TÉCNICO LISBOA

Statistical Methods in Data Mining 2020/21

Classification of Ventricular Fibrillation

Project 1 | Group 4

Inês Pereira | 77059 André Roque | 86694 Carolina Ruivo | 98540 André Luís | 98638

CONTENTS

1 INTRODUCTION

In a classification problem, one aims at predicting the class or category to which a given observation belongs to. The easiest type of classification is binary, where our main goal is just to allocate our observation in one of two classes. We can also have multi-classification problems, in which we consider more than two classes. In both cases, we need to construct a model that learns the patterns of our data, based on a set of input features (i.e. explanatory variables), that can be generalized to new data, in order to make predictions. Classification techniques can be applied to a wide range of different areas. It can be used, for instance, to identify an email as spam or not spam, to understand if a potential customer is a good fit for business, or even to predict if a patient has a certain disease.

In this work, we explore different classifiers to predict if a patient with Myocardial Infarction has developed ventricular fibrillation at the end of the second day, after admission to the hospital. To this end, we train, evaluate and compare different classification algorithms, namely Bayes classifier, K-Nearest Neighbor, Support Vector Machines and Linear Discriminant Analysis, in terms of their performance. We consider different performance metrics, which were decided based on the problem type (i.e. binary) and problem domain (i.e. healthcare).

The document is organized as follows:

- **Section 2 - Data**, where we describe the original dataset and explain the data processing approaches used to deal with missing values;
- **Section 3 - Feature Selection**, where we explore three feature selection techniques to select the features that contribute the most to predict our target variable;
- **Section 4 - Data Visualization**, where we present the data in a graphical way to assess trends and patterns in the data;
- **Section 5 - Classification Methods**, here we present the methods used in the section 6, i.e., the classifiers chosen, and how we addressed the data imbalance between the two prediction classes;
- **Section 6 - Classification**, where we focus on the prediction for ventricular fibrillation, presenting and discussing the results obtained;
- **Section 7 - Conclusions**, where we present the main conclusions of this work and discuss possible paths for future work.

2 DATA

The dataset used for the development of this project incorporates information about Myocardial Infarction, collected between 1992-1995 in Krasnoyarsk, Russia (Golovenkin et al. 2020). In the following sections, we detail the information comprised in the original dataset, as well as the pre-processing techniques needed, before proceeding to achieve the goals of this work.

2.1 Original Dataset

The original dataset contains 1700 observations (i.e. patients), 111 explanatory variables, that contain information about the patient at the time of admission until the third day of hospitalization, and 11 possible complications developed after admission. Nonetheless, since the goal of this work is to predict if a patient has developed a particular complication at the end of the second day after admission, the dataset used consists of a subset of the aforementioned one, composed of 108 explanatory variables and just one response variable. The dataset and a more detailed description of each variable can be found in UCI Machine Learning Repository (Dua & Graff 2017).

2.2 Response Variable

The original dataset has a wide range of possible myocardial infarction complications observed after admission to the hospital. However, in the context of this project, we are only interested in predicting if a patient has developed a specific complication - ventricular fibrillation - at the end of the second day in the hospital. According to the American Heart Association (2016), ventricular fibrillation is considered as the most severe and life-threatening cardiac rhythm disturbances, known for causing sudden cardiac arrest and leading a person to collapse within seconds. This disturbance is caused by a disordered electrical activity that forces the ventricles to quiver uselessly, instead of pumping the blood as expected.

From our dataset, the best response variable to directly measure the occurrence of ventricular fibrillation is the `FIBR_JELUD`. This variable takes the value 1 if a patient has been diagnosed with ventricular fibrillation, and 0 otherwise. On the presented dataset, ventricular fibrillation was detected in only 71 patients, which corresponds to only 4.18% of all observations.

2.3 Missing Values

Before proceeding to the feature selection and classification task, we first assessed the missing values of the dataset. We found our dataset had a total of 8.49% missing values across 107 explanatory features. To solve this problem, we decided to impute the missing observations, but before doing so, we decided to investigate if there were any variables whose percentage of missing values justified its removal from the dataset. In figure 1, all variables that had more than 25% of missing values are shown. As we can see, the variables `KFK_BLOOD` and `IBS_NASL` had 99.76% and 95.76% of missing values, respectively. Imputation of these variables could lead to serious bias of the results, given that there were not enough observations. For that reason, they were removed from the analysis.

To tackle the obstacle of the missing values, we imputed the values using the k-Nearest Neighbors algorithm (kNN) (Kuhn & Johnson 2013). With this algorithm, the missing observations are imputed by finding the samples closest to them, and, based on those observations, statistically inferring the values. In simple words, the kNN algorithm uses the missing values as centroids, and finds their k closest neighbors through a defined distance measure (*e.g.*, Euclidean distance or Mahalanobis distance). After these observations are identified, the values for the missing observations are imputed, for instance, by averaging the values of the neighbors.

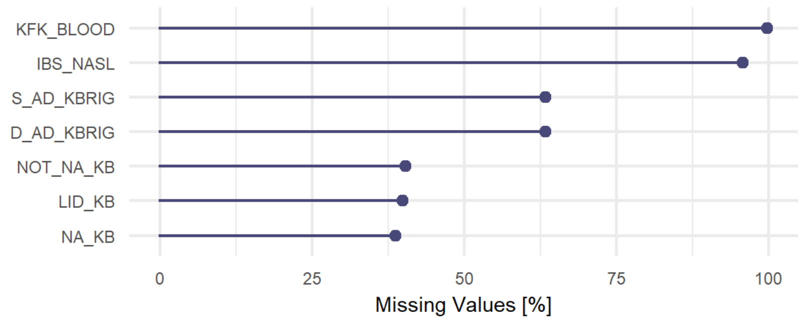


Figure 1: Variables with over 25% of missing values, and respective percentage of actual missing values.

Our dataset is composed of mixed attributes, that is, our explanatory variables range from real values to categorical. Naturally, this must be taken into consideration during the imputation, to assure the imputed values represent the data as best as possible. With that in mind, we used the *R*'s package *VIM* (Kowarik & Templ 2016) to impute the values for our missing data, and resorted to the `kNN` function with $k = 9$. This package was developed to allow for the use of mixed attributes. Particularly, the implementation of the `kNN` function resorts to generalized distances, meaning it is adequate to be used when addressing mixed and differently scaled variables (Kowarik & Templ 2016).

At the end of this pre-processing task, our dataset consists of 107 explanatory variables, which still is a considerable amount of variables to be used for classification. Therefore, we decided to also use feature selection methods in an attempt to find the most representative set of explanatory variables for our problem.

3 FEATURE SELECTION

In order to retain the interpretation of each variable, our initial approach will consist on using different feature selection techniques, with the aim of reducing the dimensions used to predict the occurrence of ventricular fibrillation (`FIBR_JELUD`).

3.1 Data Preparation

The original data set had different data types and some numeric variables were in different scales, which would negatively influence the classifiers applied below.

So, we transformed all ordinal variables into binary ones, by creating a dummy variable for all levels of the previous ones, which increased the initial number of features, 106, to 161 explanatory variables. Then, we standardized all numeric variables.

