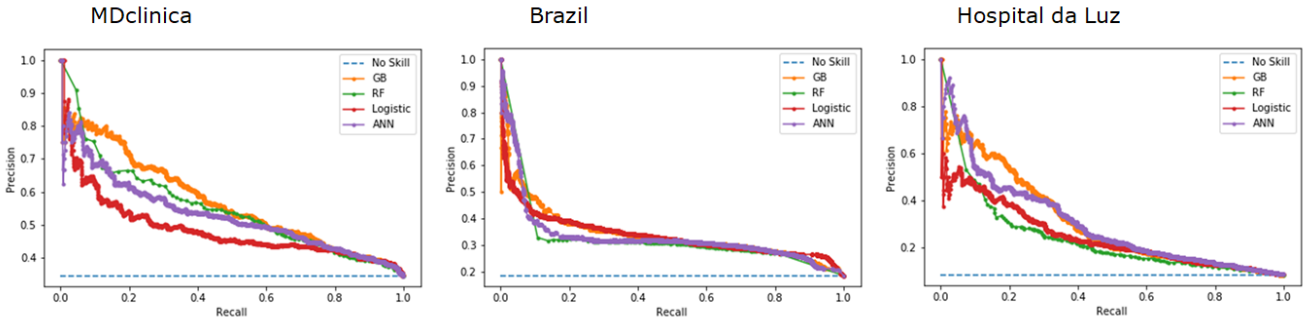

Figure 7.38: Graphs that compares the Precision-Recall curves in all datasets.

results. The curves in this case should be as close to the upper-right edge as possible. One prediction algorithm stands out on the dataset from MD Clnica and the dataset from Hospital da Luz, which is Gradient Boosting. It was able to have better results in both of these cases.

The Brazil dataset is the one with the worst results for all the prediction algorithms on both of the plots.

7.3.5 Conclusion

No prediction algorithm was found to be better in all the scenarios, the most consistent one is Gradient Boosting, but it will also depend on what is required. If a more conservative approach with more precision is required than Random Forest or Gradient Boosting are the best options. If the goal is to find as many no-shows as possible with the cost of many misclassifications then Logistic Regression or Artificial Neural Network are the best options.

No prediction algorithm can be discarded with the tests made, as larger datasets or different features can change the type of results, this is especially the case for Artificial Neural Network which needs many data to learn efficiently. More tests will be required to choose a prediction algorithm, but at the last stages of the prediction model for MedClick, it will be important to have just one prediction algorithm. This would reduce the computation time and resources spent. If a prediction algorithm is found to be constantly outperforming the others then it should be chosen.

8

Conclusions

Contents

8.1 Contributions

This thesis was done in the healthcare area focusing on the no-show problem. It seeks to find and implement a solution capable of reducing no-shows and subsequently increase efficiency in the healthcare providers. The work was done in the MedClick context, which is a company with the same goals.

The major contributions in new solutions and tests are done for the no-show problem were the following ones:

- **Creation of a prediction model to optimize and automate testing:** A prediction model was created to make the training of new models and obtaining of predictions from datasets easier and more efficient. Since all the datasets come with different characteristics and features, it would be required to change the code every time. This way the pre-processing phase and training phase were optimized, requiring a configuration file only to train the model and to make predictions. The prediction model was also integrated into the MedClick no-show module.

- **Prediction model integrated with MedClick API:** The prediction model was integrated to work with the data in the MedClick API. Functions had to be created for retrieving the data and saving the models and no-show probabilities in the MedClick API.

- **Added and tested new features:** In the previous work done for the MedClick application, only a very small number of features had been tested. With this work, many new features were added and tested in an attempt to figure out which features are more relevant and improve prediction results.

- **Machine learning algorithms tested on three different datasets:** Many machine learning techniques and different algorithms were used, in an attempt to get better predictions for no-shows. These techniques were tested in three different datasets with different characteristics, which helps to validate the techniques used and find out which prediction algorithms work better.

8.2 Conclusions

The no-show problem affects clinics and hospitals all over the world and reducing this problem can bring many social and economic benefits. As mentioned in the previous section, the work done is this thesis tries to attenuate this problem by implementing a solution in the MedClick platform that can predict and prevent no-shows from happening. There were already two theses made in the MedClick context on the same subject. This thesis adopted previously used ideas and further developed the prediction model

adding new machine learning techniques and more features to the datasets. It is hard to compare these results to the previous ones since different techniques were used and the way the other theses were evaluated were not ideal.

The main conclusions that can be made are on the results obtained while testing the prediction algorithms on three different datasets. These datasets had different sizes and provided different features. The first thing we can conclude is that the size of the dataset did not have a large impact on results. What impacted more was the type of features available and how much imbalanced the data was. An imbalanced dataset here is one where there are many more shows than no-shows. In these cases, we can conclude that the more the dataset is imbalanced the more false positives will happen. Having the right type of features is also very important to obtain good results. The features that were considered more important to identify no-shows are waiting time and distance. Other features that were chosen by every feature selection algorithm and also have a good level of importance are the days since the last appointment, number of previous appointments, number of previous no-shows, ratio and whether the last appointment was a no-show. Some datasets also had some unique features considered important these were appointment duration and whether an SMS had been received. These features should be added to the other datasets whenever possible.

