TÉCNICO LISBOA

MedClick: Last Minute Medical Appointments No-Show Management

Inês Tormenta Pinheiro Duarte Ferreira

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. Luís Manuel Antunes Veiga
Supervisor: Prof. André Ferreira Ferrão Couto e Vasconcelos
Member of the Committee: Prof. Bruno Emanuel Da Graça Martins

May 2019

Acknowledgments

First of all, I would like to thank my supervisor, Professor André Vasconcelos for all the support and orientation provided during this year.

I would also like to thank to Rui Cruz for sharing his wisdom and useful critiques through the development process of this thesis and to my colleagues from MedClick for always being available to clear up any doubts.

I would also like to express my gratitude to MD Clínica for providing the data that was considered during the evaluation process and also to INESC-ID for their scholarship, which gave me the opportunity to attend the 12th International Conference on Health Informatics, in Prague, as a speaker.

Last but not least, a special thanks to my family for all their love, support and dedication and to my friends that helped me to overcome times of greater pressure by providing me fantastic moments of leisure and fun. This accomplishment would not have been possible without them. Thank you.

Abstract

A no-show is one of the phenomena that leads to an efficiency decrease in various sectors, including in the health care sector. When a scheduled patient misses an appointment without cancelling, it will not only waste the clinic's resources, but it will also deny medical service to another patient who could have benefited from the respective time slot. This paper describes the research that has been developed in the context of MedClick, an online platform that aims to help medical service providers increase the efficiency of their practices. The solution supports the reduction of no-shows by using supervised learning techniques to predict their occurrence and also by finding replacements to fulfill last-minute vacancy slots. The prediction is performed using a classification algorithm that computes the probability of no-show for each patient based on features that have shown to influence his decision, such as the waiting time, the day of the appointment and the number of previous no-shows. These and other features were extracted from two distinct healthcare datasets that were considered in this research. To reduce the occurrence of no-shows, the system sends reminders and then, the prediction of no-show is performed enough days before each appointment so that there is still enough time to find a replacement, if necessary. In order to select the most suitable classification algorithm to be applied in this research, a 10-fold cross validation was used to perform a comparative analysis between some of the most commonly used algorithms in this type of classification problems, in which the Gradient Boosting proved to have the best performance.

Keywords

No-show; Health Care; Supervised Learning; Classification Algorithm; 10-Fold Cross Validation.

Resumo

Um dos fenómenos que tem vindo a reduzir a eficiência dos vários setores, incluindo o setor da saúde, é a ocorrência de *no-shows*. Quando um paciente falta a uma consulta médica sem aviso prévio, não só desperdiçará os recursos da clínica como também negará serviço médico a outro paciente que poderia ter beneficiado do respectivo horário. Este artigo descreve a pesquisa que foi desenvolvida no contexto da MedClick, uma plataforma online que visa a aumentar a eficiência dos serviços médicos portugueses. A solução suporta a redução de *no-shows* através do preenchimento das vagas de última hora juntamente com a previsão da sua ocorrência, na qual foram utilizadas técnicas de aprendizagem supervisionada. Para este último passo, foi implementado um algoritmo de classificação que está a ser utilizado no sistema da MedClick para prever se o paciente irá faltar, com base em atributos que provaram ter impacto na sua decisão, como o tempo de espera, o dia da consulta e o número de *no-shows* anteriores. De forma a extrair e analisar o impacto destes e outros atributos, foram consideradas duas bases de dados distintas durante a fase de avaliação. A fim de reduzir a ocorrência de *no-shows*, o sistema enviará notificações de forma a relembrar os pacientes que tenham uma consulta agendada para breve, na qual estes devem confirmar a sua presença. A sua resposta será depois considerada durante a respectiva previsão de no-show que será feita alguns dias antes da consulta, de forma a garantir que sobra tempo suficiente para encontrar um substituto, caso necessário. O algoritmo de classificação utilizado neste projeto foi escolhido com base numa análise comparativa entre alguns dos algoritmos mais utilizados neste tipo de problemas de classificação, nos quais os dados são altamente desequilibrados. Nesta análise foi utilizada *10 fold Cross Validation*, na qual o algoritmo *Gradient Boosting* demonstrou ter os melhores resultados.

Palavras Chave

No-show; Sector da Saúde; Aprendizagem Supervisionada; Algoritmo de Classificação; Validação Cruzada.

Contents

List of Figures

List of Tables

1

Introduction

Contents

The world is going through a phase of rapidly escalating costs which implies an efficient use of resources. A no-show is one of the phenomena that leads to an efficiency decrease in various sectors, including the health care sector which is the focus of this thesis. When a scheduled patient misses an appointment without cancelling, it will not only waste the clinic's resources, but it will also deny medical service to another patient who could have benefited from the respective time slot.

Choosing the right scheduling system is vital for the health providers. Due to the large amount of missed appointments, some clinics choose to overbook time slots. Overbooking is widely used to reduce no-shows since it manages to alleviate its negative effects. However, it is an imperfect solution, as it can lead to a long waiting list and, consequently, to decreased patient satisfaction.

Hospital and clinics have been struggling to improve patient satisfaction which in addition of being an important measure of healthcare quality, is also one of the major factors leading to no-shows. In addition to patient satisfaction, there are more factors leading to no-shows that should be addressed such as scheduling problems and logistical issues.

1.1 Motivation

Many health care centers already use some strategies to reduce the occurrence of no-shows including patient education and SMS reminders [2, 3]. However, there is much more that can be done to improve the health care system.

This research was proposed by MedClick, which is an online platform that aims to help medical service providers increase the efficiency of their practices.

All the work of this thesis was developed in the context of that platform and it is focused on helping MedClick to achieve their goals by providing tools that, among other features, allow for predicting no-shows and also for fulfilling "last-minute" vacancy slots, by notifying patients whose needs and restrictions are best suited to the time slot.

The implementation of these features was rewarding because it will not only help the Portuguese health care providers but also the patients. Furthermore, there are not many systems using techniques based on machine learning to reduce no-shows so if this research proves to be a reliable solution, it may be useful for other businesses such as airline companies, restaurant sector and hospitality sector.

1.2 Objectives

The goal of this thesis is focused in one of MedClick differentiation functionalities: the reduction of no-shows from patients in medical appointments in order to increase the productivity and the resource usage in health care services. To achieve the desired goal, this research is focused on providing a

solution that must be able to:

- Minimize the occurrence of no-shows by using strategies to reduce their probability, such as reminder notifications;
- Build a supervised learning model capable of predicting no-shows based on a given set of features. For this step, several classifications algorithms must be explored in order to find the most suitable for no-shows problem;
- In the case of detecting a future no-show, the system must try to find a suitable replacement and notifying the health care provider about the change.
- Extract relevant data from an health care dataset and pre process it in order to being ready to be sent through the learning model and provide reliable predictions.

The above aims are expected to complement and improve the no-show algorithm structure that was previously implemented in the MedClick platform.

1.3 Thesis Outline

This thesis is structured as follows:

Chapter 2 includes a review of no-show literature covering the main reasons reported for missed appointments, some strategies to reduce their occurrence, some variables already considered as predictors of no-shows and finally, machine learning techniques that have already been used to predict no-shows.

Chapter 3 first provides information on the most common used supervised learning algorithms for classification problems and, after describing in detail each algorithm, section 3.1.6 summarizes and compares their advantages and disadvantages. Then, in section 3.2 are mentioned several approaches focused on validating the model performance and finally, this chapter concludes by presenting different strategies to handle imbalanced datasets.

Chapter 4 describes the no-show algorithm structure that was previously implemented by MedClick and presents their major limitations.

Chapter 5 starts by providing a detailed description of the proposed solution. Then, the solution is presented in the form of the actual implemented algorithm, including, in addition, a description of the final project structure and also a list of the most relevant tools used during the implementation process.

Chapter 6 provides the evaluation tests that were performed along with the respective results.

Finally, Chapter 7 summarizes the developed work, presents the respective conclusions and reveals the main limitations of this thesis along with suggestions for the respective improvements in future work.

2

Literature Review on No-Shows

Contents

The efficient use of resources is increasingly important, and as such, several studies have arisen, focused on detecting the origin of no-shows and finding possible solutions to this problem. These include the following: a structured and representative review of no-show literature up to 2011 [4], a research presenting a no-show algorithm implemented in the context of the MedClick application [5], and finally, a solution focused on detecting no-shows in the hospitality sector, which provides a comparative analysis between some popular classification algorithms [6].

These were the studies that most influenced this research, although there are many others that left their mark and as such, this chapter will mention some of them as it will cover the main reasons reported for no-shows, some strategies to reduce their occurrence and also, some variables already considered as predictors of no-shows.

2.1 Causes of No-Shows

Missing an appointment can be a voluntary or an involuntary act. Involuntary no-shows, as their name suggest, occur when the patient has no intention of doing so and includes one of the most commonly reported reason: forgetting the appointment [7, 8].

Several other reasons are reported for no-shows, such as scheduling problems, the patient's health status and other personal or logistical issues.

Scheduling problems may be related to the quality of the service including issues getting an appointment, wrong information about the date and time and difficulty in cancelling the appointment [9, 10].

Patient health status include being physically or mentally ill [10] and feeling better and not needing the appointment anymore [9].

Finally, the personal and logistical issues can include financial problems, lack of transportation or competing priorities, like work schedules or family problems [7, 8, 10].

2.2 Strategies to Reduce No-Shows

Healthcare providers struggle to reduce no-shows and for that they use strategies including appointment reminders, patient's education, follow-up after a no-show appointment, overbooking and open-access scheduling.

As discussed in section 2.1, forgetting the appointment is one of the most commonly reported reasons for no-shows, and as such, appointment reminders are used to prevent that situation, through text messages, phone calls or letters [2, 11]. Some health providers also focus on patient education that consists of providing all the important information in order to ensure that patients feel secure about their appointment. However, it does not result in a significant reduction of no-shows [3].

In addition to the last two strategies, some clinics use patient sanctions and methods of follow-up after a no-show, such as sending messages asking to reschedule the missed appointment, as an attempt to change patient's behavior [12].

There are some scheduling systems aimed at reducing no-shows such as overbooking and open-access scheduling. Overbooking involves scheduling more patients than the actual number that the clinic and staff can handle. This method is widely used to reduce no-shows since it manages to alleviate its negative effects. However, it's an imperfect solution, as it can lead to a long waiting list and, consequently, to decreased patient satisfaction. Open-access scheduling allows patients to see their physician within a day or two of scheduling the appointment. Unlike overbooking, this method is used to minimize waiting time [13].

Besides the above-mentioned techniques, there is a field of machine learning that has been increasingly studied, which allows the prediction of no-shows based on prior patient's behavior. In section 2.4 there will be mentioned some studies that have been exploring that field in problems related to no-shows. This field is called supervised learning, which will be further described on chapter 3.

2.3 Predictors of No-Shows

Ample literature is available discussing predictors of no-shows, which can be divided into two categories: patient's characteristics and appointment's characteristics. The first includes patient's age, gender, marital status and insurance status. The second includes waiting time, the day of the scheduled appointment and clinic's proximity. Several studies have demonstrated that no-show patients tend to be younger [14], unmarried [15], uninsured [16], with psychosocial problems [17] and finally, with prior no-show history. Regarding gender, although not a significant difference, some studies have shown that women are less likely to no-show.

Long waiting lists are one of the major problems in healthcare services and it causes patient's dissatisfaction, which in turn leads to a higher no-show rate [18]. The day, time and season of the scheduled appointment were also explored as predictor variables and it was concluded that no-shows were slightly more likely during winter [15]. Clinic's proximity is also a factor to consider since studies show that the greater the distance to the clinic, the greater the probability of no-show.

Although there are several studies related to no-shows that prove the impact of these features, it is important to bear in mind that the results may vary depending on the country where the study is done.

In the context of the MedClick application, some features were previously tested in order to find out if they affected the patient's behavior. However, after analyzing the data, only two were considered relevant (patient's age and appointment's day) since the remaining two (patient's sex and distance) did not feature major patterns [5]. In this thesis, a further analysis was performed, based on different data,

and some features proved to have a negative impact on the patient probability of no-show, as described in section 6.2.

2.4 Using Machine Learning to Predict No-Shows

Supervised Learning is a field of machine learning which has been increasingly studied and that supports the prediction of no-shows based on prior patient's behavior. Several studies have arisen focused on using this approach in different sectors and this section will cover some of those.

One of the most relevant for this project is also focused on reducing no-shows in the health care sector and it uses an hybrid approach that combines logistic regression, as a population-based method and Bayesian Inference, as an individual-based method [6]. The relevance of the previous study comes from the fact that it was the main inspiration for the previous solution implemented in MedClick [5].

There are other studies aimed at reducing no-shows in medical appointments, in which logistic regression models are commonly used to compute their predictions [19].

Besides the above-mentioned studies, there are many others focused on predicting no-shows in different sectors, such as the hospitality sector [20], in which different classification algorithms have been tested. In order to select the one that best predicts booking cancellation probabilities, the researchers performed a 10-fold cross validation for each algorithm.

Machine learning techniques have also been explored in airline companies to predict whether booked passengers will cancel or miss their flights [21].

These predictions are commonly used to optimize the overbooking in the various sectors and, as mentioned above, all of them are based on supervised learning techniques, which will be further detailed on chapter 3.

3

Supervised Learning

Contents

Supervised learning is responsible for mapping from an input to an output. The idea is to analyze a set of training data and to learn a function capable of predicting the output given new input data.

As mentioned on section 2.4, there are several studies that have arisen focused on applying supervised learning techniques in problems related to no-shows. This thesis uses that strategy to detect no-shows and to find a suitable replacement, by notifying patients whose needs and restrictions are best suited to the time slot. The effectiveness of this strategy will depend on the performance of the chosen algorithm and therefore, it is important to test and consider different approaches, which requires a wide knowledge of the available techniques.

The first section of this chapter will present some of the most widely used algorithms on classification problems, which is one of the two fields of supervised learning. Then, in order to validate the model performance, some evaluation techniques will be presented in section 3.2, and finally, in section 3.3 will be discussed several strategies to handle imbalanced datasets.

3.1 Classification Algorithms

Supervised learning problems can be further divided into regression and classification problems. A classification problem is represented by discrete output variables. In this type of problem, the function must predict the class of a given observation. Contrarily, in regression problems the output variable takes continuous values. The problem addressed in this thesis corresponds to a classification problem and, as such, this section will present some of the most widely used classification algorithms.

3.1.1 Logistic Regression

Logistic regression model is used to describe data and to measure the relationship between one dependent variable, normally binary, and one or more independent variables.

In fact, logistic regression is very similar to linear regression. The difference between the two models is that the first one predicts whether something is True or False instead of predicting something continuous and, as such, the first uses a logistic function, described above, instead of using a linear equation.

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

where:

- $x_1, \ldots, x_k$ correspond to the set of features (or predictors)
- β_0 corresponds to the intercept term
- β_k correspond to the coefficient associated to the feature x_k

This function is advantageous because, regardless of the variables it receives, the output always takes values between 0 and 1, which can be interpreted as the probability of the problem occurring.

The algorithm starts by building the model. In this phase, the training data will be used to estimate the best coefficients that will shape the logistic function to fit the given problem. The coefficients values correspond to log odd ratios which may give relevant information on the impact of each feature. Positive coefficients corresponds to higher odds of occurring no-show and negative coefficients corresponds to lower odds. A feature with a coefficient near 0 has low impact on the prediction. In order to predict the probabilities as accurately as possible is crucial to build a cost function that quantifies the error by comparing the predicted probability with the correct answer. Then, it is possible to estimate better values for the coefficients and, consequently, minimize the error, using the gradient descent.

After the model is built, the system just need to provide a set of features to predict the probabilities using the logistic function along with the previously estimated coefficients.

In addition to the output having a straightforward probabilistic interpretation, this algorithm reveals other advantages, such as the speed of prediction and the ease in adding or removing features. The prediction of the probabilities is extremely fast because after the model has been built, the system only need to apply the logistic function with the previously estimated coefficients. The model building is a computationally expensive process, but it does not have a great impact because it is only necessary at the start of the algorithm or when the model is rebuilt, which happens when features are added or removed. Finally, it is important to notice that the model requires a large dataset in order to obtain accurate and stable results.

3.1.2 k-Nearest Neighbors

The k-Nearest Neighbors (k-NN) is one of the simplest machine learning algorithms and its output depends on whether it is used for regression or classification [22], which will be our focus.

The idea of this algorithm is to predict the class of a new instance by finding the most common class among the k most similar instances (k nearest neighbors).

To find those k instances, there are several distance metrics that can be used, such as, Euclidean Distance, Hamming Distance, Manhattan Distance or Minkowski Distance [23]. The training data are represented as vectors and each of them is associated with a label that indicates the class to which it belongs. As such, the distance metric will have to calculate the distance from the new vector to every other vector in the dataset and then, select the k nearest.

In figure 3.1 it is represented an example of k-NN classification in which a new instance should be assigned to the class to which majority of the k nearest points belong. This means that if k=3, it is classified as a red triangle, however, if k=5, it is assigned to the class of blue circles.

Euclidean Distance uses the Pythagorean formula to calculate the distance between two vectors

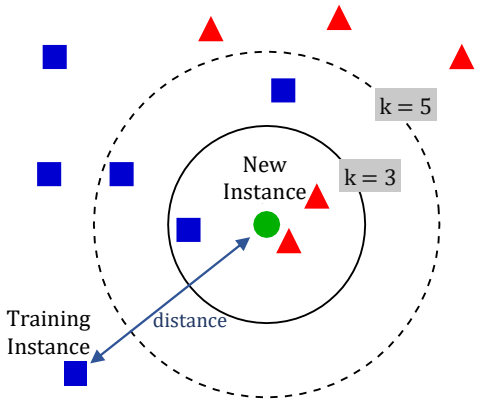

Figure 3.1: k-Nearest Neighbors Classification

and it is the most popular for continuous input variables. However, it works better when those variables are similar in type, such as widths, heights and lengths. In the case of no-show's problem, the input variables (such as gender, age and distance) are not similar in type so the recommended metric distance is Manhattan Distance, which calculates the distance between two vectors $\vec{p}$ and $\vec{q}$, using the sum of their absolute difference:

$$d(\vec{p}, \vec{q}) = \sum_{i=0}^{n} |p_i - q_i| \tag{3.2}$$

where:

- n corresponds to the number of attributes;
- p_i and q_i corresponds to the attribute i of p and q, respectively;

After finding the k nearest neighbors, we can use a common technique that consists in assigning weight to the contributions of those neighbors, so that the nearest have a greater impact [24].