3.2 Minimum Redundancy, Maximum Relevance (mRMR)

This method uses the Mutual Information Matrix to select a predetermined number of features (De Jay et al. 2013). This matrix is computed using different correlation metrics, depending on the variable type, such that, between continuous features, pearson, spearman, kendall and frequency methods are available. Between discrete features, Cramer's V method is used, and, for other combinations, concordance index is the method applied (De Jay et al. 2020).

Then, variables are ranked according to their correlation with a class (relevance) and the least correlation between themselves (redundancy). We decided the subset composed of the 25% features best ranked, which corresponds to 40 variables, although we propose ourselves to change this percentage if the performance of our model is not satisfactory. The selected variables are described in table 1.

NITR_S	GT_POST	zab_leg_01	INF_ANAM.2	nr04
DLIT_AG.1	lat_im.4	TIKL_S_n	IBS_POST.1	inf_im.4
n_p_ecg_p_10	ZSN_A.1	n_p_ecg_p_09	ZSN_A.3	fibr_ter_07
NA_KB	GB.2	lat_im.3	fibr_ter_06	n_p_ecg_p_03
zab_leg_03	ritm_ecg_p_01	GB.1	O_L_POST	TIME_B_S.5
zab_leg_02	S_AD_KBRIG	endocr_01	GEPAR_S_n	n_r_ecg_p_01
STENOK_AN.6	ALT_BLOOD	GB.3	MP_TP_POST	fibr_ter_03
ROE	DLIT_AG.7	B_BLOK_S_n	FK_STENOK.2	SEX

Table 1: 40 variables selected using mRMR method.

3.3 Boruta Method

In this method, features do not compete among themselves, but with a randomized version of them. We fit the random forest model to the data, and compare the importance of the original features in the model with the shadow features. If a feature performs better than its random version, it is kept, otherwise it is eliminated (Mazzanti 2020).

Then, features are characterized into one of three regions, which correspond to the tails of a Binomial distribution: Refusal, Irresolution or Acceptance (Kursa & Rudnicki 2020). This can be seen in figure 2.

The variables that compose the Boruta features model are presented in table 2.

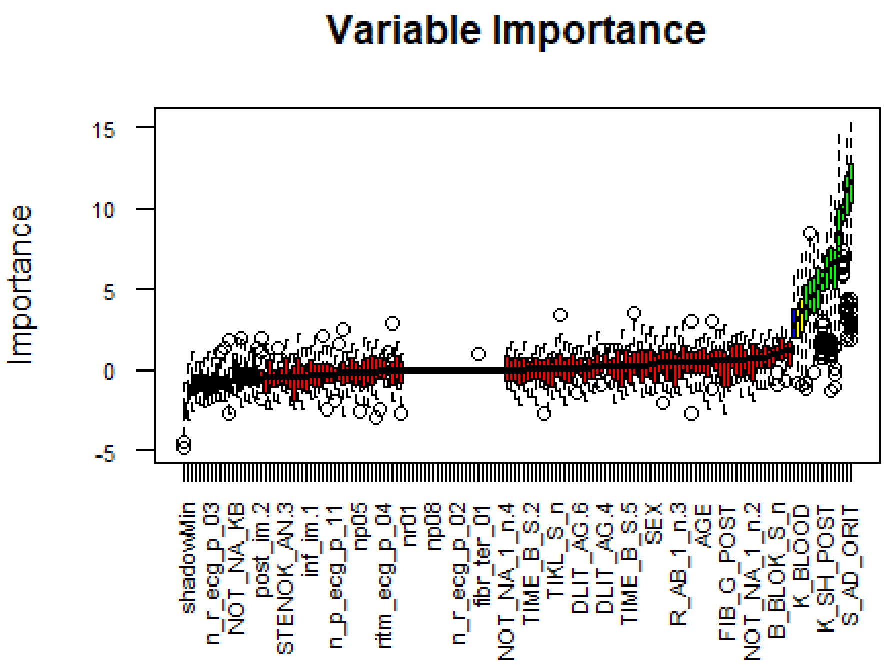


Figure 2: Variable importance with Boruta Method.

S_AD_KBRIG	D_AD_KBRIG	S_AD_ORIT	D_AD_ORIT	K_SH_POST
n_r_ecg_p_10	K_BLOOD	LID_S_n	ASP_S_n	GB.3
ZSN_A.2	R_AB_2_n.2	NA_R_2_n.2		

Table 2: 13 variables selected using Boruta method.

3.4 Recursive Feature Elimination (RFE)

RFE method selects the most relevant features to predict a predetermined target. We define upfront different number of features we want RFE to test.

It starts by building a model on the training set, and computing feature importance on the test set. Then, RFE builds models with different subset sizes, using the feature importance information. Finally, the performance of each model is compared, and the subset of features with smallest error is chosen, as it can be seen in the figure 3 (Prabhakaran 2018). Based on this, forty variables were selected, and they can be consulted in table 3.

n_r_ecg_p_10	R_AB_2_n.3	n_r_ecg_p_08	np01	NOT_NA_2_n.3
nr07	SVT_POST	fibr_ter_05	GB.1	np09
NA_R_2_n.2	n_p_ecg_p_08	fibr_ter_06	n_p_ecg_p_03	ritm_ecg_p_04
K_SH_POST	fibr_ter_08	ZSN_A.4	fibr_ter_07	ZSN_A.2
FIB_G_POST	n_r_ecg_p_09	nr02	np07	np05
STENOK_AN.2	ritm_ecg_p_08	n_r_ecg_p_06	R_AB_2_n.2	TIKL_S_n
n_p_ecg_p_04	STENOK_AN.4	STENOK_AN.1	LID_S_n	STENOK_AN.3
FK_STENOK.1	STENOK_AN.6	GT_POST	n_p_ecg_p_01	fibr_ter_02

Table 3: 40 variables selected using RFE method.

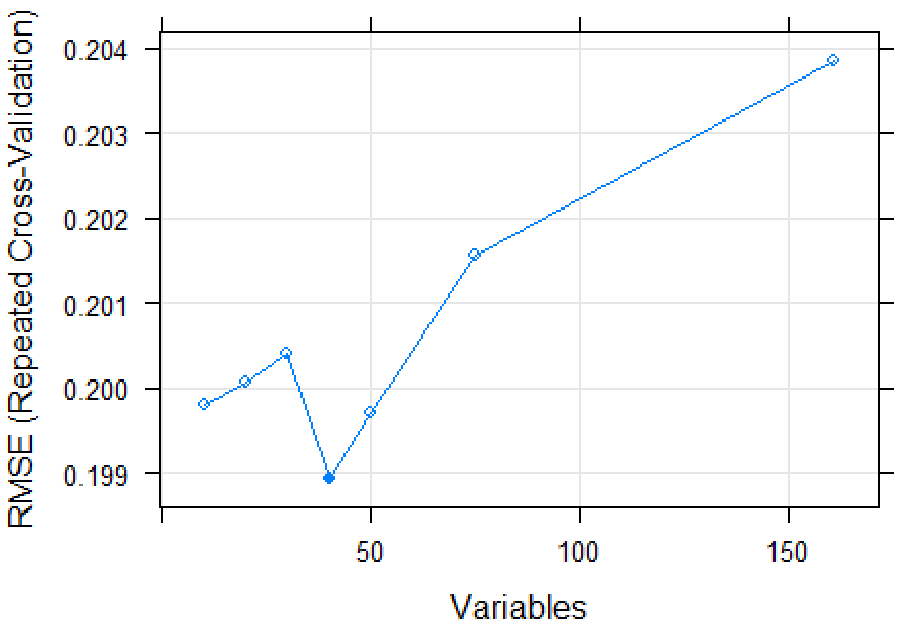


Figure 3: RMSE for 10, 20, 30, 40, 50 and 75 subsets of variables.