Using a feature selection algorithm was concluded to be good for computation time but did not lead to better results. Using many features or very few did not affect predictions too much, this might be because the prediction algorithm used to evaluate, in this case, Gradient Boosting, can weight the features well and discard the least important. This means that the feature selection algorithm that can find only the relevant features will be more useful. The algorithm that did this was Boruta, which constantly chose the least number of features.

Since all these datasets were imbalanced, sampling techniques were used to counter this problem. These techniques balance the data by generating new data until the number of shows matches the number of no-shows. Using a sampling technique allowed the prediction algorithms to find a much larger number of no-shows but at the cost of being less precise. There is always a trade-off between recall and precision, the higher recall there is the more no-shows are found, but also the number of false positives increases and consequently the precision decreases. Whether more precision is required or more recall will depend upon the clinic or hospital strategy. Some hospitals and clinics will want to keep waiting time to a minimum and favour precision, while others with less volume of patients might prefer higher recall. Having a confirmation strategy will also be very important to reduce many of these false positives.

In the case of the prediction algorithms four were tested, Artificial Neural Network, Gradient Boosting, Logistic Regression and Random Forest. All the prediction algorithms achieved different results some favouring more precision, like Gradient Boosting and Random Forest and the other two more recall. No

prediction algorithm can be considered better at this stage, more tests will be required, but the prediction algorithm that achieved better results consistently was Gradient Boosting.

The results obtained are far from ideal and more features will be required to make these predictions better. Many no-shows can already be found but at a cost of some mistakes. We conclude that these predictions can help but are not still strong enough as a standalone strategy and should be combined with other scheduling strategies like patient confirmation.

.

8.3 Limitations and Future Work

The major limitation found in this thesis is related to the number of features and quantity of data available. With more data from different places and new features, more questions could be answered about how the location of the healthcare providers and the size of the clinics and hospitals affect the quality of predictions. Also, some prediction algorithms like Artificial Neural Network, benefit from large amounts of data. Another question that cannot be answered at this time is how a confirmation strategy can be combined with the prediction model to get better predictions. It will also be important to test the data in the MedClick API once it is populated and in a more advanced stage.

In the future, many more datasets should be tested to validate the prediction model. There should also be an attempt to find more features than the ones currently used, as this is the only way to improve the results obtained by the prediction algorithms. This will be especially important at a later stage when data from the API is more composed with a large number of clinics and hospitals. At this stage, patients would have registered in the MedClick platform and much more information can be retrieved to be used as a feature. Also having access to numerous clinics and hospitals can help create more powerful features, like the total number of no-shows, when was the patient last appointment, if the patient has another appointment scheduled for other healthcare providers, his health condition, what type of treatments is he undergoing and many others. With larger datasets composed of these features, much stronger predictions could be made.

Another important thing is to test the prediction model on real-time, to find out if the no-show probabilities obtained are realistic or there are many mistakes. Beyond this, a confirmation strategy should also be tested, to check if patients still miss the appointment after confirming it, and whether they show when they do not answer the confirmation. Afterwards, this can be combined with the probabilities to help remove false positives and validate the predictions.

In the end, there is still much work to be done until an efficient strategy to reduce no-shows, combined with the current machine learning developments, can be applied in the real world. The results obtained are still promising and it will be very important for the future of the healthcare industry to be able to, at

least, attenuate this no-show problem.

Bibliography

[1] D.Sousa. “Medclick: Last minute medical appointments no-show”, Master’s thesis, Instituto Superior Técnico, Lisbon, 2017.

[2] I.Ferreira. “Medclick: Last minute medical appointments no-show management”, Master’s thesis, Instituto Superior Técnico, Lisbon, 2019.

[3] A. Turkcan, L. Nuti, P.-C. DeLaurentis, Z. Tian, J. Daggy, L. Zhang, M. Lawley, and L. Sands. “Noshow modeling for adult ambulatory clinics”, in Handbook of Healthcare Operations Management, 01 2013, pp. 251–288.

[4] A. Alaeddini, K. Yang, P. Reeves, and C. K. Reddy. “A hybrid prediction model for no-shows and cancellations of outpatient appointments,” IIE Transactions on Healthcare Systems Engineering, vol. 5, no. 1, pp. 14–32, 2015.

[5] C. Elvira, A. Ochoa, J. C. Gonzálvez, and F. Mochón. “Machine-Learning-Based No Show Prediction in Outpatient Visits”, International Journal of Interactive Multimedia and Artificial Intelligence, Vol. 4, Nº7, 2017.

[6] M. M. Rinder. “An Integrated Decision-Support Tool to Forecast and Schedule No-Show Appointments in Healthcare”, Dissertation, Russ College of Engineering and Technology, 2012.