The better the choice of k parameter, the greater the accuracy of the results. Choosing a small value for k results in a higher influence of the noise on the classification, however, choosing a large value can be computationally expensive and it makes boundaries between classes less distinct.

3.1.3 Bayesian Inference

The idea of Bayesian inference is to update the probability for a hypothesis, using Bayes theorem, as more data becomes available. The Bayes theorem is used to compute the posterior distribution, which is the distribution of the parameters after considering the observed data:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \tag{3.3}$$

where:

- θ represents the set of parameters and y represents the observed data;

- p(θ| y) corresponds to the posterior distribution;
- p(y| θ) corresponds to the likelihood function;
- p(θ) corresponds to the prior distribution;

As presented above, the posterior distribution is computed as a consequence of two antecedents: a likelihood function and a prior distribution, which represents the distribution of the parameters before observing any data.

This method is already implemented in MedClick application, by Daniel Sousa [5], and it is used to update the initial probability of no-show, which is computed with a population-based approach. To do this, Daniel applied the equation 3.4, which had previously been used in another study for the same purpose [6]. In this problem were considered three different events (no-show, cancellation and show-up) and as such, it was required to adapt Bayes theorem as follows:

$$E(a_k|y_1, y_2, ..., y_k) = \frac{y_k + a_k}{\sum_{k=1}^{K} y_k + \sum_{k=1}^{K} a_k} \tag{3.4}$$

where:

- K corresponds to the number of events;
- a_k corresponds to the probability of the event k;
- y_k corresponds to the number of occurrences of the event k;

3.1.4 Decision Trees

Decision trees are commonly used for predictive modeling machine learning. The goal is to use a tree to represent all the possible outcomes given a set of features. There are two types of tree models: classification trees, where the output can take a discrete set of values and regression trees, where the output can take continuous values. The topmost node of a tree corresponds to the root node, where the most important feature is placed. Each interior node represents a feature of the problem and it splits into branches, which correspond to the possible outcomes of the respective feature. At the end of each branch there is a leaf, whose value represents the respective outcome, which is used to make a prediction. For example, in figure 3.2, it is represented the tree resultant of a simple problem where we want to decide whether we should wait for a table at a restaurant or not, in a given situation. The attributes are the number of patrons in the restaurant, whether we are hungry or not, the type of restaurant and the Fri/Sat which indicates if it is Friday or Saturday. As we can see, each branch corresponds to a possible outcome of a given attribute and the leaves are labelled with yes or no, which represents whether we should wait or not, respectively.

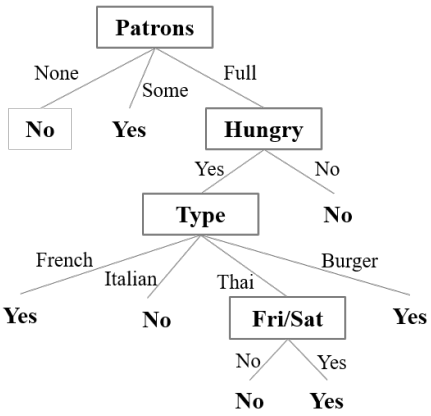

Figure 3.2: Decision Tree Example [1]

Despite being a simple model to understand, interpret and visualize, one of the main concerns when generating decision trees is that they are prone to overfitting, since these models commonly have low bias and high variance.

Another problem is that their complexity depends on the order chosen for the attributes. As such, to avoid over-complex trees, it is important to make an extra computation to decide in advance which is the best order for the attributes. This can be done by using, for example, the information gain or pruning, which is also a technique used to reduce the complexity of the decision tree by removing sections that have an insignificant impact on the classification. With this extra computation, the system can become extremely slow, depending on the complexity of the problem.

3.1.5 Ensemble Learning

Ensemble methods produce a strong learner by combining a group of weak learners [25]. Boosting and Bagging are both ensemble techniques that are commonly used in supervised learning problems.

Boosting consists on using weighted averages to produce stronger learners. New learners are built in a sequential way and at each step, the samples subset is created depending upon the performance of the previous models. The idea is to increase the weights of previously misclassified samples in order to learn the error made by prior tree.

Unlike boosting methods, which are commonly used to reduce the bias with sequential models, bagging methods are aimed at reducing the variance and each model runs in parallel and independently, with random subsets. Another difference is that in bagging methods, the final outputs are combined without preference to any model.

Random Forest and Gradient Boosting are both ensembles of decision trees. The first is an extension over bagging and the second is an extension over boosting and both will be described in detail in section 3.1.5.A and 3.1.5.B, respectively.

3.1.5.A Random Forests

Random Forests are an ensemble learning algorithm that can be used both for classification and regression problems. As previously mentioned in section 3.1.5, ensembles produce a strong learner by combining a group of weak learners. In other words, this method builds multiple decision trees, in parallel, and combines them together to obtain a more stable and accurate prediction.

Random Forests use the technique of bagging to reduce the variance without increasing the bias and, therefore, are known for solving the problem of overfitting, which is very common in decision trees [26]. This is achieved by training the individual trees on different and random subsets of data, which means that while splitting a node, this method searches for the best feature among a random subset of data, instead of searching for the most important feature.

After all the decision trees have been built individually, the model averages their results in order to obtain its final prediction.

The model performance and the predictions stability can be increased with a higher number of trees, which, in turn, may lead to a slower computation. For this and other reasons, the model hyperparameters should be properly adjusted.

3.1.5.B Gradient Boosting

Gradient boosting is an ensemble learning technique for regression and classification problems, which produces a strong prediction model by combining weak prediction models. This method is typically used with decision trees as base learners and, as mentioned on section 3.1.5, it uses an ensemble method called boosting which consists on building the trees in a sequential way and on using weighted averages to produce a strong learner [27].

At each step, the idea is to learn the error made by prior tree and for that, the model increases the weights of the samples that were misclassified in the previous step. This means that the samples subset is created depending upon the performance of the previous models.

As in random forests, the number of trees should be carefully chosen because, although a larger number of trees results in a reduced error on training set, setting it too high may lead to overfitting.

The main limitation of boosting methods is that their sequential model building leads to a longer computation time. However, this also results in better prediction since each tree corrects the classification error of the previous tree.

3.1.6 Comparative Analysis

Throughout this sub-section, some classification techniques have been described. All these algorithms are widely used in supervised learning problems, having the common goal of making its predictions

based on a set of given features. In order to compare them more easily, there is a table 3.1 at the end of this chapter that lists their main advantages and disadvantages, covering several properties such as, among others, accuracy, time complexity and space complexity.

Algorithm	Advantages	Disadvantages
Logistic Regression	The output has a straightforward probabilistic interpretation. After the model is built, it is extremely fast to predict the probabilities. Easy to add and remove features as it is only required to rebuild the model.	Building the model is computationally costly. A large dataset is required in order to obtain accurate and stable results. To maximize the accuracy, a prior feature selection must be done.
k-Nearest Neighbors	It is a very simple and intuitive classifier. No training phase is required.	The accuracy of the results can be severely degraded by the presence of noise in the training data. A good value of k parameter must be chosen in order to get good results. It can be computationally costly since it requires the computation of several distances.
Bayesian Inference	Only needs to solve a simple equation.	It requires a prior probability distribution.
Decision Trees	Easy to understand, interpret and visualize.	The complexity of the model depends on the order chosen for the attributes, which leads to the need to make a prior computation to decide which is the best sequence. The complexity increases as features are added.
Random Forests	It reduces the variance without increasing the bias, by analyzing random subsamples of data and, therefore, it reduces overfitting. Individual DTs can be trained in parallel, which results in faster computation time. It has methods for balancing error in unbalanced data sets.	RFs tend to be biased when the data contain groups of correlated features of similar relevance for the prediction [28]. A more accurate prediction requires more trees, which results in a slower model. Low level of interpretability.
Gradient Boosting	It reduces the bias of the model. Overall performance is increased since each tree corrects the classification error of the previous tree. It has methods for balancing error in unbalanced data sets.	Sequential model building results in longer computation time. Comparing to Random Forests, GBs are more sensible to overfitting. Low level of interpretability.

Table 3.1: Supervised learning algorithms comparison

3.2 Evaluation Methods

In section 3.1, it becomes clear that there is a wide range of existing machine learning algorithms. Even only focusing on classification, which is one of the two fields of supervised learning, it is possible to notice a great variety of algorithms that deal with data from different categories since they have their own style of prediction. For instance, unlike k-NN algorithm that directly creates a class output, algorithms like Logistic Regression, Random Forest and Gradient Boosting return probability outputs that are then converted into classes by using a threshold probability.

Due to these disparities between the algorithms, each particular problem must find their most suitable algorithm and for that, it can be used several approaches focused on validating the model performance.

Model validation corresponds to the technique of computing the difference between the predicted data and the actual data in our dataset. This process needs to ensure that the model will get good results dealing with unseen data and, therefore, it is not a good practice to test the model with the data that was used to train it. Considering that, the most common technique to evaluate classification problems is called Cross Validation and it will be described on section 3.2.1.

3.2.1 Cross Validation

Cross-validation is a well-known and widely used technique to estimate how accurately a predictive model will perform in practice [29]. This method ensures that the model is evaluated with unseen data and, in addition, it can be used for determining the best model hyper parameters. There are several methods based on Cross validation, which can be classified into two categories: Exhaustive and Non-exhaustive Methods.

The first learn and test on all possible ways the data can be split into training and validation sets. Leave-P-Out Cross Validation is one of the most commonly used within this category and it consists of using p observations as the validation set and the remaining observations as the training set. This is repeated for all possible combinations, in which the data can be splited with a validation set of p observations. The overall performance is estimated by averaging the results obtained in the multiple iterations.

Leave-One-Out Cross Validation is a particular case of the Leave-P-Out cross validation method, where the value of p is 1. This method is generally preferred over the previous one since it requires less computation time.

Non-exhaustive methods include techniques like Holdout and k-fold Cross Validation and they may be more appropriate because, unlike the previous methods, they do not compute all ways of splitting the data.

Holdout consists on splitting the data into two sets. The training set, which will be used to train the

model and the test set, which will be used to evaluate how well the model performs on unseen data. In cross-validation, the final performance is typically estimated by averaging the results of multiple runs. However, this method involves a single run and therefore it must be carefully used to avoid misleading results. Due to its simplicity, there are sources in which holdout is classified as a simple validation method instead of cross-validation [30].

Finally, k-Fold Cross Validation consists on partitioning the sample data into k sub-samples, one of which will be used for testing the model (validation data), and the remaining k-1 sub-samples will be used to train the model (training data). Then, the process will be repeated k times so that each fold will be used once as a test set. As already demonstrated in several studies, 10 is a typical number of folds in this type of problems since it provides good results [31].

This method is one of the most popular techniques because it is simple to understand and, compared to other methods, it requires less computation time and it also results in a less biased prediction [29].

However, none of the above-mentioned methods should be applied in time series problems, in which the primary goal is to avoid look-ahead bias. This means that the model should be capable of predicting future exclusively based on the past, which is not ensured on k-fold cross validation, for instance, since the training set may include information that occurs after the test set.

Forward Chaining Nested Cross-Validation is a possible solution for this problems since it provides unbiased estimations by ensuring independence between training and test sets. To implement this method, *TimeSeriesSplit* from *python scikit-learn* package can be used, however, it is crucial to ensure that the data is chronologically ordered.

The figure 3.3 illustrates side by side the k-fold Cross Validation and the Forward Chaining Nested Cross-Validation.

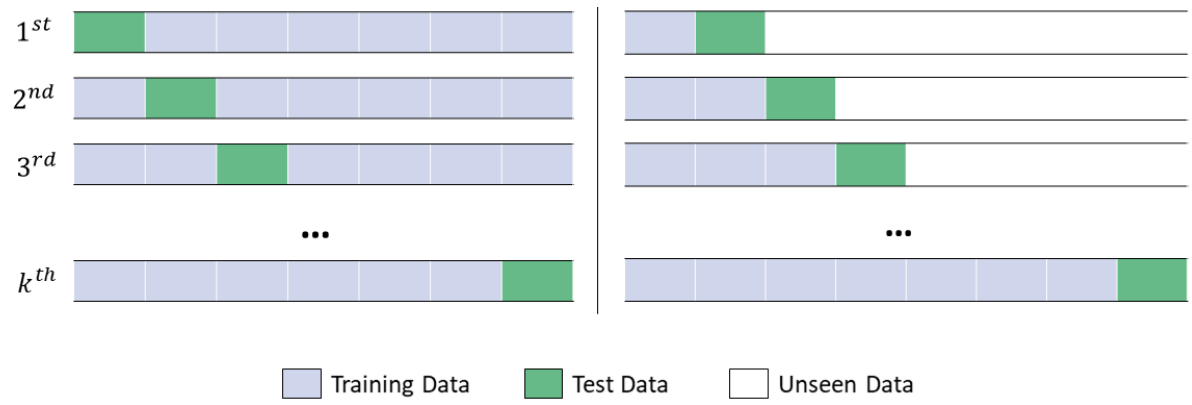

Figure 3.3: k-Fold Cross Validation (left) vs. Forward-Chaining Nested Cross Validation (right)

3.2.2 Performance Measures

After building a supervised learning model, it is crucial to use the appropriate metrics to evaluate its performance. The choice of the metrics depends on the type of problem and, consequently, on the type of model because different outputs require different approaches.

Regression models have continuous output so the most common metrics are Mean Squared Error, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and R Squared (R^2).

In this research, the goal is to predict weather a patient is going to miss the appointment, which means that the output will correspond to a label (show or no-show). As such, this section will mainly focus on classification models, in which the output is always nominal or binary.

The most common metrics used in this type of problems will be presented below but first it is important to introduce the confusion matrix concept, which, as the name suggests, it is very useful to see if the system is confusing the classes. In this metric, the predicted classes are described as Positive or Negative while actual classes are described as True or False. The instances of each predicted class are represented in the rows of the matrix while the instances of the actual class are represented in the columns. The following table represents a confusion matrix for a binary classification problem containing the values for all classes combined.

		Actual	
		Yes	No
Predicted	Yes	TP	FP
	No	FN	TN

Table 3.2: Confusion matrix for binary classification

Where:

- TP (True Positive) corresponds to the number of instances that were correctly assigned to the positive class;
- TN (True Negative) corresponds to the number of instances that were correctly assigned to the negative class;
- FP (False Positive) corresponds to the number of instances that were incorrectly assigned to the positive class;
- FN (False Negative) corresponds to the number of instances that were incorrectly assigned to the negative class;

The above-mentioned concepts are fundamental in the evaluation stage since they are used to calculate the following metrics [32]:

Accuracy: the fraction of correct predictions made by the model. This is a good measure when the target variable classes are nearly balanced. However, considering a dataset in which 90% of the data belongs to one class, it is easy to create a classification model that gets an accuracy of 90% by simply assigning all data to the majority class. For this reason, accuracy it is not a reliable metric when dealing with imbalanced classes. In such cases, it is recommended to use other metrics, such as precision or recall.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.5}$$

Precision: it measures the classifier exactness by computing the fraction of instances that are actually positive among all instances that were classified as Positive. This metric is one of the most suitable for imbalanced data since it is exclusively focused on the positive class, which typically corresponds to the minority class.

$$Precision = \frac{TP}{TP + FP} \tag{3.6}$$

Recall: it is also called sensitivity and it measures the classifier completeness by computing the fraction of instances correctly classified as Positive among all actually positive instances. Like precision, this metric is also recommended when dealing with imbalanced data since it is also focused on the positive class, which typically corresponds to the minority class.

$$Recall = \frac{TP}{TP + FN} \tag{3.7}$$

F1-score: it is also called as F-score or F-measure and it is used to find a balance between precision and recall. When a model gets high precision and low recall means that all instances classified as Positive were correct, however there are several remaining Positive instances that were classified as Negative. In the other hand, when it is obtained low precision and high recall it means that the model detects correctly each actual Positive instance, but also classifies several Negative instances as Positive. So, ideally, the model should use F-measure to get the harmonic average of precision and recall:

$$F1 - score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{3.8}$$

AUC - ROC: it means *Area under the ROC Curve* and, as the name suggests, it measures the area underneath the entire ROC curve. ROC stands for *Receiver Operating Characteristic* and it is a graph that plots False Positive Rate (equation 3.9) against True Positive Rate, which is also known as Recall (equation 3.7).

$$FPR = \frac{FP}{FP + TN} \tag{3.9}$$

The ROC curve, illustrated in Figure 3.4, shows the performance of a classification model at all classification thresholds. The AUC score corresponds to the model's ability to distinguish the two classes and it ranges from 0 to 1 [33]. Typically, the higher the score, the better the model. However, it should not be used with imbalanced data.

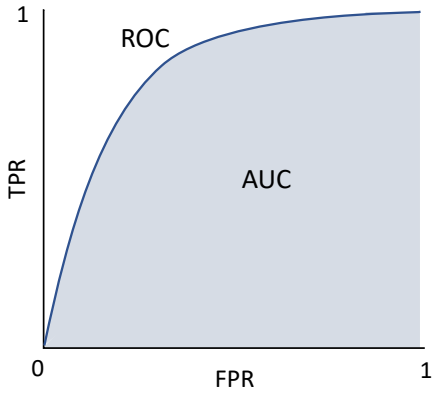

Figure 3.4: AUC-ROC Curve

Matthews correlation coefficient: it is very suitable for imbalanced datasets and it represents the correlation coefficient between the observed and predicted classifications, in which a result of 1 corresponds to a perfect prediction.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.10}$$

3.3 Learning from Imbalanced Data

Dealing with highly imbalanced data is one of the biggest challenges in data science. Several techniques have been tested and proposed to solve this problem that is very common in real world applications, such as, fraud detection, medical diagnosis, spam detection and also, no-shows prediction, which is the focus of this thesis.