4 DATA VISUALIZATION

Exploratory data analysis aims to analyze the dataset we are working with, specially its main characteristics, which is frequently done through the use of visualization. Given that the dataset prior to the feature selection techniques was composed by 107 explanatory variables, presenting graphical visualization for all of them would be unfeasible. Due to that, we decided to perform data visualization to the significant explanatory variables according to the Boruta Method 3.3. As a result, the present section contains some tables and graphics regarding the 13 variables pointed out by the method mentioned above, namely `GB`, `DLIT_AG`, `S_AD_KBRIG`, `D_AD_KBRIG`, `S_AD_ORIT`, `D_AD_ORIT`, `K_SH_POST`, `n_r_ecg_p_10`, `R_AB_2_n`, `NA_R_2_n`, `LID_S_n`, `ANT_CA_S_n`, `ASP_S_n`.

Initially, the correlations among variables were assessed. The blue circles represent positive relationships between variables, while the red circles represent negative ones. The darker the colour and the bigger the circle, the stronger the relationship, and vice-versa.

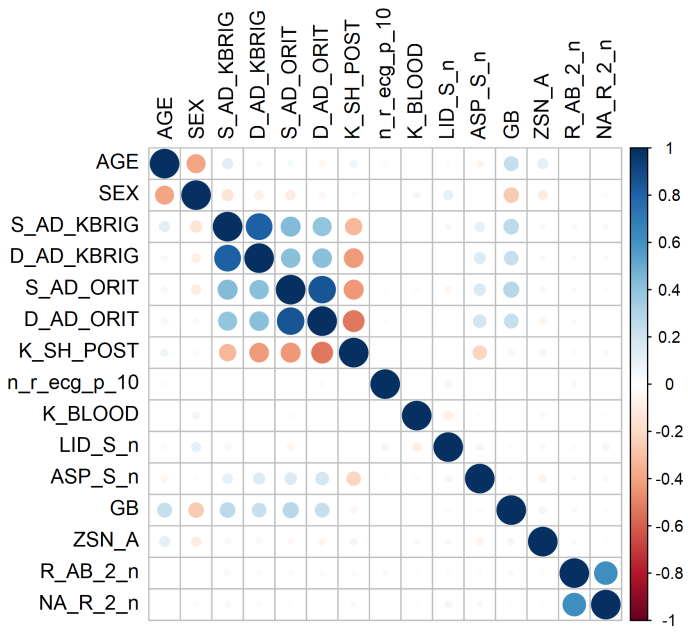


Figure 4: Correlation matrix.

Given the results from the correlation matrix, we decided to plot some graphics with the variables that were positively and negatively correlated. Beyond that, and even though the variables `Age` and `Sex` were removed when we performed feature selection, we believe that including those variables in this visualization part of the project would allow a better understanding and characterization of the patients that compose our data.

The plot below shows the patients' `Age` density distribution by `Sex`, providing a visual form of the distribution of the patient's ages and sex. Being a smoothed version of a histogram, we can analyze that the age range that has the highest concentration of patients is 60 to 70 years old, and that, in general, the male patients admitted to the hospital are younger than the female patients.

The following figure presents the patients' `Age` distribution according to the essential hypertension levels `GB`. High blood pressure levels may lead to severe health complications, namely several heart associated diseases, and even death. Stage 1 stands for prehypertension, Stage 2 stands for moderate hypertension, and Stage 3 stands for severe hypertension. Stage 1 hypertension occurred mostly on younger patients, specially around

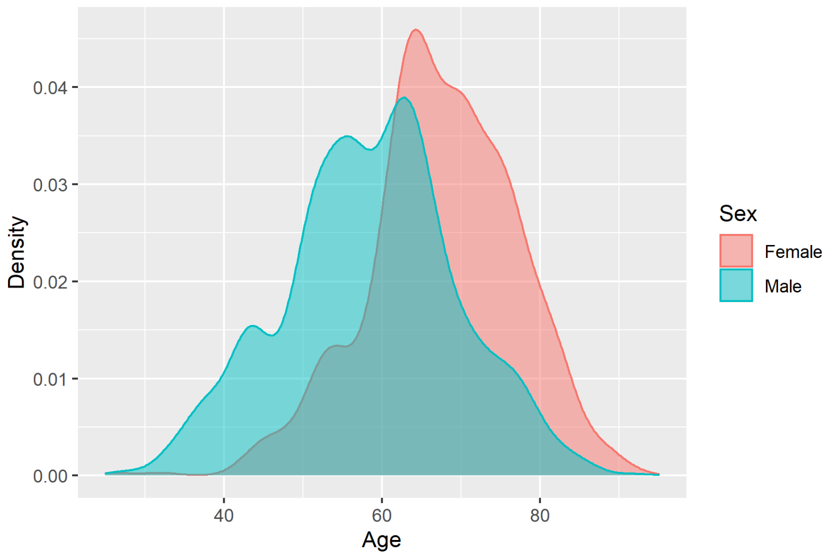


Figure 5: Density plot of the patients' age, divide by sex.

45 years. Stage 2 and 3 have more or less the same distribution, and tend to peak in patients over 60 years old.

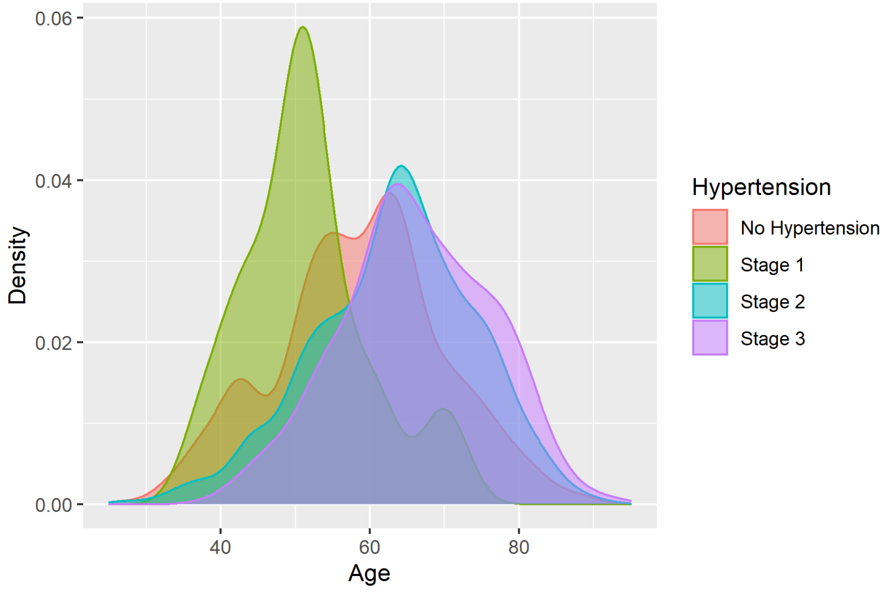


Figure 6: Density plot of the patients' age, group by their stage of essential hypertension.