[7] R. D. Neal, M. Hussain-Gambles, V. L. Allgar, D. A. Lawlor and O. Dempsey. “Reasons for and consequences of missed appointments in general practice in the UK: questionnaire survey and prospective review of medical records.”, 2005.

[8] K. Leong, W. Chen, K. Leong, I. Mastura, O. Mimi, M. Sheikh, A. Zailinawati, C. Ng, K. Phua and C. Teng. “The use of text messaging to improve attendance in primary care: a randomized controlled trial”, 2006.

[9] S.-M. Liew, S. Tong, V. Lee, C. Ng, K. Leong and C. Teng. “Text messaging reminders to reduce non-attendance in chronic disease follow-up: a clinical trial,” 2009.

[10] J. Daggy, M. Lawley, D. Willis, D. Thayer, C. Suelzer, P.-C.DeLaurentis, A. Turkcan, S. Chakraborty and L. Sands. “Using no-show modeling to improve clinic performance,” 2010.

[11] A. George and G. Rubin: “Non-attendance in general practice: a systematic review and its implications for access to primary health care,” 2003.

[12] Y. Huang and D. Hanauer. “Patient no-show predictive model development using multiple data sources for an effective overbooking approach,” Applied clinical informatics, vol. 5, pp. 836–60, 2014.

[13] Leila F. Dantas, Silvio Hamacher, Fernando L. Cyrino Oliveira, Simone D. J. Barbosa, Fábio Viegas. “Predicting Patient No-show Behavior: a Study in a Bariatric Clinic,” Obesity Surgery, 2018.

[14] LaGanga LR, Lawrence SR. “Clinic overbooking to improve patient access and increase provider productivity,” Decis Sci. 38(2): 251–76, 2007.

[15] Parikh A, Gupta K, Wilson AC, et al. “The effectiveness of outpatient appointment reminder systems in reducing no-show rates,” Am J Med. 123(6):542–8, 2010.

[16] Samorani M, LaGanga LR. “Outpatient appointment scheduling given individual day-dependent no-show predictions,” Eur J Oper Res. 240(1):245–57, 2015.

[17] Dantas LF, Fleck JL, Oliveira FLC, Hamacher S. “No-shows in appointment scheduling–a systematic literature review,” Health Policy. 122(4):412–21, 2018.

[18] Lee VJ, Earnest A, Chen MI, et al. “Predictors of failed attendance in a multi-specialty outpatient centre using eletronic databases,” BMC Health Serv Res. 5(1):51, 2005.

[19] Bennett KJ, Baxley EG. “The effect of a carve-out advanced access scheduling system on no-show rates,” Fam Med. ;41(1):51–6, 2009.

[20] Norris JB, Kumar C, Chand S, et al. “An empirical investigation into factors affecting patient cancellations and no-shows at outpatient clinics,” Decis Support Syst. 57:428–43, 2014.

[21] Charity G. Moore, Patricia Wilson-Witherspoon, Janice C. Probst. “Time and Money: Effects of No-Shows at a Family Practice Residency Clinic,” Family medicine 33(7):522-7, 2001.

[22] Joanne Peng, Kuk Lida Lee, Gary M. Ingersoll. “An Introduction to Logistic Regression Analysis and Reporting,” The Journal of Educational Research 96(1):3-14, 2002.

[23] Michael E. Tipping. “Bayesian inference: An introduction to Principles and practice in machine learning,” Advanced Lectures on Machine Learning, pp. 41–62, 2004.

[24] Alexey Natekin, Alois Knoll. “Gradient Boosting Machines, A Tutorial,” Frontiers in Neurorobotics 7:21, 2013.

[25] Anuj Sharma, Prabin K. Panigrahi. “A Neural Network based Approach for Predicting Customer Churn in Cellular Network Services” International Journal of Computer Applications (0975 – 8887) Volume 27– No.11, 2011.

[26] Jing Li,Ji-hang Cheng, Jing-yuan Shi,Fei Huang. “Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement,” Advances in CSIE, Vol. 2, AISC 169, pp. 553–558, 2012.

[27] I. Ferreira, A. Vasconcelos. “MedClick: Last Minute Medical Appointments No-Show Management”, 12th International Joint Conference on Biomedical Engineering Systems and Technologies, HEALTHINF, Prague, Czech Republic, 2019.

[28] Gustavo E. A. P. A. Batista, Ana L. C. Bazzan, Maria Carolina Monard. “Balancing Training Data for Automated Annotation of Keywords: a Case Study”, Brazil.

[29] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, Maria Carolina Monard. “A study of the behavior of several methods for balancing machine learning training data”, ACM SIGKDD Explorations Newsletter, June, 2004, https://doi.org/10.1145/1007730.1007735.

[30] SNIRH, Sistema Nacional de Informação de Recursos Hídricos. URL: https://snirh.apambiente.pt/

[31] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324