As mentioned in section 3.2.2, it is crucial to select the right performance metrics when working with

imbalanced data. Accuracy, for example, is an misleading performance measure if there is a negative majority class highly dominating over a positive minority class because it can lead classifiers to assign all test data to majority class. In such cases, precision and recall are a more suitable metric since they are more focused in the positive class than in the negative class.

The ROC curve is neither a reliable measure when dealing with imbalanced data because the false positive rate (FPR) does not decrease drastically when the total real negatives is huge. Alternatively, it can be used precision-recall curve, which is a better solution for this type of problems [34].

Besides choosing the right performance measures, there are other strategies that can be used to overcome imbalanced datasets, such as cost sensitive learning and resampling techniques.

Cost sensitive learning consists on assigning cost to misclassifications while trying to minimize the overall cost. This is not a straightforward technique since misclassification errors may have different repercussions depending on the problem. Typically, the minority class corresponds to the positive class and the most expensive misclassifications are usually when an actual positive instance is labeled as negative. Considering the different possible classifications, it is built a cost matrix with the respective misclassification costs [35].

Resampling is another strategy for imbalanced problems and it is commonly used to improve the performance of the model by balancing the data. This approach includes undersampling and oversampling techniques. Both aim to adjust the class distribution of the dataset. In short, the first consists on removing instances from the majority class and the second consists on adding instances to minority class. Undersampling is less frequent and it is mainly used when dealing with large datasets. Conversely, oversampling is recommended when the amount of data is insufficient and it is typically preferable since it avoids loss of important information. One of the most common techniques based on this approach is called Synthetic Minority Over-sampling Technique (SMOTE) [36] and it consists on oversampling the minority class by using the feature space to draw lines between the k minority class nearest neighbors and then it randomly generates synthetic samples along those lines. This technique requires an extra attention when splitting the data into training and test sets, in which the goal is to create two disjoint sets to prevent the model of being trained based on the data that will be used to test it. However, applying an oversampling technique before splitting the data will lead to data leakage since data from the test set will be considered when creating new instances which means that information from test set may be leaked into the training data. Even after splitting the data, another thing that must be taken into consideration to ensure the reliability of the results, is that the SMOTE should not be applied in the test set, since the model should be tested with data as similar as possible to the real world data (i.e., imbalanced).

4

MedClick Previous Solution

Contents

A no-show algorithm structure was already implemented in the context of MedClick platform, by Daniel Sousa [5]. This section presents that solution, describing the chosen approach and the implemented algorithm. At the end, the major limitations are also revealed.

4.1 Hybrid Approach

In order to compute the no-show probability for each patient, MedClick was using a hybrid approach that combines logistic regression, as a population-based method, and Bayesian inference, as an individual-based method [5, 6].

Logistic regression was being responsible for building a model that computes an initial estimation of the no-show probability for each patient, based on the general behavior of the population. To train this model, a dataset with all the appointment data available was used to extract some previously studied features, which revealed a significant impact on the no-show's studies, such as sex, age, distance and day of the appointment.

After that, the algorithm was using Bayesian Inference to adapt the initial probability to each specific patient using their appointment record, if any. A simple query on the database was being used to get all the appointments made by the specific patient and then, the system counts his total number of appointments and his number of no-shows. This information was required to apply the equation 3.4 which outputs the final no-show probability for that patient.

4.2 Algorithm

The algorithm that was previously implemented on MedClick aimed to find patients interested in filling the "last-minute" vacancy slots and it starts when the system receives a no-show notification. This may arise for three different reasons, namely, the patient canceled his appointment, the patient failed to respond to the appointment confirmation or the system detected, using patient's location, that the patient will not arrive on time.

The first step is to obtain the filtered list of candidate patients, from which two sub-lists are highlighted. The first one includes all patients that have already scheduled an appointment at a later date in the same health care center and with the same health professional. The second list includes all patients within a certain distance from the health care center. These two lists will be considered separately and, as such, the patients of the second list will only be notified after all patients of the first list have been notified. Within each list, the patients will also be considered individually and consecutively, going from the least likely to miss the appointment to the one with the greatest probability of missing it. This requires a prior computation of the no-show probability associated with each candidate patient, using the approach

described in section 4.1.

After the patients have been ordered accordingly to their no-show probabilities, the algorithm goes into a loop until it finds an appropriate replacement or until there are no more candidate patients. A notification is always sent to the first person on the list, which corresponds to the patient with the lowest no-show probability, and the algorithm only moves on to the next patient if the previous one rejects or if they do not respond within 12 hours. At the end, if no replacement was found, the system notifies the health care center that the algorithm was unable to fulfill the time-slot.

4.3 Limitations

Despite the satisfactory results, there are some aspects that should be considered in order to improve the quality of the system.

One of the major limitations of this solution is that it is focused exclusively on finding patients interested in filling the ”last-minute” vacancy slots and as such, the algorithm that estimates the no-show probabilities is only used to sort the candidate patients list, from the least likely to miss the appointment to the one with the greatest probability of missing it. Instead of looking for a solution that is applied only after a no-show has been detected, the algorithm should be leveraged to detect no-shows, by predicting the time slots where the patient is most likely to miss the appointment. If the system was able to make that prediction, the health care center would be able to overbook another patient in that time-slot, reducing the occurrence of a no-show.

Another problem that could have negative repercussions not only for the platform but also for the health care center is the method used to find a replacement which consists of sending numerous notifications to patients that may not be interested.

Regarding the features considered by the learning model, the selection was made based on foreign studies since the company could not provide in time real world data. As such, it was not possible to determine which features are best suited to the Portuguese population. After analyzing the data, only two features were considered relevant (patient’s age and the day of the appointment) since the remaining two (patient’s sex and distance) did not feature major patterns.

Finally, the system considers neither possible changes in patient’s behavior over time nor the no-show rate of the clinic, which is important since a high no-show rate may be associated with a lack of quality in the service, which in turn may lead to patient’s no-show.

5

No-Show Management Approach

Contents

This chapter starts by presenting an overview of the proposed and implemented solution, in section 5.1, which aims at improving the previous algorithm implemented in the MedClick application.

In order to overcome the major limitations mentioned in section 4.3, this solution improves the existing no-shows learning algorithm which is leveraged to predict when and if a patient will miss the appointment. A detailed description of the solution implementation is provided in section 5.2, in which the technical aspects are also presented.

As mentioned in section 4.3, in the previous solution only two features were considered relevant (patient's age and the day of the appointment) since the remaining two (patient's sex and distance) did not feature major patterns. As such, in order to improve the performance of the learning algorithm, this solution will extract some new features, as detailed on section 5.3, in order to analyze their impact on the no-shows problem.

5.1 Solution Overview

This section provides a detailed description on how the system will work for each appointment. The flow diagram in figure 5.1 represents an overview of the implemented solution. Red and yellow boxes were used in the diagrams of this section in order to clarify what was added from scratch and what was changed in the system, respectively.

The system starts by sending a notification 5 days before each appointment, in which the respective patient must confirm their presence (step 1).

Then, 3 days before the appointment, it will be predicted whether the patient is going to miss the appointment (step 2) and, finally, in the case of detecting a no-show, the system will try to find another patient interested in scheduling an appointment for that time slot (step 3).

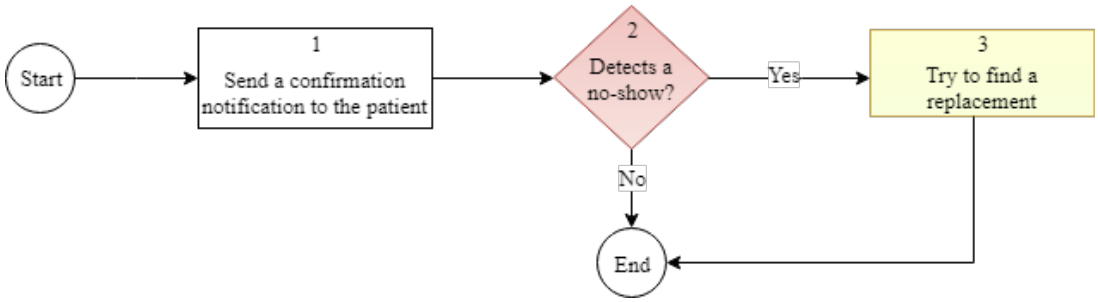

Figure 5.1: Solution Overview - Red and yellow boxes represent what was added from scratch and what was changed in the system, respectively

This chapter is divided into two sections. Section 5.1.1 describes, in detail, how the system will detect no-shows (step 2) and section 5.1.2 describes the changes that will be made to the solution proposed by Daniel for finding replacements (step 3), which is described in chapter 4.

5.1.1 Detecting a No-Show

The flow diagram in figure 5.2 represents how the system will act in order to detect no-shows, which is the main addition of this thesis.

By default, the system will run this algorithm 3 days before each appointment so that there is still enough time to find a replacement, if necessary, using the algorithm described in section 5.1.2.

The sooner the no-show is detected, the longer the system will have to find a replacement, however, the prediction may be less accurate. As such, this process may be anticipated in the future depending on the priorities of the hosting clinic, since it may lead to a higher risk of overbooking. Following, there is a detailed description of each individual step.

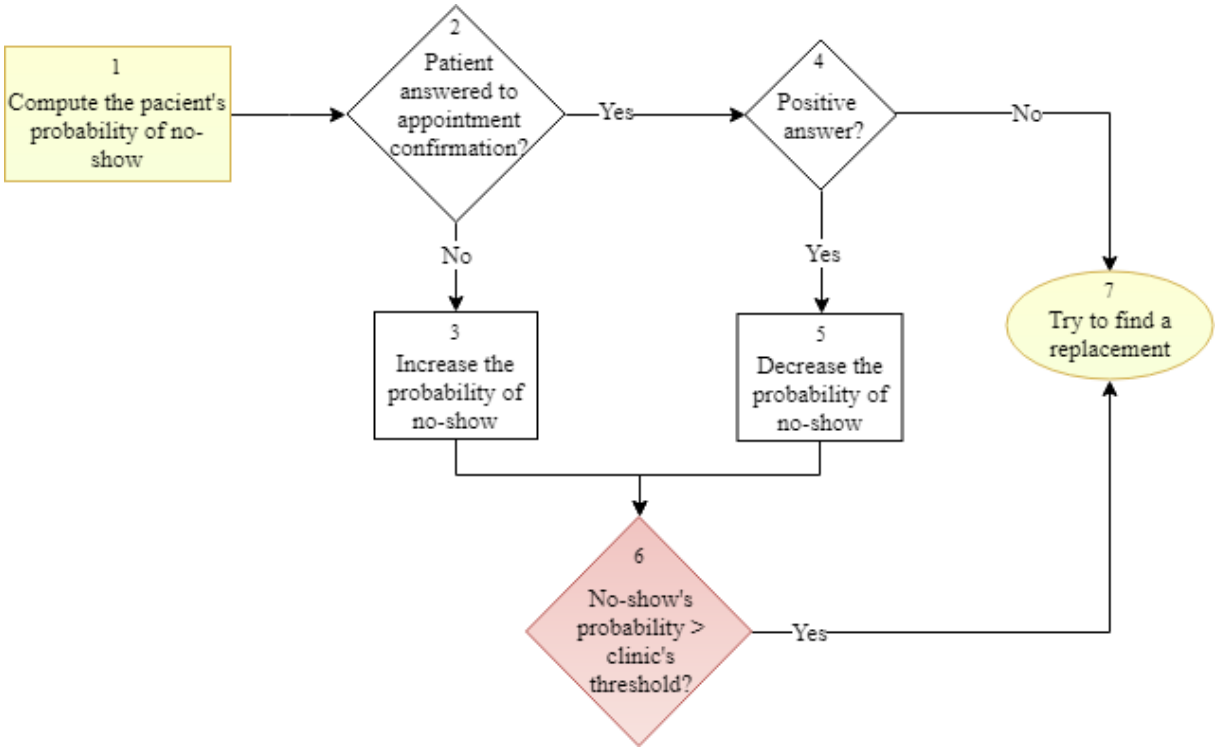

Figure 5.2: Detecting a No-show Flow Chart - Red and yellow boxes represent what was added from scratch and what was changed in the system, respectively

1. **Compute the patient's probability of no-show:** The algorithm uses a supervised learning model to perform this computation. This requires access to the database in order to extract information about the appointment and to send it through the model, which will subsequently output the respective no-show probability. The required information may include patient's age, gender, waiting time, day of the appointment, patient's history, among other features that proved to influence the patient's decision. The default learning model is Gradient Boosting and it was chosen based on a comparative analysis in which four classification algorithms were tested (see section 6.4).

2. , 3. **Patient answered to appointment confirmation?:** 5 days before the scheduled appointment, by default, the patient receives a notification to confirm their presence. In order to complete the computation of the no-show probability, the system will act according to the patient's answer. First, the algorithm checks whether the notification has been answered. If not, the algorithm proceeds to step 3 and the patient's probability of no-show will increase. Otherwise, it proceeds to step 4.

4. , 5. **Positive answer?:** After verifying that the patient answered to the notification, the question remains as to whether the answer was positive or not. If the patient confirmed their presence, the algorithm proceeds to step 5, where the system will greatly decrease the respective probability of no-show. Otherwise, there is no further need to calculate the no-show probability and as such, the algorithm proceeds directly to step 7.

6. **No-show's probability $>$ clinic's threshold?:** Once the computation of no-show's probability is completed, the system compares the result to the clinic's threshold, which corresponds to the maximum acceptable no-show's probability for the clinic. When the result exceeds the threshold, the clinic assumes that the patient will miss the appointment and therefore the algorithm proceeds to step 7. The threshold value is previously defined and it will depend on the priorities of the hosting clinic (see section 6.3). A smaller threshold will reduce the occurrence of last minute vacancies but it may lead to a higher risk of overbooking, which in turn, may lead to longer waiting times and, consequently, to a decreased patient satisfaction.

7. **Try to find a replacement:** If the patient's probability of no-show exceeds the clinic's threshold, the system will try to find another patient interested in scheduling an appointment for that time slot. This process is described in detail in section 5.1.2. However, there is a possibility of the system making a wrong prediction which will result in a longer waiting list since both scheduled patients will show up for the same time slot.

5.1.2 Finding a Replacement

As discussed in Chapter 4, an algorithm has already been implemented in MedClick application to find patients interested in filling the "last-minute" vacancy slots. That algorithm was slightly changed in order to improve the quality and efficiency of the system.

Next, the flow chart of this solution will be presented, in figure 5.3, along with a list that provides a detailed description of each step.

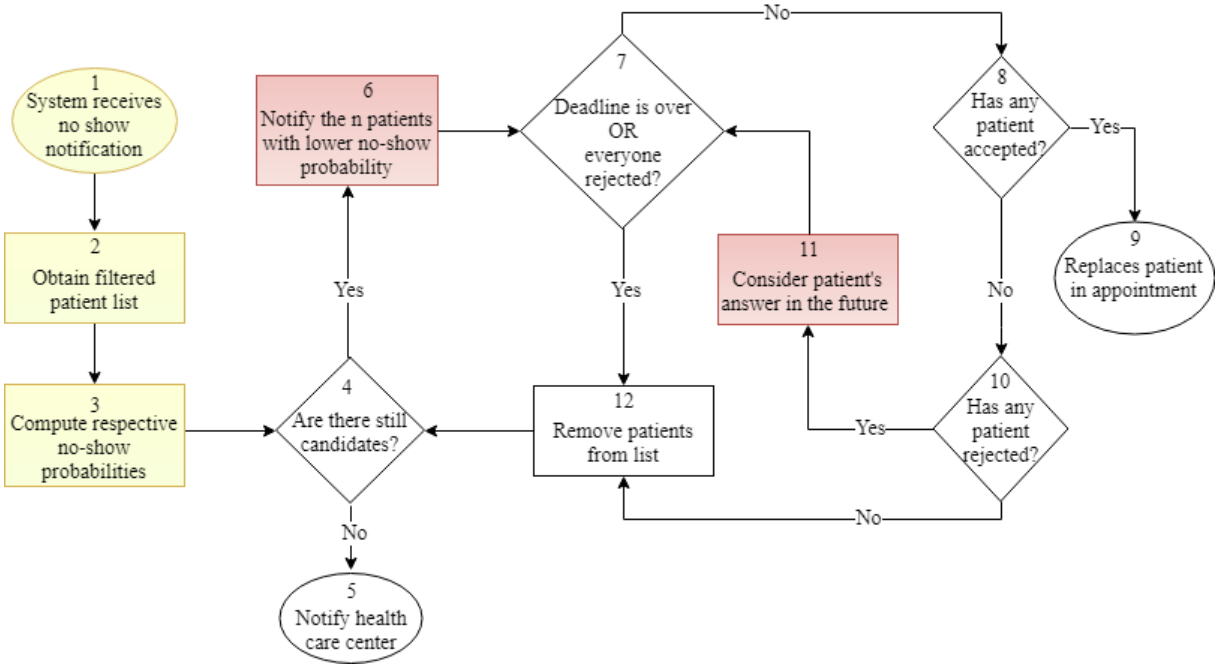

Figure 5.3: Finding a Replacement Flow Chart - Red and yellow boxes represent what was added from scratch and what was changed in the system, respectively

1. **System receives no-show notification:** With this solution, the algorithm will not only start when it receives a cancellation notification but also when the system detects that the patient will not show up for the appointment, by applying the algorithm described in section 5.1.1.

2. **Obtain filtered patient list:** The list of candidate patients consists of two sub-lists. The first one requires a new feature in the application for allowing patients to add themselves in waiting lists. All appointments will have associated a list of patients interested in filling last minute vacancies, which will correspond to the priority sub-list of this algorithm. The second sub-list includes all patients who are not on the waiting list but who have already scheduled an appointment at a later date in the same health care center and with the same health professional.

3. **Compute no-show probabilities:** The model used for computing the no-show probability of each candidate is exactly the same as the one used in the algorithm described in step 1 of section 5.1.1. This computation aims at improving the efficiency of finding replacements because, in the second sub-list, the candidates will be notified from the least likely to miss the appointment to the one with the greatest probability of missing it.