We thought it would be interesting to analyze graphically two variables that were positively correlated. Figure 7 shows the evolution of the use of opiod drugs `NA_R_2_n` and relapse of the pain `R_AB_2_n` by patients' ages (`AGE`). The graphics are presented side by side for a better comparison, and as expected, are quite similar in their distributions. Most cases of pain relapse, and consequent use of opiod drugs, happen in patients around 60 years of age. Beyond that, a small peak in patients around 40 years old can be seen.

Finally, to complement the analysis of Figure 7, we present below the graphic of two variables negatively correlated, namely diastolic blood pressure `D_AD_ORIT` and cardiogenic shock `K_SH_POST`, by patients' sex

(SEX). As expected, when the pressure in the arteries, when the heart rests between beats, decreases (when D_AD_ORIT decreases), it is registered cases of patients' hearts inability of pumping enough blood to meet their body's needs, i.e., they suffer from cardiogenic shocks.

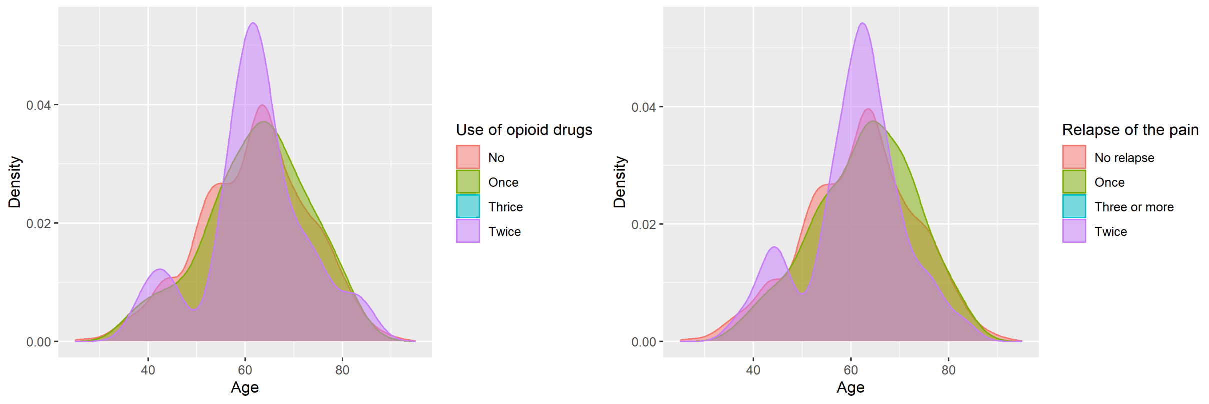


Figure 7: Density plot of the patients' age.

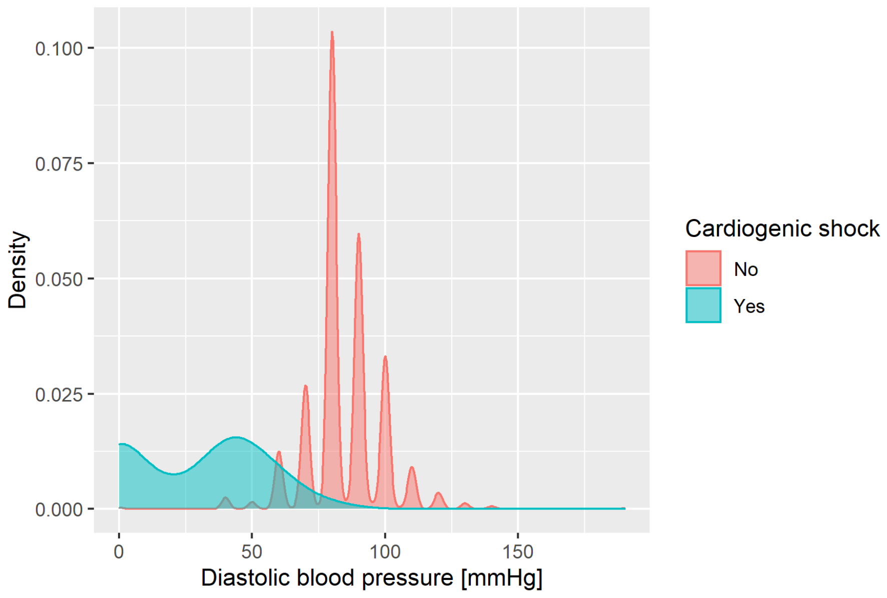


Figure 8: Density plot of the patients' diastolic blood pressure in mmHg.

5 CLASSIFICATION METHODS

This classification project comprises supervised learning algorithms that build a model by learning from labeled data to classify new unseen data with the best accuracy possible.

In the following sections, we present the methods used in this work, namely a brief introduction of the four classifiers used, how the dataset was split into train and test sets, and finally, which performance measures we decided to use according to the problem specificities.

5.1 Classifiers

K-Nearest Neighbors (KNN) is one of the most intuitive supervised learning methods, in the sense it relies on the assumption that similar observations are close to each other, i.e. are neighbors. Given a new observation, KNN considers a set *k* of nearest neighbors and classifies it according to the most represented class in the set.

The Bayes classifiers are a family of classification algorithms based on the Bayes' Theorem[1]. Given an observation $\boldsymbol{x}$ from a random vector $\boldsymbol{X} = (X_1, ..., X_p)$, the classifier tries to find the class ω that maximizes (1), the $a\ posteriori$ probability:

$$P(Y = \omega|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|Y = \omega)P(Y = \omega)}{p(\boldsymbol{x})} \tag{1}$$

where p is the conditional distribution function (continuous or discrete) of $\boldsymbol{X}|Y = \omega$. However, in order to calculate this probability, the conditional distribution function p of a random vector, i.e. the joint conditional distribution of the features, must be known. The motivation for **Naive Bayes Classifier** is given by its strong naive independence assumption between the features, making it possible to transform the joint conditional distribution into the product of distribution functions.

The denominator being constant, and considering the independence assumption between features, the Naive Bayes Classifier becomes:

$$\hat{y} = f(\boldsymbol{x}) = \operatorname*{arg\,max}_{\omega} P(Y = \omega) \prod_{i=1}^{p} P(x_i|Y = \omega) \tag{2}$$

Linear Discriminant Analysis is a tool mainly used for classification problems, and it also allows for data dimensionality reduction. This method predicts the class an observation belongs to according to linear combinations of the explanatory variables. This supervised learning algorithm is reliable, given that it produces good, robust, and easy to interpret results.

The last machine learning method that we implemented was the **Support Vector Machine** (SVM) algorithm. SVM aims to maximize the distinction between the different classes, such that new observations are mapped into that same space and predicted to belong to a category based on the gap they have regarding each class.

5.2 Train and Test Sets

We consider the dataset to be quite imbalanced, since ventricular fibrillation, i.e., our response variable, was diagnosed on less than 5% of the patients. Using imbalanced data during the training stage can lead to poor performance of the classifiers, since they will become less acquainted with one of the classes and, in turn, will not be able to accurately generalize to unseen data during the test stage. Therefore, the train set must be a satisfactory representation of the classification problem with a wide range of examples of all classes.

Additionally, we only have one dataset to be used from training and testing. Thus, we must split into two different datasets: the train set, used to train the classifiers, and the test set, used to assess the performance

[1]Given two events A and B, the conditional probability of A given B is $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$, $P(B) \neq 0$

of the classifiers. Since our data is imbalanced, we must ensure this division is fair, in other words, that we have a similar percentage of all classes in both datasets. To this end, we decided to use stratification to split the dataset, and create the train set with 80% of observations and the test set with 20%. We resorted to the function `stratified` available in the `splitstackshape` library (Mahto 2019).

For the test set, we kept the original observations, since to accurately evaluate the performance of classifiers, the observations must be as close to reality as possible. However, for the train set, we decided to attempt balancing the classes as a way to avoid bias towards the majority one. We researched two different approaches to do so: undersampling and oversampling. The former aims at balancing the train set by reducing the numbers of observations of the majority class, while the goal of the latter is to increase the number of observations of the minority class by sampling the exiting ones. The main advantage of the undersampling is that it reduces the computational effort of training a classifier at the cost of eliminating true observations. On the contrary, oversampling keeps all its true observations, but the computational cost of training a classifier is exacerbated since we are using a larger dataset.

Nevertheless, according to Seiffert et al. (2009) and Cateni et al. (2014), binary classification benefits from a mixed strategy approach, where we simultaneously take advantage of under and oversampling strategies. It was this approach that we decided to follow in this project, and which was achieved with the `ovun.sample` function from `ROSE` library (Lunardon et al. 2014). Moreover, while our main goal was to balance the dataset, we still wanted it to have as many true observations as possible. Therefore, instead of trying to create a perfectly balanced train set with 50% observations of each class, we specified the probability of oversampling as 30%. This results in a train set with 70% of observations of the majority class, and 30% of the minority class.

Figure 9 summarizes the two stages used to split and balance the dataset to obtain both train and test sets.

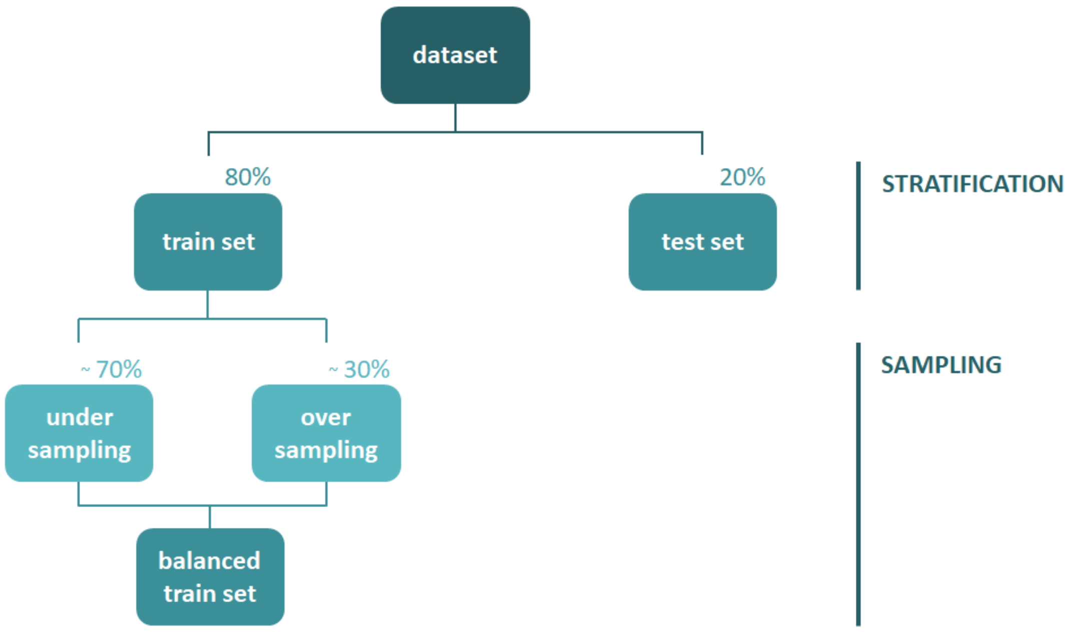


Figure 9: Scheme on how the train and test sets were created.

5.3 Performance Evaluation

When evaluating the performance of the tested classifiers, we must also take into consideration the fact that the test set, which corresponds to a subset of the original dataset, is also imbalanced. For this set, it does not make sense to use a sampling approach as an attempt to balance it, as we want to assess the performance of our classifiers regarding real data. This, in turn, means we must choose adequate performance measures.

We evaluate the performance of the different classifiers with the **recall (or sensitivity)**, *i.e.*, the ratio between the number of true positives and label positives, and the **precision**, *i.e.*, ratio between the true positives and the predicted positives We consider these metrics important in our problem domain, as we want to predict if a patient has developed ventricular fibrillation due to a complication of myocardial infarction. This means that, in our context, a true positive represents a patient with ventricular fibrillation that is diagnosed as such by our model. Conversely, a false negative represents a patient which, despite having developed ventricular fibrillation, our model classifies as not having developed that condition. The latter case is known as an error of type II. Due to the sensitiveness of this topic, we claim that it is preferable that patient is wrongly diagnosed with ventricular fibrillation (false positives or errors of type I) rather than having an error of type II, since the latter can have serious consequences in detecting and treating the disease at an early stage. Therefore, in this particular case, we are interested minimizing the false negatives, which corresponds to maximizing the recall, while simultaneously maximizing the true positives, which corresponds to maximizing the precision.

We also considered the **Confusion Matrix** which not only will allow us to have better insights about the metrics we decided to compute, but also to more accurately assess the errors of type I and type II. Figure 10 shows a scheme of the confusion matrix for the binary case, identifying where the true positives and negatives and the false positives and negatives are located.

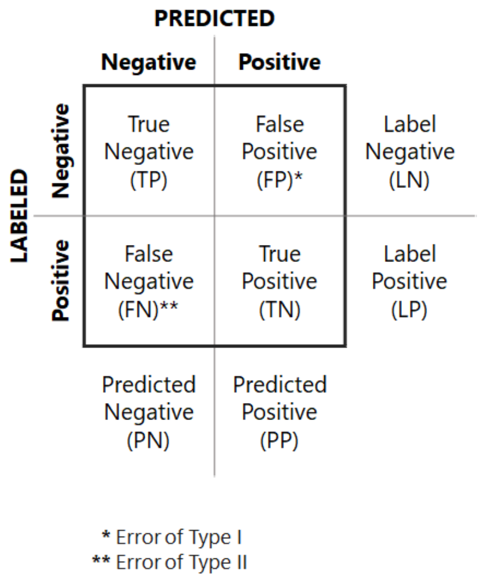


Figure 10: Confusion Matrix scheme for the binary case.