4. , 5. **Are there still candidates?:** In this step, the algorithm will enter into a loop until it finds an appropriate replacement or until there are no more candidate patients. If there are no more candidate patients to notify, the algorithm proceeds to step 5 and the health care center will be

notified that the system was unable to fulfill the time-slot. Otherwise, it proceeds to step 6.

6. **Notify the n patients with lower no-show probability:** Contrary to what was previously implemented in the MedClick application, this solution notifies more than one candidate at a time in order to optimize the remaining time. The default value of n is 2 but it may be customized later according to the preference or profile of each clinic.

7. **Deadline is over OR everyone rejected?:** After notifying the n patients with lower probability of no-show, the algorithm goes into a new loop and waits until one of the patients accepts. The loop lasts for a maximum of 12 hours but may end earlier if all patients respond negatively. In that case, the algorithm proceeds to step 12. Otherwise, it will loop through steps 7, 8, 10 and 11. As mentioned, the default value for the deadline is 12 hours but it can be changed in the future.

8. , 9. **Has any patient accepted?:** Once one of the patients accepts, the algorithm exits the loop and proceeds to step 9, in which the system is responsible for updating the appointment information, for notifying the clinic of the replacement and finally, for informing previously notified patients that the proposed time-slot is no longer available.

10. , 11. **Consider patient's rejection in the future:** This step is applied exclusively to patients on the second sub-list. The system should consider which patients were not interested in anticipating their scheduled appointment so that in the future the algorithm gives opportunity to other patients who may be more interested. In addition, if the patient has already rejected an similar opportunity more than once, the system should ask if they want to continue receiving such suggestions.

12. **Remove patients from list:** If patients rejected or failed to respond to the notification, they will not take the appointment and therefore the system removes them from the list of candidates.

5.2 Technical Approach

In the previous section, the proposed solution was presented in two parts in order to cover the way in which the two main goals of this thesis will be achieved, namely: the prediction of no-shows and the fulfilling of last minute vacancies by finding suitable replacements.

This section will consolidate both solutions and merge them into a single algorithm, which corresponds to the one implemented in the no-show module of MedClick application. A BPMN diagram was used to represent the workflow of that mentioned algorithm, as shown in Figure 5.4, in which two distinct tasks stand out.

The one that is represented at the upper part of the Figure corresponds to the starting point of the module and it consists of training the supervised learning model using the available data. The raw data

can not be sent through a model without first being pre-processed since it is often incomplete and it is likely to contain noisy and unreliable data. Therefore, the data is first pre-processed and then it is used to train the chosen model, which will subsequently be persisted into the data source in order to be used whenever the system needs to compute a no-show prediction.

In the lower part of the figure it is represented the task that will be performed daily (e.g., everyday at 8am), which consists of two sub-tasks. One is responsible for sending reminders to the patients with appointments 5 days from now and the other one will compute the probability of no-shows on the appointments 3 days from now, using the model that was previously trained. In the case of detecting a no-show, the system is responsible for notifying possible replacements.

All these steps are further detailed in section 5.1 so it is important to notice that this section is exclusively focused on presenting the way in which both solutions were merged into a single algorithm.

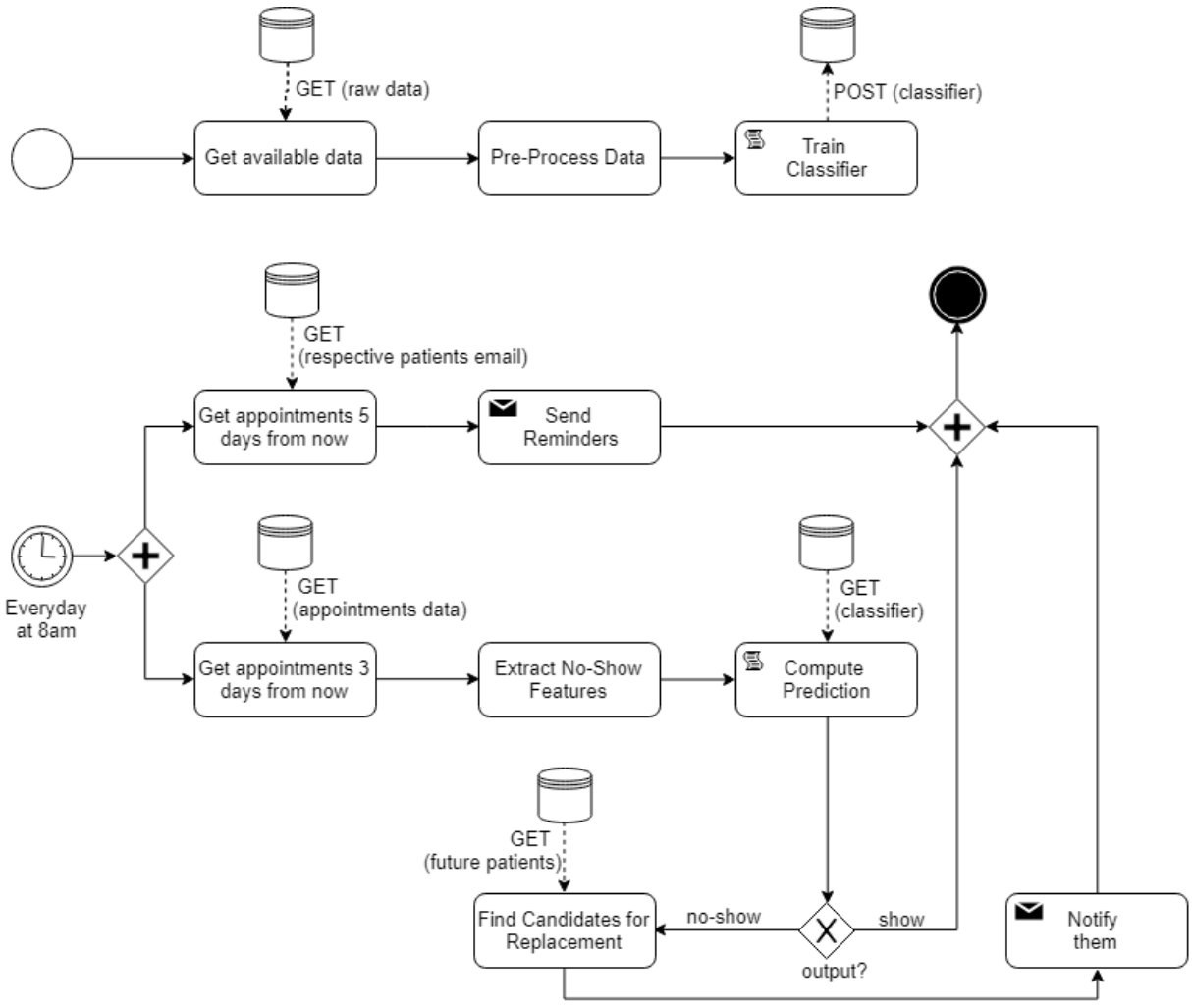

Figure 5.4: Implementation of No-Show Module

Both learning and prediction may use online or offline approaches. When machine learning techniques are embed into applications through web services, as in this project, predictions are usually computed in real-time using fresh new input data, which typically comes in the form of a REST call. This means that is being used an online prediction.

Regarding the learning, the model is typically trained once on historical data, remaining constant after being deployed to production. This means that is being used an offline learning. However, it is important to ensure that the model does not become unstable, which may happen very often. In that case, the model should be re-trained. Using an online learning requires to constantly update the model as new data arrives, however this approach is more appropriate for time series problem and it is not viable in this project.

In this type of problems, it should be used a batch learning technique which consists of combining both online and offline approaches [37]. With batch approach, the model is re-trained only after a certain number of observations have been inserted into the dataset.

5.2.1 Project Structure

For this thesis, a no-show module was developed primarily using NodeJS, with the exception of some machine learning tasks that were included in a python script.

This section will present the structure of this project, by presenting the following list, that provides a description of their most important components, along with the Figure 5.5 that presents the way in which they are connected:

- *app.js*: this file is the typical starting point for node applications and, in this project, it is responsible for calling the method that will train the supervised learning model for the first time based on the available data. In addition, it is responsible for running the tasks included on *scheduler.js* file, everyday at 8am.

- *api.js*: this file contains the methods that are responsible for fetching data from MedClick API

- *scheduler.js*: this file contains the tasks that will be daily performed, which are responsible for sending reminders to the patients with appointments 5 days from now and for computing the probability of no-shows on the appointments 3 days from now.

- *noShow.js*: this file contains the methods that are directly related to the no-show problem, which are responsible, for example, for getting the no-Show features of a given appointment, for finding replacements for a given vacancy, etc. Most of this methods need to extract information from the dataset so this file uses several methods from *api.js* file.

- *pythonConnector.js*: this file is responsible for running the *classifier.py* script, which allows to leverage the python's machine learning libraries within the NodeJS environment.
- *classifier.py*: this is the python script that contains the supervised learning model that is responsible for predicting the no-show based on a set of features.

Besides the above-mentioned files and the typical NodeJS configuration files, there are other less relevant files that contain helper methods related, for example, to the feature of sending emails or working with dates.

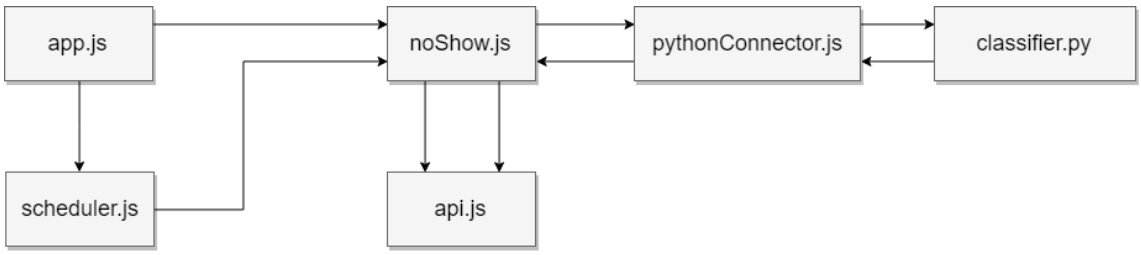

Figure 5.5: Project Structure

5.2.2 Implementation Tools

As previously mentioned, this thesis was developed in the context of MedClick platform, which is built using NodeJS. Consequently, the no-show module was predominantly built in NodeJS with the exception of classification models and their respective tests that were implemented using Python mainly due to its highly optimized libraries related to machine learning.

LoopBack is an open-source framework for Node.js which, although not considered an specific implementation tool of this thesis, it was used in MedClick to create the API and it provides an API documentation interface, called API Explorer, which was very useful during the implementation phase of this thesis since it makes API operations more intuitive.

The most relevant tools for this thesis are described below, in Table 5.1.

Node.js	*xlsx*	Parser and writer for various spreadsheet formats
	node-fetch	Allows making network requests using Promises
	node-cron	Allows scheduling tasks in Node.js using full crontab syntax
	nodemailer	Enables e-mail sending from Node.js applications
	python-shell	Provides a simple way to run Python scripts from Node.js with basic but efficient inter-process communication
Python	*pandas*	Provides tools to load and analyze data files
	matplotlib.pyplot	It can be used in Python scripts to generate 2D graphs
	sklearn.linear_model	*LogisticRegression* model implementation
	sklearn.neighbors	*KNeighborsClassifier* model implementation
	sklearn.ensemble	*RandomForestClassifier* and *GradientBoostingClassifier* model implementation
	imblearn.over_sampling	Over-sample using *SMOTE* technique
	sklearn.model_selection	Used in two different approaches: to generate cross-validated estimates using *cross_val_predict* and also to split data into random train and test subsets using *train_test_split*
	sklearn.metrics	Performance metrics computed for each model: *confusion_matrix, accuracy_score, precision_score, recall_score, f1_score*

Table 5.1: Tools used during implementation phase

5.3 Dataset

During the evaluation process of this research, two distinct datasets were considered.

The first dataset was provided by a portuguese clinic, *MD Clínica*, and the respective data was already used by Daniel Sousa [5] to test the algorithm that was previously implemented in MedClick.

The second was downloaded from *Kaggle* and it contains data related to 110k medical appointments from Brazil. Ideally, only datasets from Portugal would be used since MedClick is exclusively focused in portuguese healthcare but, unfortunately, that was not possible within the time available.

Real-world data should not be sent through a model without first being preprocessed since it is often incomplete and it is likely to contain noisy and unreliable data. Data preprocess includes data cleaning, normalization, filtering, feature extraction, and a few others [38].

Unfortunately, each dataset requires a different sequence of preprocessing techniques so it is impor-

tant to first analyze the available data. The following sections are focused on describing each considered dataset and also on presenting the pre processing techniques that were applied in order to transform the respective raw data into clean and relevant data.

In addition, an oversampling method was also applied in both datasets since the no-show's data is highly imbalanced. The implemented method is called SMOTE and it is applied exclusively in the training set in order to avoid data leakage, as explained in section 3.3.

The impact of those changes on the model performance will be detailed in section 6.2 and the log files included in the Appendix A, present all the pre-processing steps along with the respective changes on the shape of the dataset.

5.3.1 *MD Clínica* Data

The data from MD Clínica was provided in two separate worksheets, one with the personal data of each patient and the other one containing the data related to each appointment. The Table 5.2 presents the features that were provided in each worksheet along with a brief description.

	Feature	Description
Patients Worksheet (58 499 instances)	patient_id	Patient's identifier number
	postal_code	Patient's postal code
	birthday_date	Patient's birthday date
	marital_status	Patient's marital status (married, single, divorced or widowed)
	gender	M or F, depending on the patient's gender
	occupation	Patient's occupation
	insurance_id	Patient's insurance identifier
	insurance_name	Patient's insurance name
Appointments Worksheet (283 010 instances)	patient_id	Patient's identifier number
	date	Date and time of the appointment (typo: time is always T00:00:00Z)
	physician_id	Physician's identifier number
	status	Appointment's status (completed, rescheduled, canceled, reminder sent, patient gave up, etc.)

Table 5.2: Features extracted from the orginal dataset (*MD Clínica*)

5.3.1.A Pre-Processing

As mentioned on the previous section, the appointments were labeled with a feature called status whose values were very diverse. Since this project is focused on studying no-shows, only the rows containing the following values were considered:

- *"Completed"*, *"In Appointment"* and *"Patient Attended"* which represent the label *show*;
- *"Patient Missed"* and *"Patient Gave Up"* which represent the label *no-show*.

The rows whose status were different from the above mentioned values were removed.

After that, in order to combine the data from the two worksheets, a merge function was applied. This process was based on the *patient_id* column that was the only feature in common. However, the patients worksheet contained several patients that never had an appointment and the appointments worksheet contained several appointments from patients that were not registered in the patients worksheet. This means that after combining the data, several missing values have been included on the new dataset and, therefore, these rows containing patients without appointment information, and vice-versa, were also removed along with some rows whose postal code values had less than 7 digits.

After the cleaning process, the percentage of no-shows was $\sim$17% and the number of rows has reduced from 18493 to 10123, which unfortunately, represents a great loss of information.

Some features were extracted from the available raw data in order to compensate for the previous loss of information. The following table provides a brief description for each extracted feature:

Feature	Description
age	Patient's Age
weekday	1 if the day of the appointment was a weekday; 0 otherwise.
semi_postal_code	The first 4 digits of patient's *postal_code*
previous_appointments	number of patient's previous appointments. It was computed by grouping the appointments by patient ID and then using *cumcount()*.
previous_noshows	number of patient's previous no-shows. It was computed by grouping the appointments by patient ID and then using *cumsum()* to find the cumulative sum value over the *No-show* column. It is important to ensure that each instance does not consider the respective no-show value.
noshow_ratio	total number of previous no-shows divided by the total number of previous appointments
No-show	Based on *status* column, this feature takes value of 1 if the patient missed the appointment; 0 otherwise

Table 5.3: Features added to *MD Clínica* dataset

After the extraction process, the number of features has increased from 12 to 19. However, some features were then discarded, such as, the marital status, the occupation and the insurance name. The first was discarded because only 4% of patients had that field filled. The second was discarded due to their wide variety of values (even similar occupations were presented in different ways). Finally, the *insurance_name* was discarded because since the *insurance_id* was already being considered, the respective name would not give any additional information to the models. After removing these and other irrelevant columns, the number of features has reduced from 19 to 10.

5.3.2 Brazil Data

As mentioned above, this dataset contains 110527 medical appointments from Brazil, of which 22319 $(\sim 20\%)$ report the occurrence of no-show. The following table provides a brief description for each feature that was extracted from the original dataset:

Feature	Description
age	Patient's Age
gender	M or F, depending on the patient's gender
scheduled_day	Date and time the appointment was scheduled
appointment_day	Date and time of the appointment (typo: time is always T00:00:00Z)
neighborhood	Neighborhood where the appointment was hosted. They are all located in Vitoria, which is the capital of *Espírito Santo* state, in Brazil
scholarship	1 if the patient receives a scholarship; 0 otherwise
hypertension	1 if the patient has hypertension; 0 otherwise
diabetes	1 if the patient is diabetic; 0 otherwise
alcoholism	1 if the patient is alcoholic; 0 otherwise
handicap	Number of patient disabilities
SMS_received	Number of SMS sent to the patient
No-show	1 if the patient missed the appointment; 0 otherwise (Class to be predicted)

Table 5.4: Features extracted from the orginal dataset (Brazil)

5.3.2.A Pre-Processing

For this dataset, a first analysis of data was performed in which the rows containing missing values were removed due to its insignificant number. In addition, some rows were also removed for having inconsistent values, such as negative ages and scheduling days after the respective day of the appointment.

After that, the appointments were chronologically ordered and then the following features were extracted from the available data in order to provide more meaningful information to the model:

Feature	Description
scheduling_weekday	checks whether the appointment was scheduled on a weekday (1) or not (0). Note: It is not related to the actual day of the appointment day but to the scheduling day.
appointment_weekday	checks whether the day of the appointment was a weekday (1) or not (0).
waiting_time	time that the patient had to wait from the scheduling day to the actual day of the appointment.
previous_appointments	number of patient's previous appointments. It was computed by grouping the appointments by patient ID and then using *cumcount()*.
previous_noshows	number of patient's previous no-shows. It was computed by grouping the appointments by patient ID and then using *cumsum()* to find the cumulative sum value over the *No-show* column. It is important to ensure that each instance does not consider the respective no-show value.
noshow_ratio	total number of previous no-shows divided by the total number of previous appointments
number_diseases	number of patient's diseases. It is the result of combining *hypertension*, *diabetes* and *alcoholism* features.

Table 5.5: Features added to Brazil dataset

Apart from these features, *gender*, *handicap* and *age* were also considered. These additional features were created based on several studies that have already demonstrated their negative impact on patient's no-show probability, as described in section 2.3.

After cleaning the data, the total number of instances dropped from 110527 to 110466 and, although new features were created, some of them were combined and others were discarded so its total number remained the same.

6

Results

Contents

This chapter describes the evaluation process of this project, by providing detailed information about the evaluation method selection, the results at different preprocessing levels, the method of choosing the optimal threshold, the performance of different classification algorithms, and finally, a comparative analysis is provided in order to compare the solution from this research with the one that was previously implemented in MedClick system.

6.1 Comparison of Evaluation Methods

As mentioned on section 3.2 there are several methods that can be used to evaluate the performance of learning models.

Having imbalanced data or data tagged with relevant timestamps are important factors that must be considered when choosing the right method.

Regarding the problem of no-shows, it is known that there are typically more shows than no-shows and, as such, the dataset used in this project is highly imbalanced since there is a negative majority class highly dominating over a positive minority class. Considering this, measuring only the accuracy would not be sufficient to evaluate the models, so in order to get more reliable results, three additional metrics were also measured: precision, recall and f1-score.

Initially, it was discussed whether this problem would be related to time series learning or not. That doubt has arisen when features based on past information were extracted from the available data, namely the number of previous appointments and the number of previous no-shows. Their values depend on the respective timestamp and, as such, it is important to ensure that the model would not be biased if the training set contained information that occurred after the test set.

This concern can be illustrated by considering the data that is represented on table 6.1, which provides information on two appointments of a given patient.

patientID	date	...	previous_appointments	previous_noShows	noShow
1	February 3, 2018	...	0	0	Yes
1	December 17, 2018	...	1	1	Yes

Table 6.1: Example Data

Considering that the first row is part of the test data and that the second row is part of the training data, one can assume that the model easily detect that the patient missed the appointment in February, since it has access to the respective number of previous appointments and number of previous no-shows until December. However, it is important to notice that the *patientID* does not belong to the set of features, which means that the model would not realize that it was dealing with two appointments from the same patient.

In order to support the previous statement, the Brazil data was used to perform two distinct evaluation methods. The first consists of a random split between training and test data and the second method consists of a non-random split, which required the data to be ordered chronologically in order to use the first 80% of data to train the model and preserve the last 20% for testing. In either case, the models were evaluated using disjoint sets of training and test data.

The table 6.2 presents the results of each method over the different classification models.

	Evaluation Metric	Classification Algorithm			
		Logistic Regression	k-Nearest Neighbors	Random Forest	Gradient Boosting
Random Split	Confusion Matrix	$\begin{bmatrix} 2181 & 2400 \\ 2370 & 15143 \end{bmatrix}$	$\begin{bmatrix} 2654 & 3605 \\ 1897 & 13938 \end{bmatrix}$	$\begin{bmatrix} 2049 & 1712 \\ 2502 & 15831 \end{bmatrix}$	$\begin{bmatrix} 2574 & 2303 \\ 1977 & 15240 \end{bmatrix}$
	Accuracy	0.7841	0.7510	0.8093	0.8063
	Precision	0.4761	0.4240	0.5448	0.5278
	Recall	0.4792	0.5831	0.4502	0.5656
	F1-Score	0.4777	0.4910	0.4930	0.5460
Non-Random Split	Confusion Matrix	$\begin{bmatrix} 2609 & 3400 \\ 1487 & 14597 \end{bmatrix}$	$\begin{bmatrix} 2525 & 3961 \\ 1571 & 14036 \end{bmatrix}$	$\begin{bmatrix} 2085 & 1828 \\ 2011 & 16169 \end{bmatrix}$	$\begin{bmatrix} 2519 & 2252 \\ 1577 & 15745 \end{bmatrix}$
	Accuracy	0.7788	0.7496	0.8262	0.8266
	Precision	0.4342	0.3893	0.5328	0.5279
	Recall	0.6369	0.6164	0.5090	0.6149
	F1-Score	0.5163	0.4772	0.5206	0.5681

Table 6.2: Random vs. Non-Random Train/Test Split

If the considered data had any temporal dependence between the instances, then the results of random split would be better since the training set would include information that occurred after the test set. However, as expected, the results were very similar which means that the models were not affected by look-ahead bias.

K-Fold Cross Validation is commonly used in this type of problems as it is one of the most efficient methods that allows model hyper parameters optimization and it also evaluates the model performance with different subsets of data, as described in section 3.2.1.

In this thesis, 10-Fold Cross Validation was the chosen method to perform all tests and it consists on splitting the data into 10 folds, one of which is used for testing the model and the remaining 9 are used to train the model. Then, the process is repeated 10 times so that each fold will be used once as a test set.

The Table 6.5 presents the average of the final performance results and, as expected, they are also very similar to the non-random split results.

6.2 Impact of Pre-Processing Techniques

As mentioned on section 5.3, raw data should not be sent through a model without first being preprocessed since it is often incomplete and likely to contain noisy and unreliable information. Considering that, the following sequence of preprocessing techniques was applied to the initial dataset:

- Removal of instances containing missing values;
- Removal of instances with inconsistent values, such as postal codes with less than 7 digits, in the case of the MD Clinica dataset, or negative ages and scheduling days after the respective day of the appointment, in the case of the Brazil dataset;
- Using the available data to extract features that already proved their negative impact on patient's no-show probability;
- Balancing the data by applying SMOTE after splitting the data into training and test sets. Since the method of evaluation is based on cross-validation, the oversampling technique is repeated for each fold;

The progress of the performance throughout these steps is supported by the results presented in Table 6.3 and in Table 6.4, in which the several measures from *10-Fold Cross Validation* are organized based on different pre-processing levels.

Initially, each model was evaluated with raw data and, as expected, the accuracy was the only measure getting good results, due to the imbalanced data. This proves that accuracy is not a reliable measure in this type of problems since any model can get good results by simply assigning all data to the majority class (show), which happened in the logistic regression model, whose precision, recall and f1-score had values of 0.

As mentioned above, SMOTE technique was applied in order to balance the data. With this amendment, the models improved in general and their performance measures became more reliable, including the accuracy measure whose scores have decreased.

Finally, some features were extracted from the available data and others were discarded for being irrelevant to the problem. With this final step the models increased their performance, as shown on the lower part of both tables.

	Evaluation Metric	Classification Algorithm			
		Logistic Regression	k-Nearest Neighbors	Random Forest	Gradient Boosting
Raw Data	Accuracy	0.8278	0.8236	0.8160	0.8274
	Precision	0.0000	0.3211	0.2517	0.5178
	Recall	0.0000	0.0180	0.0315	0.0154
	F1-Score	0.0000	0.0336	0.0540	0.0295
Raw Data (Resampled)	Accuracy	0.5846	0.5660	0.6824	0.7155
	Precision	0.2399	0.2021	0.2205	0.2720
	Recall	0.5293	0.5959	0.3330	0.3307
	F1-Score	0.2943	0.3005	0.2639	0.2798
Processed Data	Accuracy	0.7508	0.6527	0.7601	0.7681
	Precision	0.3552	0.2510	0.3618	0.3861
	Recall	0.3925	0.5049	0.4981	0.5098
	F1-Score	0.3591	0.3334	0.4163	0.4328

Table 6.3: Performance obtained at different levels of pre-processing (MD Clínica Data)

	Evaluation Metric	Classification Algorithm			
		Logistic Regression	k-Nearest Neighbors	Random Forest	Gradient Boosting
Raw Data	Accuracy	0.7981	0.7605	0.7951	0.7981
	Precision	0.0000	0.2583	0.3356	0.5296
	Recall	0.0000	0.1024	0.0163	0.0010
	F1-Score	0.0000	0.1429	0.0309	0.0022
Raw Data (Resampled)	Accuracy	0.6246	0.5761	0.6009	0.6369
	Precision	0.2621	0.2259	0.2531	0.2709
	Recall	0.4756	0.4579	0.5099	0.4803
	F1-Score	0.2933	0.2991	0.3268	0.3222
Processed Data	Accuracy	0.7304	0.7416	0.8071	0.8091
	Precision	0.4035	0.4015	0.5210	0.5285
	Recall	0.6304	0.5793	0.4373	0.5342
	F1-Score	0.4820	0.4734	0.4732	0.5176

Table 6.4: Performance obtained at different levels of pre-processing (Brazil Data)

The features that were extracted, based on the available data, are further described in section 5.3.1.A and in section 5.3.2.A, depending on the considered dataset, and their respective importances are illustrated in the bar charts below.

In Figure 6.1, the importances of *MD Clínica* features are represented, which corresponds to their impact on predicting no-shows. In contrast to *Brazil* values, these features have shown a balanced distribution of importance.

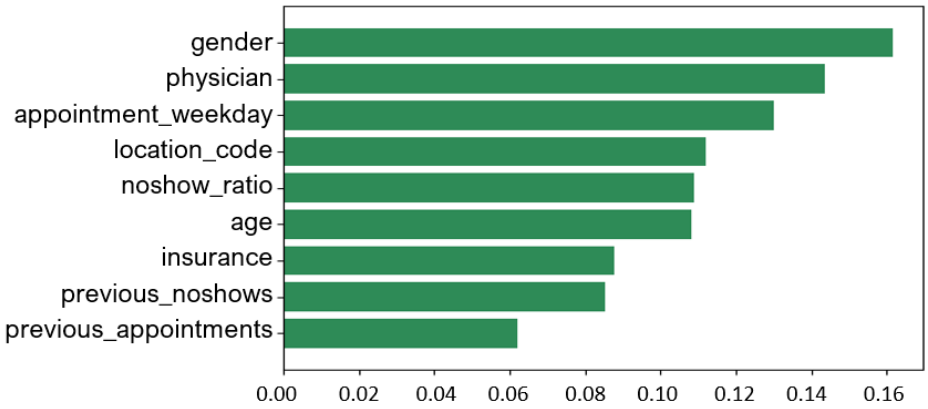

Figure 6.1: Feature Importance Bar Chart (MD Clínica Data)

The bar chart of Figure 6.2 represents the importances of Brazil features, in which the feature that proves to have the biggest impact on predicting no-shows is the time that the patient had to wait to see their physician.

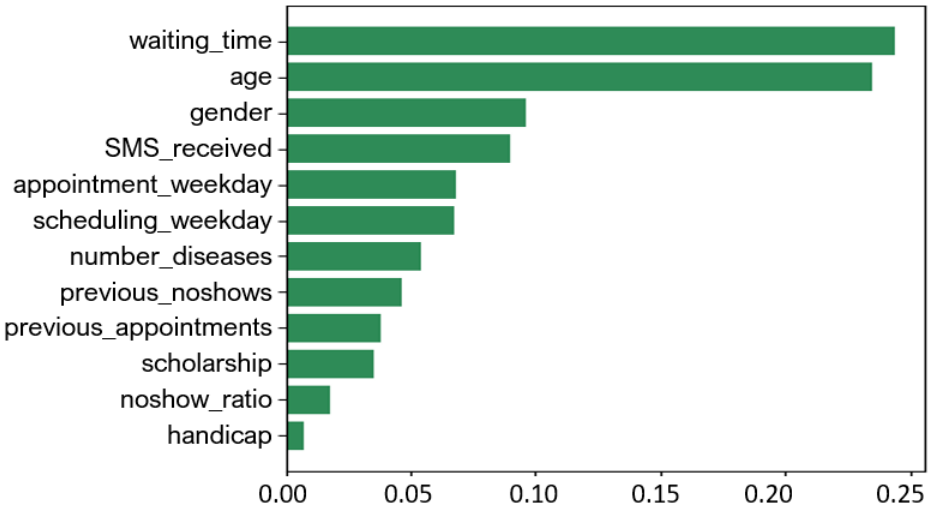

Figure 6.2: Feature Importance Bar Chart (Brazil Data)

6.3 Choosing an Optimal Threshold

With the exception of k-NN algorithm that directly creates a class output (0 or 1), the remaining models return probability outputs that are subsequently converted into classes by using a threshold probability. The default value for this threshold is 0.5, which means that a probability above that value indicates positive class and a probability below indicates negative class. However, each problem must find their optimal threshold.

In balanced classifications, the threshold is typically chosen considering the trade-off between sensitivity and specificity. Sensitivity corresponds to the proportion of actual positives that were correctly classified, while specificity corresponds to the proportion of actual negatives that were correctly classified. These two measures are inversely related which means that as the sensitivity increases, the specificity decreases and vice-verse. Decreasing the threshold probability leads to a increased sensitivity but also to a decreased specificity. To choose the optimal threshold, the *ROC curve* is typically used since it provides a graphical visualization of the relation between the sensitivity and the false positive rate (1-Specificity) [39].

However, when the data contains a negative majority class highly dominating over a positive minority class (such as in the no-show problem), the false positive rate is barely affected due to the high amount of true negatives. In these cases, the threshold must be chosen considering the precision-recall trade-off, since the precision does not depend on the number of true negatives [34]. Precision and recall are also inversely related and, as such, decreasing the threshold leads to a decreased precision but to an increased recall.

When choosing the optimal threshold for this research, it is important to consider the several clinics in which the solution will be applied since each approach may lead to different consequences. Hence, the threshold must be chosen considering three possible approaches:

- **High Precision & Low Recall:** the model is not able to detect many no-shows but it is highly trustable when it does. This means that the clinic's resources will continue to be wasted but, at least, there will be no overbooking since the system will not schedule replacement patients in slots whose original patient will not fail the appointment. This may be an advantage since it decreases the waiting lists and, consequently, does not decrease patient satisfaction.
- **Low Precision & High Recall:** most of the no-shows are detected but the model also classifies some shows as no-shows. This means that the clinic's resources will not be wasted with last-minute vacancy slots but, the system will accidentally overbook appointments since it will try to find replacements to slots in which the no-show was incorrectly detected. This may lead to long waiting lists and, consequently, to decreased patient satisfaction.
- **Precision $\simeq$ Recall:** In this case, since both have a similar formula, saying that precision is equal

to recall is the same as saying that the number of false positive (FP) is equal to the number of false negatives (FN). In other words, the number of no-shows that were incorrectly classified as show (FP) is equal to the number of shows that were incorrectly classified no-shows (FN).

Instead of using ROC curves to compare the model performance over imbalanced data, it is highly recommended to use precision-recall curve, which uses different probability thresholds to summarize the trade-off between the precision and recall [34].

In this research, the threshold for each algorithm at each dataset was chosen as a way of getting similar precision and recall, which has resulted in a threshold of 0.5 for Brazil data and a threshold of 0.3 for *MD Clínica data*. The respective precision-recall curves are presented in figure 6.3, in which having a curve above another curve means that the first has a better performance level.

As mentioned above, the value of this threshold must be adapted, in the future, according to each clinic approach.

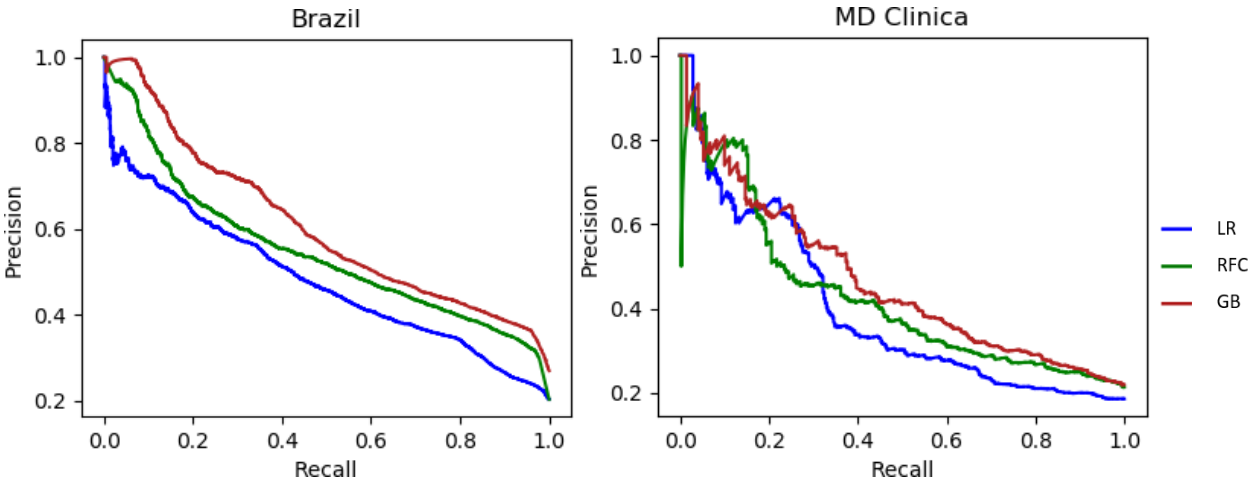

Figure 6.3: Precision-Recall Curves

6.4 Comparative Analysis of Classification Algorithms

There is a wide range of classification algorithms and each particular problem must find their most suitable algorithm since the effectiveness of the solution will depend on their performance. For this reason, it is important to test and consider different options and as such, four different algorithms were tested in this thesis, namely Logistic Regression, k-Nearest Neighbors, Random Forests and Gradient Boosting.