Another metric we took into consideration is the **accuracy** of our classifiers. Nonetheless, since our test set is imbalanced, simply measure the percentage of right predictions is not a good strategy, as it can be achieved by an unskilled classifier that only predicts the majority class. Therefore, we were more interested in the **balanced accuracy**, which is a fairer metric when data is imbalanced. Balanced accuracy is the arithmetic mean between the recall and the specificity [2].

A brief description of all performance metrics used in this work can be found at Appendix A.

[2]**Specificity** - the ratio between the true negatives and the actual number of negatives in the train set.

6 CLASSIFICATION

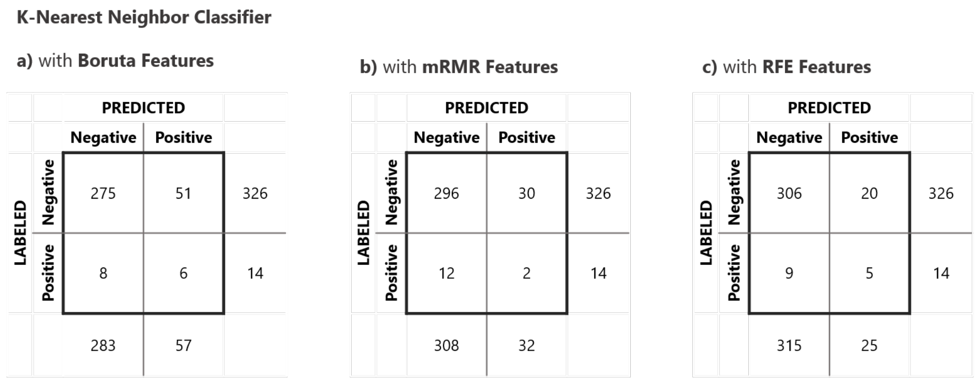
In this section, we present the results we obtained. Each one of the classifiers we used (presented in Section 5.1) was trained and tested using a different set of features, using 5-fold cross validation and resulting of the three feature selection approaches performed in Section 3. Our goal is to assess not only which classifier performed the best for the given problem, but also which feature selection method help achieve those results. At the end of this section, we also presented a brief discussion regarding the overall results.

6.1 K-Nearest Neighbor Classifier

We applied the k-Nearest Neighbors classifier to our data, using the R library `caret` (Kuhn (2019)).

The three implementations of the algorithm selected a number of neighbors(k) equal to 5 as the best option, which can be explained by the imbalance of our data set. Since we have more negative than positive values, the probability of being near negative class neighbors is higher. Thus, a smaller number of nearest neighbors is preferred to increase probability of being close to positive class neighbors, increasing accuracy.

As we can see in table 4, the performance of this algorithm is higher using the subsets of features selected by Boruta and RFE methods. Between both, we may want to consider the trade-off between balanced accuracy and sensitivity of such classifications, because although RFE features present a higher balanced accuracy, the Boruta features present a higher sensitivity, which in our imbalanced data set may be of superior importance, having in mind our objective to minimize errors of type II.


K-Nearest Neighbor Classifier

a) with **Boruta Features**

		PREDICTED		
		Negative	Positive	
LABELED	Negative	275	51	326
	Positive	8	6	14
		283	57	

b) with **mRMR Features**

		PREDICTED		
		Negative	Positive	
LABELED	Negative	296	30	326
	Positive	12	2	14
		308	32	

c) with **RFE Features**

		PREDICTED		
		Negative	Positive	
LABELED	Negative	306	20	326
	Positive	9	5	14
		315	25	

Figure 11: Confusion matrices for the KNN classifiers.

	KNN		
Performance Metrics	Boruta Features	mRMR Features	RFE Features
Precision	0.105	0.0625	0.2
Recall/Sensitivity	0.429	0.143	0.357
Specificity	0.844	0.908	0.939
Accuracy	0.827	0.877	0.915
Balanced Accuracy	0.636	0.525	0.648

Table 4: Performance indicators for the KNN classifiers.

6.2 Gaussian Naive Bayes Classifier

We decided to test a simpler classifier to see how its performance compares with the other ones. For this purpose, we decided to use the Gaussian Naive Bayes classifier from the R library `naivebayes`. Similar to the other models, this classifier was trained using the selected features for algorithms tested in Section 3.

However, the results with the RFE features were poor: the Naive Bayes model classified all observations as positive. Thus, we decided not to present these results. In addition, in the work Vora & Yang (2017), the Naive Bayes classifier performed the best when took advantage of all explanatory variables. For that reason, we also tried the Gaussian Naive Bayes with all our features, however, in our case, the results were similar with one obtained with the RFE: it predict only the positive class. The performance metrics for obtained for the other features are represented in Table 5 and the respective confusion matrices in Figure 12.

When observing just the results displayed in the table, we can infer that these classifiers yield quite similar results. Nonetheless, the one that was trained and tested with the mRMR features scored a higher recall and a slightly lower precision, indicating that it returned a higher number of true positives at cost of making more type I errors. This can be corroborated by the respective confusion matrix. Regarding the accuracy, the model trained with Boruta features has a higher accuracy, however its balanced accuracy is much lower that its regular accuracy. The other model, however, while having a marginally lower accuracy, have a better balanced accuracy. Based on this observations, from these two model, we prefer the Naive Bayes trained with the mRMR features to solve this problem.

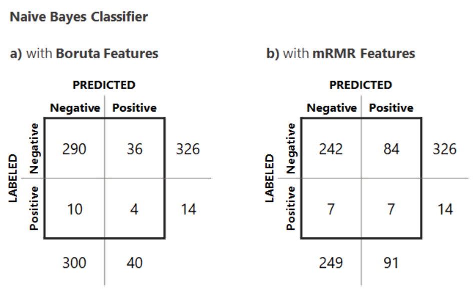


Figure 12: Confusion matrices for the Naive Bayes classifiers.

	Naive Bayes	
Performance Metrics	Boruta Features	mRMR Features
Precision	0.077	0.100
Recall/Sensitivity	0.286	0.5
Specificity	0.890	0.742
Accuracy	0.865	0.732
Balanced Accuracy	0.587	0.621

Table 5: Performance indicators for the Naive Bayes classifiers.

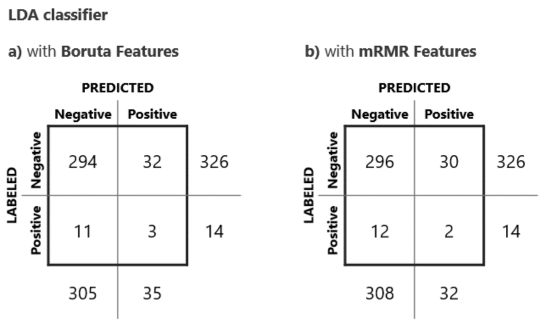
6.3 Linear Discriminant Analysis

To perform the Linear Discriminant Analysis (LDA), we resorted to the `MASS` package (Ripley (2021)) in R. The model with RFE features is not included in this classifier given that RFE features contained four variables that had the same value for all rows. This is also known as the near zero variance predictors, a problem that unabled us from performing LDA, given that one assumption made for LDA is that each attribute has the same variance (Martins (2014)). These near-zero variance predictors were immediately pointed out as soon as we tried to run *lda()* function. A typical solution would be to remove these predictor variables, but this approach would be assuming that those predictors were non-informative, besides the fact that it would no longer be the model selected by the RFE feature selection technique. Another solution would be to collect more data, but that is not possible in the scope of this project, besides it would not be guaranteed that the problem would be solved. Due to this, we decided to perform LDA only to the Boruta and mRMR features.