The evaluation method was a 10-fold Cross Validation and the following table presents the mean and the standard deviation (SD) of the results of each algorithm over the 10 iterations:

	Evaluation Metric		Classification Algorithm			
			Logistic Regression	k-Nearest Neighbors	Random Forest	Gradient Boosting
MD Clínica Data	Accuracy	Mean	0.7508	0.6527	0.7601	0.7681
		SD	0.0654	0.0297	0.0283	0.0521
	Precision	Mean	0.3552	0.2510	0.3618	0.3861
		SD	0.1483	0.0578	0.0816	0.1145
	Recall	Mean	0.3925	0.5049	0.4981	0.5098
		SD	0.0533	0.0581	0.0763	0.0803
	F1-Score	Mean	0.3591	0.3334	0.4163	0.4328
		SD	0.0924	0.0642	0.0797	0.1024
Brazil Data	Accuracy	Mean	0.7304	0.7416	0.8071	0.8091
		SD	0.0339	0.0133	0.0199	0.0119
	Precision	Mean	0.4035	0.4015	0.5210	0.5285
		SD	0.0615	0.0337	0.0546	0.0341
	Recall	Mean	0.6304	0.5793	0.4373	0.5342
		SD	0.1310	0.0675	0.0960	0.1586
	F1-Score	Mean	0.4820	0.4734	0.4732	0.5176
		SD	0.0401	0.0415	0.0797	0.0945

Table 6.5: 10-Fold Cross-Validation Results

As shown in the table 6.5, the results obtained with *MD Clínica* data proved to be consistent with the results from Brazil data.

From these results, despite the slight difference, it is clear that Gradient Boosting outperforms the remaining algorithms in each of the considered metrics.

In general, the models achieved good accuracy results but the remaining metrics showed lower

values, which might seem an indicator of a bad performance but it is important to consider that the human behavior is extremely complex, which makes it hard to predict. Also, this learning model will be running as a part of a no-show algorithm that supports others strategies aimed at reducing no-shows, such as the reminders approach.

Nevertheless, the algorithm that is preferable to implement in MedClick system is the Gradient Boosting whose recall results showed that around 50% of no-shows will be predicted, which leads to an increase in the efficiency of the clinic's resources.

Standard deviation values also shown that there was low variance among the different iterations, which means that the algorithms would perform similarly with different data sets of the same clinic.

To support the table results, this section includes several figures containing a box plot (on the right side) and a line plot (on the left side) to illustrate the variability and dispersion of the results of each evaluation metric through the 10 iterations of cross-validation.

In Figure 6.4 it is represented a box plot [40], which is a standardized way of displaying the distribution of results based on five values: minimum, first quartile (Q1), median, third quartile (Q3), and maximum:

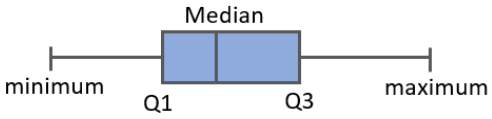

Figure 6.4: Box Plot Interpretation

The Q1 value, also known as 25h Percentile, is the middle result value between the smallest value and the median of the results, which in turn corresponds to the middle value of the results. Q3, also known as 75h Percentile, is the middle value between the median and the highest result.

Line charts were included along with the box plots to provide an easy visualization of the results of each metric across each specific iteration.

The first 8 charts (figs. 6.5 to 6.8) are focused on illustrating the variability and dispersion of the results of MD Clínica dataset while the last 8 (figs. 6.9 to 6.12) are focused on Brazil dataset.

Despite the results from this research, it is important to repeat these tests with the new data that will be provided by the Portuguese clinics since the choice of the algorithm may depend on the given data.

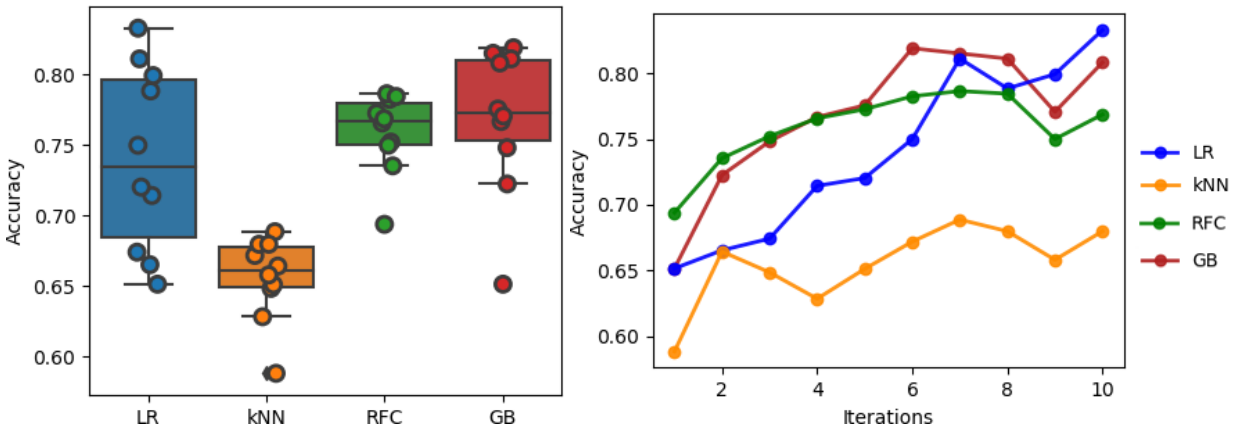

Figure 6.5: Accuracy results over 10 iterations of cross-validation (MD Clínica data)

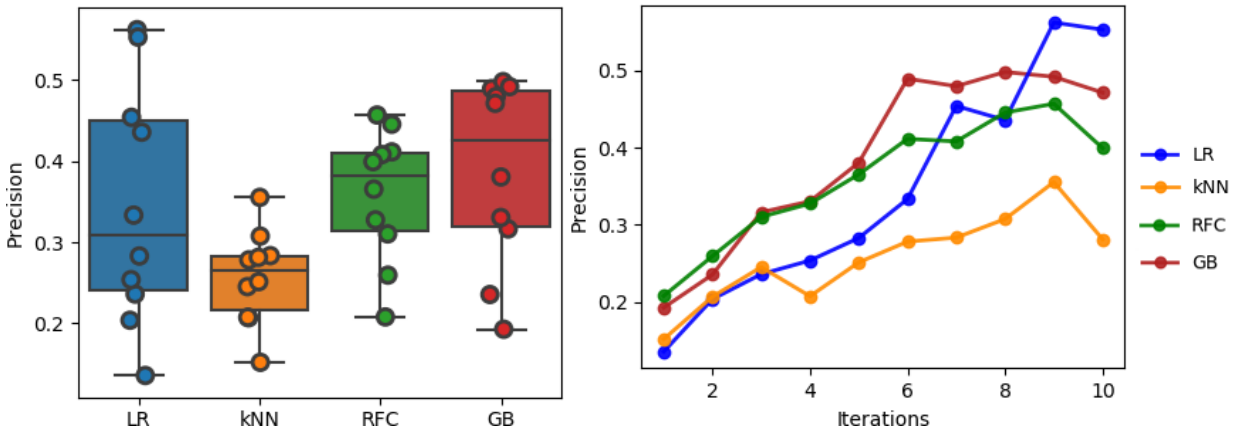

Figure 6.6: Precision results over 10 iterations of cross-validation (MD Clínica data)

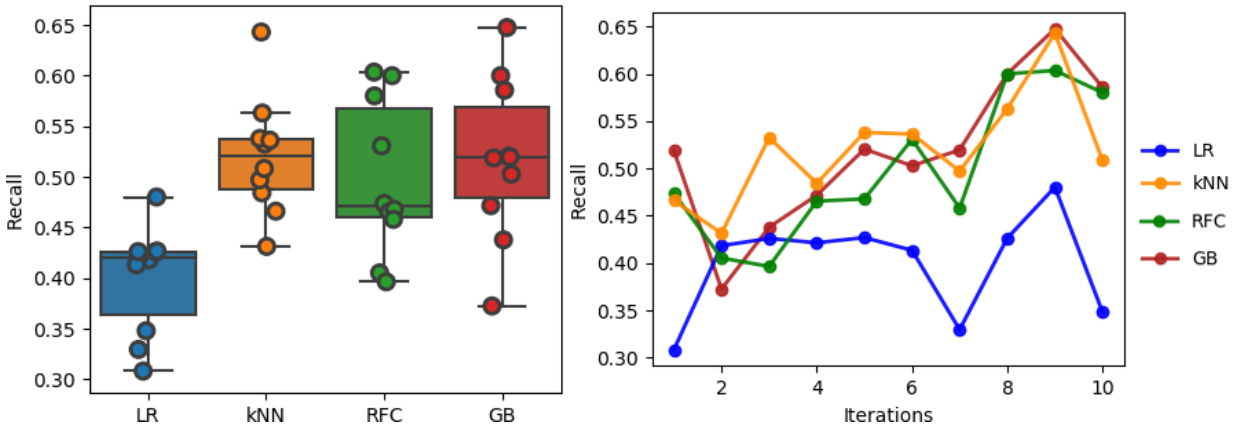

Figure 6.7: Recall results over 10 iterations of cross-validation (MD Clínica data)

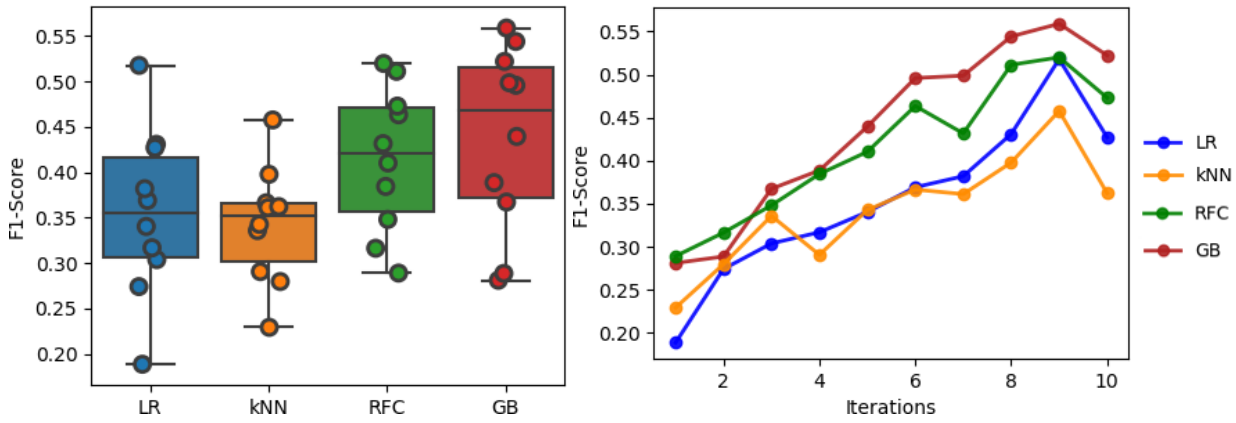

Figure 6.8: F1-Score results over 10 iterations of cross-validation (MD Clínica data)

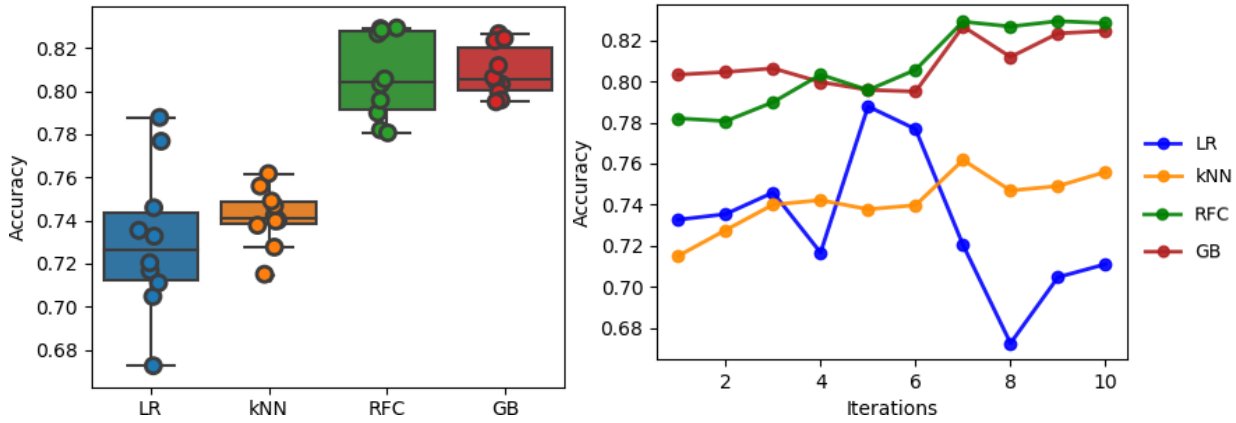

Figure 6.9: Accuracy results over 10 iterations of cross-validation (Brazil data)

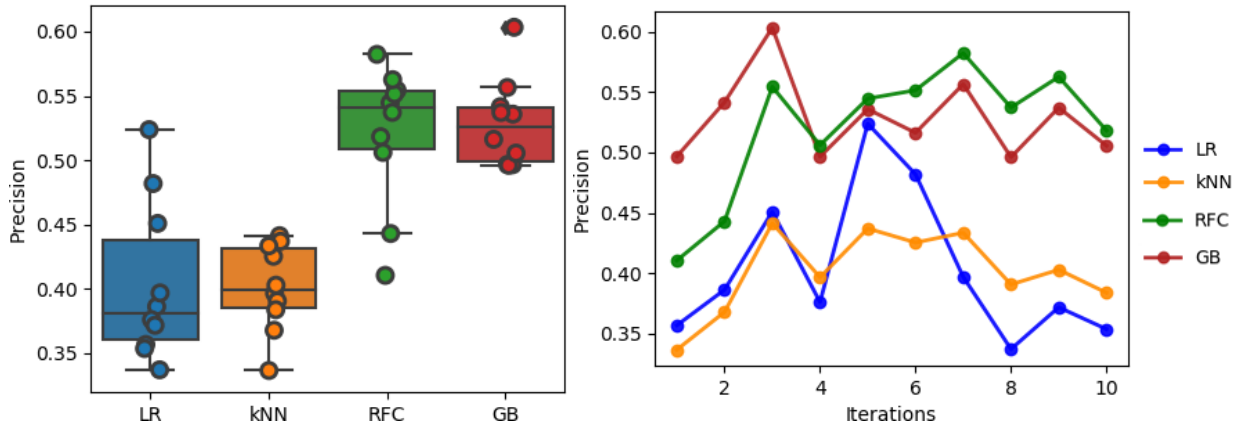

Figure 6.10: Precision results over 10 iterations of cross-validation (Brazil data)

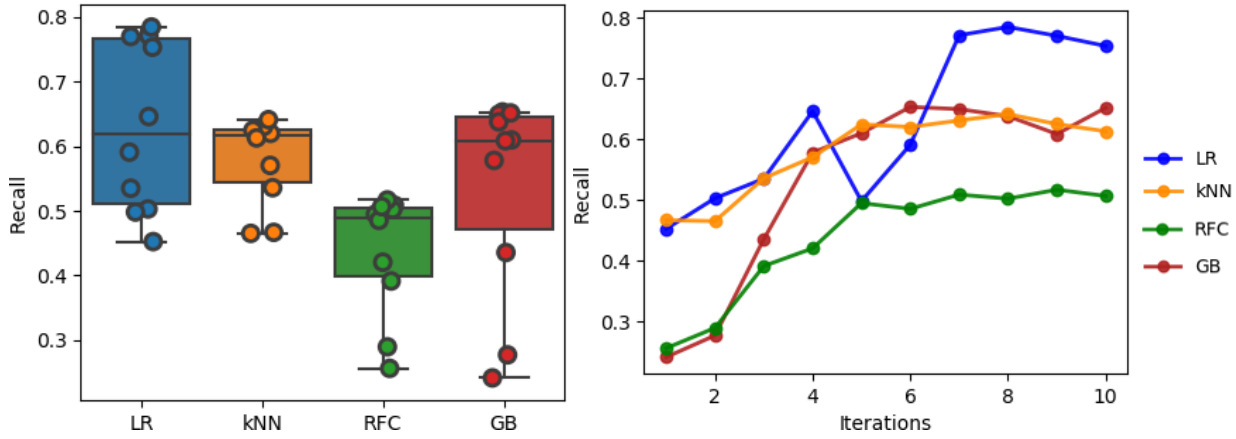

Figure 6.11: Recall results over 10 iterations of cross-validation (Brazil data)

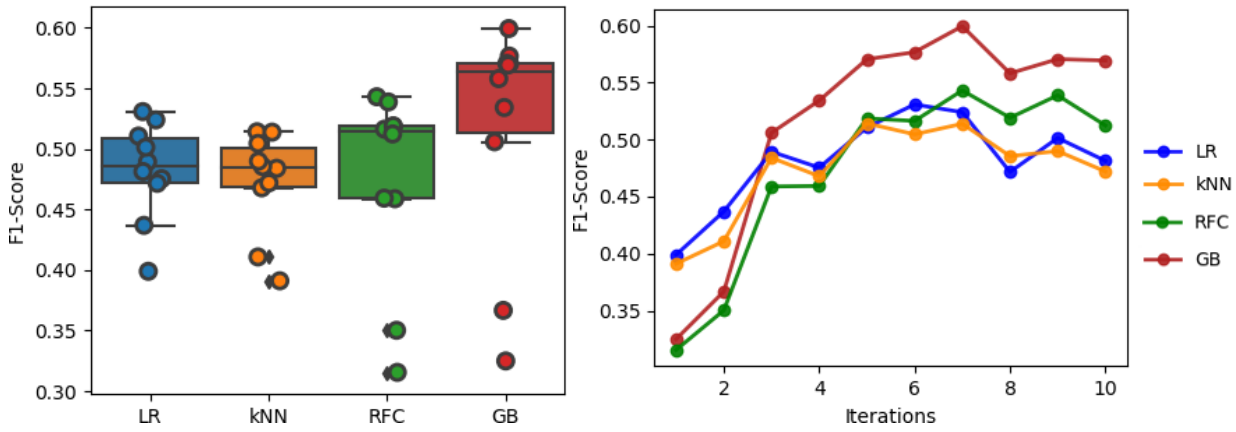

Figure 6.12: F1-Score results over 10 iterations of cross-validation (Brazil data)

6.5 Proposed Solution vs. Previous Solution

This research was applied in the context of MedClick, in which a no-show approach had already been implemented [5]. As detailed on section 4, the previous solution uses a hybrid approach to predict no-shows, which consists of using Logistic Regression, as a population based method, and Bayesian inference, as an individual method. In this research, four classification algorithms were compared and the one that showed the best performance for the considered datasets was Gradient Boosting, an ensemble of decision trees.

Although both solutions had considered data from the same clinic (*MD Clínica*) during their evaluation processes, the provided data was actually different, not only in terms of quantity but also in terms of features. The data considered by the previous solution consisted of 608 000 records while this solution only had access to 10 000 records. Also, the tests that were performed for the previous solution did not considered some important aspects, such as the fact that the data is unbalanced, which, despite the

seemingly satisfactory results, lead to unreliable conclusions.

All these factors make the comparison of both solutions a complex process, making it impossible to perform an objective analysis of the impact of new implemented strategies. However, performing the same tests for both solutions provides a better understanding of which solution is capable of achieving the best performance according to the respective conditions. As such, some tests from the previous solution were repeated using the current available data over the Gradient Boosting algorithm. The evaluation process of the previous solution was divided into three main tests:

- **Model Accuracy:** in this test, the model's accuracy is measured at different splits of data, ranging from 10% of test data to 100%.
- **Feature Behaviour:** this test focuses on measuring the importance of each considered feature.
- **Patient No-show History:** this test is focused on evaluating the personalization of each probability according to the patient's no-show record, which was the main reason of using an hybrid approach in the previous solution.

Since the features from the previous solution were different from the ones considered in this research (whose importances were already discussed in section 6.2), there is no relevance on repeating the second test. As such, only the remaining two tests were performed, results of which will be revealed in the following sections.

6.5.1 Model Accuracy

In this test, the data was divided into different portions of test and training sets, ranging from 10% of test data to 100%. In the latter case, all the data was simultaneously used to training and testing the model. The table 6.6 presents the model accuracy for each split of the previous solution, which, despite the seemingly satisfactory results, lead to unreliable conclusions.

Test data	Accuracy
10%	0.73
20%	0.72
30%	0.7
40%	0.7
50%	0.68
60%	0.67
70%	0.67
80%	0.68
90%	0.67
100%	0.78

Table 6.6: Results from Previous Solution

From 10% to 90% of test data, the previous model reaches around 70% of accuracy which can be easily achieved by assigning all the data to the majority class (show). As previously discussed, to avoid these unreliable results, other metrics should be measured, namely precision, recall and f1-score. In order to compare both solutions, those tests were repeated with the Gradient Boosting model, results of which are presented in table 6.7. This time, to ensure the reliability of the results, the accuracy was measured along with other metrics.

Test data	Accuracy	Precision	Recall	F1-Score
10%	0.79	0.43	0.50	0.46
20%	0.79	0.40	0.55	0.47
30%	0.77	0.40	0.46	0.43
40%	0.77	0.42	0.46	0.44
50%	0.75	0.38	0.47	0.42
60%	0.79	0.41	0.43	0.42
70%	0.74	0.34	0.55	0.42
80%	0.76	0.37	0.54	0.43
90%	0.73	0.32	0.50	0.39
100%	0.87	0.85	0.89	0.87

Table 6.7: Results from Proposed Solution

At first glance, comparing the accuracies from table 6.6 and table 6.7, one may assume that the model from this research is better than the previous one. However, in addition to the lack of evaluation metrics from the previous solution, the features that were considered in each solution were not the same, so the performance of both models should not be compared as a way of concluding which one is the best. In the previous solution, only 4 features were considered while the model from this research have considered 10 features, which may justify the difference between both performances.

In both tables, it is possible to notice that the performance lowers slightly as the training data diminishes in size. Also, as expected, with 100% of data being simultaneously used to training and testing the model, the global performance increases.

6.5.2 Using No-Show History for Profile Personalization

The approach from the previous solution consisted on first computing the probability of no-show based on a given set of features (e.g. age, gender, day of the appointment, etc.) and then apply Bayesian Inference to adapt that initial probability to each patient, using their record of no-shows. In table 6.8 is presented the impact of the patient attendance behavior on the prediction of no-show. The first row corresponds to the initial probability, that was previously computed by Logistic Regression, while the following rows correspond to the first ten appointments of that patient, in which is possible to compare the actual outcome of patient's decision with the predicted probability of no-show. Before comparing both solutions, it is important to notice that, in the previous solution, a probability threshold of 50% was

considered while this solution has chosen a threshold of 30%.

Actual Outcome	No-Show Probability	Predicted Outcome
	37%	
Attended	18%	Attended
Attended	12%	Attended
Attended	9%	Attended
Attended	7%	Attended
Attended	5%	Attended
Missed	17%	Attended
Missed	26%	Attended
Attended	23%	Attended
Attended	21%	Attended

Table 6.8: Impact of Patient's Attendance Behavior (Previous Solution)

From the results in table 6.8, it is possible to notice that as the patient keeps attending to his appointments, his probability of no-show gradually decreases, reaching 5%. However, human behavior is extremely complex so there is no guarantee that in the next appointment will not occur a no-show, as shown on the table 6.8. For this reason, such importance should not be attached to the patient's attendance behavior, which means that is preferable to implement the solution of this thesis in which the history of no-shows is only considered as a feature of the learning model, along with the remaining features.

To support the previous statement, a patient with the same attendance behavior was created in order to repeat this test with the new proposed solution. The respective results are presented in the following table:

Actual Outcome	No-Show Probability	Predicted Outcome
	33%	
Attended	17%	Missed
Attended	29%	Attended
Attended	32%	Missed
Attended	29%	Attended
Attended	33%	Attended
Missed	24%	Missed
Missed	29%	Attended
Attended	14%	Attended
Attended	13%	Attended

Table 6.9: Impact of Patient's Attendance Behavior (Proposed Solution)

With the solution from this research, the patient probability of no-show does not continuously decrease when he attend to 5 appointments in a row, as presented in the table 6.9. This time, the prior history of no-shows is considered without being given to much importance, which makes the model more prepared for a sudden shift in patient behaviour.

7

Conclusions

Contents

7.1 Contributions

This research is focused on no-shows of the health care sector and seeks to gather all the necessary information to implement a solution capable of reducing no-shows and, consequently, increase the efficient use of clinic resources.

The proposed solution was applied in the context of the MedClick application and aims to improve their system by using the following strategies:

- **Improve the classification algorithm:** The previous solution was based on a hybrid approach which uses both logistic regression for population-based features and bayesian inference for individual features. The only individual feature that was being used to personalize the initial patient probability of no-show was the respective prior history which could be used as a feature of the first classification model. As a way of comparing both options, this thesis performed a comparative analysis between four classification algorithms in order to choose the most suitable for the no-shows problem. Gradient Boosting was the one with the best performance, in which the patient prior history was being sent to the model as one of the many features. After comparing both solutions, the new approach have shown to be more suitable to the no-shows problem, since it proved to be more prepared to sudden shifts on patient behavior.

- **Add relevant features:** in the previous solution, only two features were considered relevant (patient's age and the day of the appointment). In order to provide more information to the model, this solution extracted the following features: patient's age, patient's gender, waiting time, day of the appointment, scheduling day, number of previous appointments, number of previous no-shows, number of patient diseases, scholarship status and finally, patient's handicaps. Most of these features already proved a negative impact on no-show probability in previous studies so, as expected, they improved the model performance.

- **Use the algorithm to detect no-shows:** the previous solution was only using the classification algorithm to sort the candidates list, from the least likely to miss the appointment to the one with the greatest probability of missing it. This solution, in addition, leverages the algorithm to predict no-shows.

- **Use strategies to reduce no-shows:** This solution supports sending notifications before each appointment, in which the patient must confirm their presence. This implementation aims at reducing the probability of no-show and it can be seen as a reminder mechanism since it prevents the patient from forgetting their appointment. In addition, it is also useful for avoiding last minute vacancies because if the patient is already planning to miss their appointment, this mechanism encourages him to notify the clinic in advance.

- **Improve the method of selecting candidates for replacements:** the previous method that was being used to get the list of candidates was not the most appropriate since it was sending numerous notifications to patients who may not be interested. To improve this, this solution uses a list that includes all patients who have already scheduled an appointment at a later date in the same health care center and with the same health professional.

7.2 Conclusions

The world is going through a phase of rapidly escalating costs which implies an efficient use of resources that can be achieved by reducing the occurrence of no-shows. As detailed in the previous section, the work of this thesis was focused on implementing a solution in the context of MedClick platform in order to prevent and predict no-shows in medical appointments.

Unfortunately, it was not possible to perform a more objective analysis of the impact of new implementations since there are many factors that make the comparison between this solution and the previous solution implemented in MedClick a complex process. However, some tests from the previous solution were repeated, which, combined with the tests that were exclusively performed on this research, had resulted in a thorough evaluation, from which the following conclusions have arisen:

- Each classification problem must find their most suitable model and, for that, there are several approaches that can be used, focused on validating the model. Before choosing the evaluation method and the performance metrics that will be used, it is important to analyze the particularities of the respective dataset. Regarding the no-shows problem, it is known that there are typically more shows than no-shows, and as such, the dataset is highly imbalanced. The performance metrics should be chosen taking this into consideration since there are some that may return unreliable results, namely, the accuracy. In addition, it is important to ensure that during the training stage, the model is not affected by look-ahead bias.

- During the pre-processing process, information from the existing studies can be used to understand which features have the biggest impact on the no-shows probability. After extracting those features from the available data, their influence may be confirmed by computing the respective feature importance. Regarding the no-shows problem, this research confirmed the impact of the following features: patient's age, gender, waiting time, insurance status, marital status, number of previous appointments, number of previous no-shows, day of the appointment, scheduling day, number of patient diseases and finally, patient's handicaps.

- Some classifiers return probability outputs that are subsequently converted into classes by using a threshold probability. In order to improve the performance of these classifiers, it is important

to choose an optimal threshold, which must provide a solution that best fits their needs. When dealing with imbalanced datasets, the threshold is typically chosen considering the precision-recall trade-off, in which different approaches may be considered. For this research, it is important to consider each clinic in which this solution will be applied since each approach may lead to different consequences, as mentioned in section 6.3.

- The effectiveness of this solution will depend on the performance of the chosen algorithm and therefore, it is important to test and consider different options. In this thesis, four different algorithms were tested, namely, Logistic Regression, k-Nearest Neighbors, Random Forests and Gradient Boosting. The last one is an ensemble of decision trees and it was the one with the best performance.

In addition to the improvements made in the supervised learning model that is being used to predict no-shows, some features were also implemented in the system that aim to reduce their occurrence, such as the reminder notifications. However, the impact of these new changes can only be measured once the MedClick application is finished and deployed in a real clinic environment.

7.3 Limitations and Future Work

One of the major limitations of this thesis is that there are features that were implemented in the system whose impact can only be measured once the MedClick application is finished and deployed in a real clinic environment. For example, the sending of reminder notifications.

Also, the MD Clínica dataset considered in this thesis would ideally be persisted into MedClick API in order to test the classification models using the API endpoints. However, due to some setbacks, the data was not persisted and consequently, the tests were locally performed.

As mentioned on section 5.2, the classification model should be re-trained to prevent it from becoming unstable. The solution that should be implemented is based on a batch approach which consists of only re-training the model after a certain number of observations have been inserted into the dataset.

Another limitation is related to a feature that was not concluded due to lack of time: the waiting lists. After detecting a last minute vacancy, the previous solution was sending numerous notifications to patients who may not be interested in order to fulfill that time-slot. This solution improved that method by only notifying patients who have already scheduled an appointment at a later date in the same health care center and with the same health professional. However, in addition, the goal was to allow patients to add themselves in waiting lists and once the system detected a no-show, these patients would be notified. This is an important feature that should be taken into consideration on future work in order to improve the efficiency of finding candidates for replacements.

Finally, throughout this research, several parameters were considered, to which default values were assigned, such as, the day of sending reminders, the day of computing the prediction of no-show, the time when the system runs the no-show module, the threshold for no-show probabilities and the number of patients simultaneously receiving the same replacement notification. All these variables must be further explored in the future in order to finding their most suitable values.

Bibliography

[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, December 2002.

[2] K. C. Leong, W. S. Chen, K. W. Leong, I. Mastura, O. Mimi, M. A. Sheikh, A. H. Zailinawati, C. J. Ng, K. L. Phua, and C. L. Teng, "The use of text messaging to improve attendance in primary care: a randomized controlled trial," *Family Practice*, vol. 23, no. 6, pp. 699–705, 2006. [Online]. Available: http://dx.doi.org/10.1093/fampra/cml044

[3] K. Hardy, S. O'Brien, and N. J Furlong, "Quality improvement report: Information given to patients before appointments and its effect on non-attendance rate," vol. 323, pp. 1298–300, 12 2001.

[4] A. Turkcan, L. Nuti, P.-C. DeLaurentis, Z. Tian, J. Daggy, L. Zhang, M. Lawley, and L. Sands, "No-show modeling for adult ambulatory clinics," in *Handbook of Healthcare Operations Management*, 01 2013, pp. 251–288.

[5] D. Sousa, "Medclick: Last minute medical appointments no-show," Master's thesis, Instituto Superior Técnico, Lisbon, 2017.

[6] 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. [Online]. Available: https://doi.org/10.1080/19488300.2014.993006

[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," *BMC Family Practice*, vol. 6, no. 1, p. 47, Nov 2005.

[8] W. B. Park, J. Y. Kim, S.-H. Kim, H. B. Kim, N. J. Kim, M.-D. Oh, and K. W. Choe, "Self-reported reasons among hiv-infected patients for missing clinic appointments," *International Journal of STD & AIDS*, vol. 19, no. 2, pp. 125–126, 2008. [Online]. Available: https://doi.org/10.1258/ijsa.2007.007101

[9] L. Corfield, A. Schizas, A. Noorani, and A. Williams, “Non-attendance at the colorectal clinic: a prospective audit,” *Annals of the Royal College of Surgeons of England*, vol. 90, no. 5, p. 377—380, July 2008. [Online]. Available: http://europepmc.org/articles/PMC2645737

[10] F. Gany, J. Ramirez, S. Chen, and J. C. F. Leng, “Targeting social and economic correlates of cancer treatment appointment keeping among immigrant chinese patients,” *Journal of Urban Health*, vol. 88, no. 1, pp. 98–103, Feb 2011. [Online]. Available: https://doi.org/10.1007/s11524-010-9512-y

[11] S. Liew, S. F. Tong, V. Lee, C. J. Ng, K. Leong, and C. Teng, “Text messaging reminders to reduce non-attendance in chronic disease follow-up: A clinical trial,” vol. 59, pp. 916–20, 09 2009.

[12] C. E. Guse, L. Richardson, M. Carle, and K. Schmidt, “The effect of exit-interview patient education on no-show rates at a family practice residency clinic,” *The Journal of the American Board of Family Practice*, vol. 16, no. 5, pp. 399–404, 2003.

[13] S. Cameron, L. Sadler, and B. Lawson, “Adoption of open-access scheduling in an academic family practice,” *Canadian Family Physician*, vol. 56, no. 9, pp. 906–911, 2010.

[14] S. Cashman, J. Savageau, C. Lemay, and W. Ferguson, “Patient health status and appointment keeping in an urban community health center,” *Journal of health care for the poor and underserved*, vol. 15, pp. 474–88, 08 2004.

[15] 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,” *Health Informatics Journal*, vol. 16, no. 4, pp. 246–259, 2010. [Online]. Available: https://doi.org/10.1177/1460458210380521

[16] K. Bennett and E. Baxley, “The effect of a carve-out advanced access scheduling system on no-show rates,” in *Family medicine*, vol. 41, 02 2009, pp. 51–6.

[17] M. T Compton, B. Rudisch, J. Craw, T. Thompson, and D. Antonio Owens, “Predictors of missed first appointments at community mental health centers after psychiatric hospitalization,” *Psychiatric services (Washington, D.C.)*, vol. 57, pp. 531–7, 05 2006.

[18] A. George and G. Rubin, “Non-attendance in general practice: A systematic review and its implications for access to primary health care,” vol. 20, pp. 178–84, 05 2003.

[19] 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, 10 2014.

[20] N. Antonio, A. De Almeida, and L. Nunes, “Predicting hotel booking cancellation to decrease uncertainty and increase revenue,” *Tourism and Management Studies*, vol. 13, pp. 25–39, 04 2017.

[21] R. D. Lawrence, S. J. Hong, and J. Cherrier, “Passenger-based predictive modeling of airline no-show rates,” in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’03. New York, NY, USA: ACM, 2003, pp. 397–406. [Online]. Available: http://doi.acm.org/10.1145/956750.956796

[22] N. S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression,” *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879

[23] K. Chomboon, P. Chujai, P. Teerarassammee, K. Kerdprasop, and N. Kerdprasop, “An empirical study of distance metrics for k-nearest neighbor algorithm,” 01 2015, pp. 280–285.

[24] K. Hechenbichler and K. Schliep, “Weighted k-nearest-neighbor techniques and ordinal classification,” 2004. [Online]. Available: http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-1769-9

[25] L. Rokach, “Ensemble-based classifiers,” *Artificial Intelligence Review*, vol. 33, no. 1, pp. 1–39, Feb 2010. [Online]. Available: https://doi.org/10.1007/s10462-009-9124-7

[26] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[27] H. Trevor, T. Robert, and F. JH, “The elements of statistical learning: data mining, inference, and prediction,” 2009.

[28] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, “Permutation importance: a corrected feature importance measure,” *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 04 2010. [Online]. Available: https://dx.doi.org/10.1093/bioinformatics/btq134

[29] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection.” Morgan Kaufmann, 1995, pp. 1137–1143.

[30] S. Arlot and A. Celisse, “A survey of cross-validation procedures for model selection,” *Statist. Surv.*, vol. 4, pp. 40–79, 2010. [Online]. Available: https://doi.org/10.1214/09-SS054

[31] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*. Morgan Kaufmann Publishers Inc., 1995, pp. 1137–1143. [Online]. Available: http://dl.acm.org/citation.cfm?id=1643031.1643047

[32] D. Powers, “Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation,” *Mach. Learn. Technol.*, vol. 2, 01 2008.

[33] J. A. Hanley and B. J. McNeil, “The meaning and use of the area under a receiver operating characteristic (roc) curve.” *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[34] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets,” *PLOS ONE*, vol. 10, no. 3, pp. 1–21, 03 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0118432

[35] C. Elkan, “The foundations of cost-sensitive learning,” *Proceedings of the Seventeenth International Conference on Artificial Intelligence: 4-10 August 2001; Seattle*, vol. 1, 05 2001.