Figure 13 presents the resulting confusion matrices for the Boruta and mRMR features models. From the values, both models' results seem quite balanced, but that can be assessed with more rigor in Table 6.

For instance, the Precision of both classifiers are poor, meaning that the patients with Myocardial Infarction they predicted to have developed ventricular fibrillation at the end of the second day, after admission to the hospital, most likely did not develop the condition.

Analyzing the Specificity, both models' results are quite good, actually the best amongst all metrics, meaning that the majority of patients that the classifier predicted to have not developed ventricular fibrillation at the end of the second day, after admission to the hospital, most likely did not develop the condition.



LDA classifier

a) with **Boruta Features**

		PREDICTED		
		Negative	Positive	
LABELED	Negative	294	32	326
	Positive	11	3	14
		305	35	

b) with **mRMR Features**

		PREDICTED		
		Negative	Positive	
LABELED	Negative	296	30	326
	Positive	12	2	14
		308	32	

Figure 13: Confusion matrices for the LDA classifiers.

	Linear Discriminant Analysis	
Performance Metrics	Boruta Features	mRMR Features
Precision	0.0857	0.0625
Recall/Sensitivity	0.214	0.143
Specificity	0.902	0.908
Accuracy	0.8735	0.8765
Balanced Accuracy	0.558	0.525

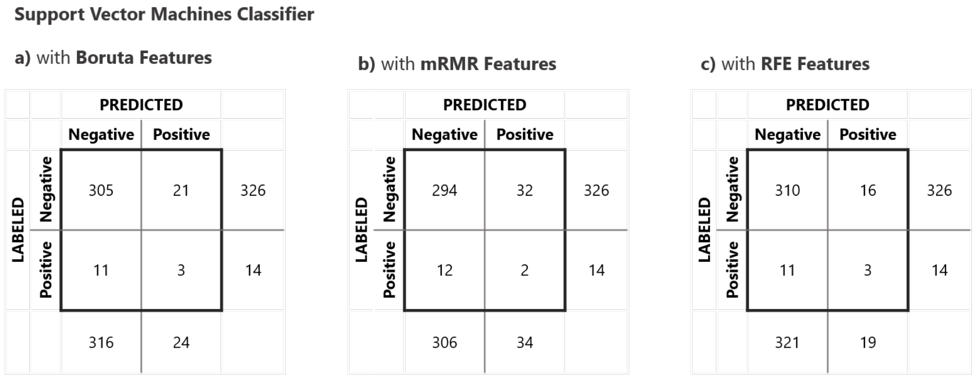
Table 6: Performance indicators for the LDA classifiers.

Finally, analyzing the overall results from table 6, the mRMR features model has better results for Specificity and Accuracy, while the Boruta results are better in Balanced Accuracy, Sensitivity and Precision. Given that we want to minimize error type II, we intend a higher sensitivity and precision, so that we can guarantee less false negatives. Beyond that, we prefer a higher value in balanced accuracy rather than the accuracy score, as it was already discussed before. Due to that, the Boruta features model was the best for the Linear Discriminant Analysis classifier.

6.4 Support Vector Machines

When performing this classifier, we tested several SVM methods, resorting to linear, radial basis, and polynomial functions. The best results overall were given using linear support vector machines, so the presented results refer to this method. This algorithm was implemented using the R library `caret` (Kuhn (2019)).

Table 7 presents the result of this algorithm. Its performance was not significantly higher than the previous classifiers, with balanced accuracies lower than 0.6. However, the best results were obtained using the RFE feature selection technique, where balanced accuracy achieved 0.583. Although the three feature selection techniques have similar results overall, we can identify Boruta and RFE as having the highest sensitivity.



Support Vector Machines Classifier

a) with **Boruta Features**

		PREDICTED		
		Negative	Positive	
LABELED	Negative	305	21	326
	Positive	11	3	14
		316	24	

b) with **mRMR Features**

		PREDICTED		
		Negative	Positive	
LABELED	Negative	294	32	326
	Positive	12	2	14
		306	34	

c) with **RFE Features**

		PREDICTED		
		Negative	Positive	
LABELED	Negative	310	16	326
	Positive	11	3	14
		321	19	

Figure 14: Confusion matrices for the SVM classifiers.

6.5 Discussion

We performed classification using four different algorithms (KNN, Naive Bayes, LDA and SVM), in four different subsets of features (original dataset, Boruta, mRMR and RFE features), which totalizes 16 attempts to predict our target: occurrence of ventricular fibrillation at the end of the second day, after admission to the hospital `FIBR_JELUD`.

From these, the best three are summarized in table 8, with them being KNN with RFE and Boruta features,

	Support Vector Machine		
Performance Metrics	Boruta Features	mRMR Features	RFE Features
Precision	0.125	0.059	0.158
Recall/Sensitivity	0.214	0.143	0.214
Specificity	0.936	0.902	0.951
Accuracy	0.906	0.871	0.921
Balanced Accuracy	0.575	0.522	0.583

Table 7: Performance indicators for the SVM classifier.

	KNN		Naive Bayes
Performance Metrics	RFE Features	Boruta Features	mRMR Features
Precision	0.2	0.105	0.077
Recall/Sensitivity	0.357	0.429	0.5
Specificity	0.939	0.844	0.742
Accuracy	0.915	0.827	0.732
Balanced Accuracy	0.648	0.636	0.621

Table 8: Performance indicators for the three best classifiers.

and Naive Bayes using mRMR features.

According to the initial metric we were more interested in the beginning our analysis, balanced accuracy, one could immediately infer that KNN with RFE features was the best classifier to our data. However, looking in more detail, we can understand that this performance is achieved due to a higher specificity and lower sensitivity when comparing to the other two techniques, influencing negatively the other metric we proposed to minimize, type II error.

Thus, we will prefer the second best balanced accuracy, achieved by KNN with Boruta features, that classifies correctly 63.6% of the patients where ventricular fibrillation occurred. This technique presents a higher sensitivity than the previous technique, which means that we will incur in fewer type II errors.

7 CONCLUSIONS

In this final section, we start by presenting a summary of the best classifier and model, regarding the overall performance metrics. As discussed in 6.5, the best results were achieved with the Boruta features model by KNN. Overall, the results obtained were more balanced than the remaining best model-classifier combinations. With a precision of 0.105, from all patients that were labeled to have developed the condition being studied, 10.5% actually did develop it. With a Sensitivity of 0.429, the classifier correctly identified 42.9% of the patients who developed the condition. A Specificity of 0.844 means that 84.4% of the patients who did not suffer from the condition were predicted correctly by the classifier. Finally, if not considering an imbalanced case, this classifier predicted correctly 82.7% of the cases, with an accuracy of 0.827. However, given that our binary classifier had imbalanced classes - with the value 0, i.e., did not develop the condition, appearing with much higher frequency, the true (balanced) accuracy of the classifier was of 0.636, i.e. 63.6%.