[36] K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” *CoRR*, vol. abs/1106.1813, 2011. [Online]. Available: http://arxiv.org/abs/1106.1813

[37] N. Littlestone, “From on-line to batch learning,” 12 1989, pp. 269–284.

[38] S. Alasadi, “Review of data preprocessing techniques in data mining,” *Journal of Engineering and Applied Sciences*, vol. 12, pp. 4102–4107, 09 2017.

[39] D. Hand, “Measuring classifier performance: A coherent alternative to the area under the roc curve,” *Machine Learning*, vol. 77, pp. 103–123, 10 2009.

[40] D. L. Massart, J. Smeyers-Verbeke, X. Capron, and K. Schlesier, “Visual presentation of data by means of box plots,” *LC-GC Europe*, vol. 18, pp. 215–218, 04 2005.

A

Execution Logs

In this thesis, a python script was developed to compare the performance of the different models over the two considered datasets.

In essence, the idea was to extract the data from the original dataset, pre-process the extracted data and, then, perform a 10-fold cross validation in which the data is split into 10 folds, one of which is used for testing the models and the remaining 9 are used to train the models. Then, the process is repeated 10 times so that each fold will be used once as a test set. In addition, after splitting the data into training and testing sets, in each iteration, the SMOTE method is applied in order to avoid the models to be biased toward the majority class. Finally, the script concludes by summarizing the performance of the models through the 10 iterations, presenting the average results of the metrics of evaluation.

This appendix includes two log files, that were created using Python's *logging* module, in order to present the steps implemented for each dataset and the respective results.

```
2019-05-03 20:53:30,588: Starting script
2019-05-03 20:53:30,589: Reading data from Brazil_dataset.csv
2019-05-03 20:53:36,703: ---- Brazil_dataset.csv shape: 110527, 13
2019-05-03 20:53:36,723: Extracting new features
2019-05-03 20:54:20,437: ---- Brazil_dataset.csv shape: 110527, 21
2019-05-03 20:54:20,437: Cleaning data
2019-05-03 20:54:20,460: ---- Brazil_dataset.csv shape: 110466, 21
2019-05-03 20:54:20,460: Removing unnecessary features
2019-05-03 20:54:20,471: ---- Brazil_dataset.csv shape: 110466, 13
2019-05-03 20:54:20,473: Starting 10-Fold Cross Validation
2019-05-03 20:54:20,473: 1 iteration
2019-05-03 20:54:20,496: ---- Applying SMOTE
2019-05-03 20:54:22,510: ---- Fit Random Forest
2019-05-03 20:55:20,145: ---- Predict w/ Random Forest & Save performance
2019-05-03 20:55:21,622: ---- Fit Gradient Boosting
2019-05-03 20:56:07,130: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 20:56:07,223: ---- Fit k-Nearest Neighbors
2019-05-03 20:56:09,888: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 20:56:11,207: ---- Fit Logistic Regression
2019-05-03 20:56:12,551: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 20:56:12,615: 2 iteration
2019-05-03 20:56:12,640: ---- Applying SMOTE
2019-05-03 20:56:14,500: ---- Fit Random Forest
2019-05-03 20:57:02,634: ---- Predict w/ Random Forest & Save performance
2019-05-03 20:57:04,150: ---- Fit Gradient Boosting
2019-05-03 20:57:49,566: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 20:57:49,621: ---- Fit k-Nearest Neighbors
2019-05-03 20:57:54,842: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 20:57:57,069: ---- Fit Logistic Regression
2019-05-03 20:57:59,112: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 20:57:59,233: 3 iteration
2019-05-03 20:57:59,325: ---- Applying SMOTE
2019-05-03 20:58:01,997: ---- Fit Random Forest
2019-05-03 20:58:56,467: ---- Predict w/ Random Forest & Save performance
2019-05-03 20:58:57,451: ---- Fit Gradient Boosting
2019-05-03 20:59:40,144: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 20:59:40,234: ---- Fit k-Nearest Neighbors
2019-05-03 20:59:45,723: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 20:59:47,421: ---- Fit Logistic Regression
2019-05-03 20:59:49,484: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 20:59:49,520: 4 iteration
2019-05-03 20:59:49,599: ---- Applying SMOTE
2019-05-03 20:59:52,217: ---- Fit Random Forest
```

```
2019-05-03 21:00:46,915: ---- Predict w/ Random Forest & Save performance
2019-05-03 21:00:47,516: ---- Fit Gradient Boosting
2019-05-03 21:01:29,937: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 21:01:29,987: ---- Fit k-Nearest Neighbors
2019-05-03 21:01:36,963: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 21:01:39,312: ---- Fit Logistic Regression
2019-05-03 21:01:41,125: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 21:01:41,243: 5 iteration
2019-05-03 21:01:41,391: ---- Applying SMOTE
2019-05-03 21:01:43,433: ---- Fit Random Forest
2019-05-03 21:02:41,394: ---- Predict w/ Random Forest & Save performance
2019-05-03 21:02:42,256: ---- Fit Gradient Boosting
2019-05-03 21:03:21,921: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 21:03:21,971: ---- Fit k-Nearest Neighbors
2019-05-03 21:03:29,144: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 21:03:31,559: ---- Fit Logistic Regression
2019-05-03 21:03:33,630: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 21:03:33,707: 6 iteration
2019-05-03 21:03:33,783: ---- Applying SMOTE
2019-05-03 21:03:35,888: ---- Fit Random Forest
2019-05-03 21:04:22,025: ---- Predict w/ Random Forest & Save performance
2019-05-03 21:04:23,094: ---- Fit Gradient Boosting
2019-05-03 21:05:00,354: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 21:05:00,552: ---- Fit k-Nearest Neighbors
2019-05-03 21:05:06,822: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 21:05:08,451: ---- Fit Logistic Regression
2019-05-03 21:05:10,296: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 21:05:10,316: 7 iteration
2019-05-03 21:05:10,339: ---- Applying SMOTE
2019-05-03 21:05:13,010: ---- Fit Random Forest
2019-05-03 21:06:03,345: ---- Predict w/ Random Forest & Save performance
2019-05-03 21:06:04,507: ---- Fit Gradient Boosting
2019-05-03 21:06:42,855: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 21:06:43,045: ---- Fit k-Nearest Neighbors
2019-05-03 21:06:50,082: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 21:06:52,016: ---- Fit Logistic Regression
2019-05-03 21:06:54,144: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 21:06:54,167: 8 iteration
2019-05-03 21:06:54,193: ---- Applying SMOTE
2019-05-03 21:06:56,469: ---- Fit Random Forest
2019-05-03 21:07:47,171: ---- Predict w/ Random Forest & Save performance
2019-05-03 21:07:47,711: ---- Fit Gradient Boosting
2019-05-03 21:08:27,717: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 21:08:27,769: ---- Fit k-Nearest Neighbors
2019-05-03 21:08:36,005: ---- Predict w/ k-Nearest Neighbors  & Save performance
```

```
2019-05-03 21:08:38,111: ---- Fit Logistic Regression
2019-05-03 21:08:40,463: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 21:08:40,546: 9 iteration
2019-05-03 21:08:40,622: ---- Applying SMOTE
2019-05-03 21:08:43,035: ---- Fit Random Forest
2019-05-03 21:09:31,709: ---- Predict w/ Random Forest & Save performance
2019-05-03 21:09:32,477: ---- Fit Gradient Boosting
2019-05-03 21:10:08,710: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 21:10:08,765: ---- Fit k-Nearest Neighbors
2019-05-03 21:10:15,519: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 21:10:17,928: ---- Fit Logistic Regression
2019-05-03 21:10:19,609: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 21:10:19,632: 10 iteration
2019-05-03 21:10:19,657: ---- Applying SMOTE
2019-05-03 21:11:35,400: ---- Fit Random Forest
2019-05-03 21:12:23,891: ---- Predict w/ Random Forest & Save performance
2019-05-03 21:12:25,092: ---- Fit Gradient Boosting
2019-05-03 21:13:04,844: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 21:13:04,895: ---- Fit k-Nearest Neighbors
2019-05-03 21:13:11,454: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 21:13:13,885: ---- Fit Logistic Regression
2019-05-03 21:13:15,431: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 21:13:15,453: Summarizing performance results
2019-05-03 21:13:15,453: ---- Random Forest Results
2019-05-03 21:13:15,453: -------- Accuracy: 0.807082
2019-05-03 21:13:15,453: -------- Precision: 0.521024
2019-05-03 21:13:15,453: -------- Recall: 0.437256
2019-05-03 21:13:15,454: -------- F1-score: 0.473211
2019-05-03 21:13:15,456: ---- Gradient Boosting Results
2019-05-03 21:13:15,456: -------- Accuracy: 0.809136
2019-05-03 21:13:15,456: -------- Precision: 0.528528
2019-05-03 21:13:15,456: -------- Recall: 0.534249
2019-05-03 21:13:15,456: -------- F1-score: 0.517613
2019-05-03 21:13:15,457: ---- k-Nearest Neighbors Results
2019-05-03 21:13:15,457: -------- Accuracy: 0.741595
2019-05-03 21:13:15,457: -------- Precision: 0.401528
2019-05-03 21:13:15,457: -------- Recall: 0.579252
2019-05-03 21:13:15,457: -------- F1-score: 0.473371
2019-05-03 21:13:15,457: ---- Logistic Regression Results
2019-05-03 21:13:15,457: -------- Accuracy: 0.730387
2019-05-03 21:13:15,457: -------- Precision: 0.403480
2019-05-03 21:13:15,459: -------- Recall: 0.630434
2019-05-03 21:13:15,459: -------- F1-score: 0.482028
```

```
2019-05-03 19:35:10,661: Starting script
2019-05-03 19:35:10,661: Reading data from MD_patients.csv
2019-05-03 19:35:11,124: ---- MD_patients.csv shape: 58499, 8
2019-05-03 19:35:11,125: Reading data from MD_appointments.csv
2019-05-03 19:35:11,821: ---- MD_appointments.csv shape: 283010, 5
2019-05-03 19:35:11,822: Merging data using patient_id
2019-05-03 19:35:21,549: ---- merged_data.csv shape: 18493, 13
2019-05-03 19:35:21,549: Cleaning data
2019-05-03 19:35:21,786: ---- merged_data.csv shape: 10123, 12
2019-05-03 19:35:21,788: Extracting new features
2019-05-03 19:35:24,624: ---- merged_data.csv shape: 10123, 19
2019-05-03 19:35:24,624: Removing unnecessary features
2019-05-03 19:35:24,627: ---- merged_data.csv shape: 10123, 10
2019-05-03 19:35:24,627: Starting 10-Fold Cross Validation
2019-05-03 19:35:24,628: 1 iteration
2019-05-03 19:35:24,631: ---- Applying SMOTE
2019-05-03 19:35:24,710: ---- Fit Random Forest
2019-05-03 19:35:28,190: ---- Predict w/ Random Forest & Save performance
2019-05-03 19:35:28,242: ---- Fit Gradient Boosting
2019-05-03 19:35:30,464: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 19:35:30,822: ---- Fit k-Nearest Neighbors
2019-05-03 19:35:31,042: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 19:35:31,163: ---- Fit Logistic Regression
2019-05-03 19:35:31,273: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 19:35:31,282: 2 iteration
2019-05-03 19:35:31,290: ---- Applying SMOTE
2019-05-03 19:35:31,480: ---- Fit Random Forest
2019-05-03 19:35:33,831: ---- Predict w/ Random Forest & Save performance
2019-05-03 19:35:33,878: ---- Fit Gradient Boosting
2019-05-03 19:35:35,224: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 19:35:35,303: ---- Fit k-Nearest Neighbors
2019-05-03 19:35:35,326: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 19:35:35,364: ---- Fit Logistic Regression
2019-05-03 19:35:35,411: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 19:35:35,417: 3 iteration
2019-05-03 19:35:35,421: ---- Applying SMOTE
2019-05-03 19:35:35,513: ---- Fit Random Forest
2019-05-03 19:35:39,434: ---- Predict w/ Random Forest & Save performance
2019-05-03 19:35:39,479: ---- Fit Gradient Boosting
2019-05-03 19:35:42,520: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 19:35:42,703: ---- Fit k-Nearest Neighbors
2019-05-03 19:35:42,749: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 19:35:42,793: ---- Fit Logistic Regression
```

```
2019-05-03 19:35:42,855: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 19:35:42,861: 4 iteration
2019-05-03 19:35:42,867: ---- Applying SMOTE
2019-05-03 19:35:42,990: ---- Fit Random Forest
2019-05-03 19:35:46,344: ---- Predict w/ Random Forest & Save performance
2019-05-03 19:35:46,395: ---- Fit Gradient Boosting
2019-05-03 19:35:48,598: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 19:35:48,690: ---- Fit k-Nearest Neighbors
2019-05-03 19:35:48,716: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 19:35:48,756: ---- Fit Logistic Regression
2019-05-03 19:35:48,811: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 19:35:48,815: 5 iteration
2019-05-03 19:35:48,819: ---- Applying SMOTE
2019-05-03 19:35:48,917: ---- Fit Random Forest
2019-05-03 19:35:53,864: ---- Predict w/ Random Forest & Save performance
2019-05-03 19:35:53,913: ---- Fit Gradient Boosting
2019-05-03 19:35:55,213: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 19:35:55,490: ---- Fit k-Nearest Neighbors
2019-05-03 19:35:55,568: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 19:35:55,711: ---- Fit Logistic Regression
2019-05-03 19:35:55,871: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 19:35:55,884: 6 iteration
2019-05-03 19:35:55,898: ---- Applying SMOTE
2019-05-03 19:35:56,204: ---- Fit Random Forest
2019-05-03 19:36:00,036: ---- Predict w/ Random Forest & Save performance
2019-05-03 19:36:00,078: ---- Fit Gradient Boosting
2019-05-03 19:36:02,930: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 19:36:03,236: ---- Fit k-Nearest Neighbors
2019-05-03 19:36:03,309: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 19:36:03,450: ---- Fit Logistic Regression
2019-05-03 19:36:03,621: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 19:36:03,647: 7 iteration
2019-05-03 19:36:03,664: ---- Applying SMOTE
2019-05-03 19:36:03,759: ---- Fit Random Forest
2019-05-03 19:36:07,611: ---- Predict w/ Random Forest & Save performance
2019-05-03 19:36:07,766: ---- Fit Gradient Boosting
2019-05-03 19:36:09,615: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 19:36:09,688: ---- Fit k-Nearest Neighbors
2019-05-03 19:36:09,707: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 19:36:09,746: ---- Fit Logistic Regression
2019-05-03 19:36:09,783: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 19:36:09,788: 8 iteration
2019-05-03 19:36:09,792: ---- Applying SMOTE
2019-05-03 19:36:09,861: ---- Fit Random Forest
2019-05-03 19:36:14,404: ---- Predict w/ Random Forest & Save performance
```

```
2019-05-03 19:36:14,447: ---- Fit Gradient Boosting
2019-05-03 19:36:17,101: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 19:36:17,180: ---- Fit k-Nearest Neighbors
2019-05-03 19:36:17,200: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 19:36:17,240: ---- Fit Logistic Regression
2019-05-03 19:36:17,280: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 19:36:17,285: 9 iteration
2019-05-03 19:36:17,289: ---- Applying SMOTE
2019-05-03 19:36:17,529: ---- Fit Random Forest
2019-05-03 19:36:21,956: ---- Predict w/ Random Forest & Save performance
2019-05-03 19:36:22,046: ---- Fit Gradient Boosting
2019-05-03 19:36:23,497: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 19:36:23,780: ---- Fit k-Nearest Neighbors
2019-05-03 19:36:23,859: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 19:36:23,926: ---- Fit Logistic Regression
2019-05-03 19:36:23,967: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 19:36:23,970: 10 iteration
2019-05-03 19:36:23,974: ---- Applying SMOTE
2019-05-03 19:36:24,042: ---- Fit Random Forest
2019-05-03 19:36:28,243: ---- Predict w/ Random Forest & Save performance
2019-05-03 19:36:28,285: ---- Fit Gradient Boosting
2019-05-03 19:36:30,637: ---- Predict w/ Gradient Boosting  & Save performance
2019-05-03 19:36:30,739: ---- Fit k-Nearest Neighbors
2019-05-03 19:36:30,809: ---- Predict w/ k-Nearest Neighbors  & Save performance
2019-05-03 19:36:30,960: ---- Fit Logistic Regression
2019-05-03 19:36:31,155: ---- Predict w/ Logistic Regression  & Save performance
2019-05-03 19:36:31,168: Summarizing performance results
2019-05-03 19:36:31,168: ---- Random Forest Results
2019-05-03 19:36:31,170: -------- Accuracy: 0.760149
2019-05-03 19:36:31,170: -------- Precision: 0.361842
2019-05-03 19:36:31,171: -------- Recall: 0.498077
2019-05-03 19:36:31,171: -------- F1-score: 0.416303
2019-05-03 19:36:31,173: ---- Gradient Boosting Results
2019-05-03 19:36:31,174: -------- Accuracy: 0.768070
2019-05-03 19:36:31,174: -------- Precision: 0.386126
2019-05-03 19:36:31,174: -------- Recall: 0.509806
2019-05-03 19:36:31,176: -------- F1-score: 0.432840
2019-05-03 19:36:31,177: ---- k-Nearest Neighbors Results
2019-05-03 19:36:31,177: -------- Accuracy: 0.652691
2019-05-03 19:36:31,177: -------- Precision: 0.251011
2019-05-03 19:36:31,178: -------- Recall: 0.504912
2019-05-03 19:36:31,178: -------- F1-score: 0.333379
2019-05-03 19:36:31,184: ---- Logistic Regression Results
2019-05-03 19:36:31,186: -------- Accuracy: 0.750765
2019-05-03 19:36:31,187: -------- Precision: 0.355228
```

```
2019-05-03 19:36:31,187: -------- Recall: 0.392499
2019-05-03 19:36:31,187: -------- F1-score: 0.359164
```