Regarding the RFE feature selection, and the resulting model that included the variables `np01`, `nr07`, `np07` and `fibr_ter_08`, which had the value 0 for all rows - problem we only identified when performing Linear Discriminant Analysis, this probably happened due to the imbalanced dataset that we were given. Actually, inspecting these variables, `np01`, `nr07` and `np07` only had an observation with value 1, while `fibr_ter_08` only had two observations with value 1, having all the others observations the value 0. Given that the dataset was lately divided into train and test sets, the imbalanced problem for these dummy variables was enhanced, and we could not have the two classes represented in both datasets. This is a problem for some techniques or models, as we experienced with Linear Discriminant Analysis.

Coming back to our initial research question, related with the prediction of Myocardial Infarction at the end of second day after admission in the hospital, we understood that, besides solving the classification problem explained above, some features were more relevant to explain this condition. Both RFE and mRMR feature selection techniques selected the following 4 variables, which can indicate their higher importance in predicting the occurrence of Myocardial Infarction.

- `TIKL_S_n` - Use of Ticlid in the ICU;
- `fibr_ter_07` - Fibrinolytic therapy by Ð¡Ðµliasum 250k IU;
- `fibr_ter_06` - Fibrinolytic therapy by Ð¡Ðµliasum 500k IU;
- `n_p_ecg_p_03` - First-degree AV block on ECG at the time of admission to hospital;

Regarding future work, more feature selection techniques can be explored, such as Least Absolute shrinkage and Selection Operator (LASSO) technique, that performs both variable selection and regularization[3] for a better classifier prediction accuracy and easier results interpretability (Jain (2020)). Our proposal is motivated given the near zero-variance predictors the Recursive Feature Elimination technique selected, which unabled us from applying all the classifiers to that model.

The R notebook developed in the aim of this project is available at https://notebooks4smdm.github.io/.

[3] Regularization is used to reduce overfitting of highly complex models. To do so, a penalty term is used so that as the model complexity increases the cost function increases by a huge value.

REFERENCES

American Heart Association (2016), 'Ventricular Fibrillation', https://www.heart.org/en/health-topics/arrhythmia/about-arrhythmia/ventricular-fibrillation. [Last access 07/04/2021].

Cateni, S., Colla, V. & Vannucci, M. (2014), 'A Method for Resampling Imbalanced Datasets in Binary Classification Tasks for Real-World Problems', *Neurocomputing* **135**, 32–41.

De Jay, N., Cavanagh, C. O., Bontempi, G., Haibe-Kains, B. & Haibe-Kains, M. B. (2020), 'Package 'mRMRe''.

De Jay, N., Papillon-Cavanagh, S., Olsen, C., El-Hachem, N., Bontempi, G. & Haibe-Kains, B. (2013), 'mRMRe: an R package for parallelized mRMR ensemble feature selection', *Bioinformatics* **29**(18), 2365–2368.

Dua, D. & Graff, C. (2017), 'UCI Machine Learning Repository: Myocardial infarction complications Data Set', https://archive.ics.uci.edu/ml/datasets/Myocardial+infarction+complications. [Last access 05/04/2021].

Golovenkin, S. E., Bac, J., Chervov, A., Mirkes, E. M., Orlova, Y. V., Barillot, E., Gorban, A. N. & Zinovyev, A. (2020), 'Trajectories, bifurcations, and pseudo-time in large clinical datasets: applications to myocardial infarction and diabetes data', *GigaScience* **9**(11).

Jain, A. (2020), 'Learn how to do Feature Selection the Right Way', https://towardsdatascience.com/learn-how-to-do-feature-selection-the-right-way-61bca8557bef.

Kowarik, A. & Templ, M. (2016), 'Imputation with the R Package VIM', *Journal of Statistical Software* **74**(7), 1–16.

Kuhn, M. (2019), 'The caret Package'.

Kuhn, M. & Johnson, K. (2013), *Applied Predictive Modeling*, Vol. 26, Springer.

Kursa, M. B. & Rudnicki, W. R. (2020), 'Package 'Boruta''.

Lunardon, N., Menardi, G. & Torelli, N. (2014), 'ROSE: A Package for Binary Imbalanced Learning', *R Journal* **6**(1).

Mahto, A. (2019), 'splitstackshape: Stack and Reshape Datasets After Splitting Concatenated Values', https://github.com/mrdwab/splitstackshape. [Last access 14/04/2020].

Martins, T. G. (2014), 'Near-zero variance predictors. Should we remove them?', https://tgmstat.wordpress.com/2014/03/06/near-zero-variance-predictors/.

Mazzanti, S. (2020), 'Boruta Explained Exactly How You Wished Someone Explained to You', https://towardsdatascience.com/boruta-explained-the-way-i-wish-someone-explained-it-to-me-4489d70e154a. [Last access 10/04/2020].

Prabhakaran, S. (2018), 'Caret Package – A Practical Guide to Machine Learning in R', https://www.machinelearningplus.com/machine-learning/caret-package/. [Last access 11/04/2020].

Ripley, B. (2021), 'Support Functions and Datasets for Venables and Ripley's MASS'.

Seiffert, C., Khoshgoftaar, T. M. & Van Hulse, J. (2009), 'Hybrid sampling for imbalanced data', *Integrated Computer-Aided Engineering* **16**(3), 193–210.

Vora, S. & Yang, H. (2017), A Comprehensive Study of Eleven Feature Selection Algorithms and Their Impact on Text Classification, *in* '2017 Computing Conference', IEEE, pp. 440–449.

A PERFORMANCE METRICS

Definition of the performance metrics, we used to assess the performance of our classifiers:

Confusion Matrix: Also known as an Error Matrix, is a table that describes the performance of a classification model. The number displayed in element a_{ij}, for $i = j$, is the number of correct classifications of an observation with class i. If $i \neq j$, then it corresponds to the number of observations that a class i was incorrectly labeled as j.

Precision: Is the ratio of true positives of the total predicted positives.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{3}$$

Recall or Sensitivity: Is the ratio between the true positives and the actual number of positives in the data.

$$Sensitivity = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{4}$$

Specificity: Is the ratio between the true negatives and the actual number of negatives in the data.

$$Specificity = \frac{TrueNegatives}{TrueNegatives + FalsePositives} \tag{5}$$

Accuracy: Is the ratio of observations that were correctly classified, from the total of predictions made.

$$Accuracy = \frac{TruePositives + TrueNegatives}{TruePositives + TrueNegatives + FalsePositives + FalseNegatives} \tag{6}$$

Balanced Accuracy: Is similar to the Accuracy metric, but it takes into consideration unbalanced classes. Is the arithmetic mean between the recall and the specificity.

$$BalancedAccuracy = \frac{Sensitivity + Specificity}{2} \tag{7}$$