Instituto Superior Técnico

Departamento de Matemática

Statistical Methods in Data Mining

First Project

Authors

87005 - Gonçalo Mestre
87101 - Pedro Chanca
87118 - Rui Vasconcelos
97160 - João Bernardo Morais
97136 - Maria Almeida

2019/2020

30 of April of 2020

Contents

Abstract

This report was created with the purpose of presenting the procedure involved in modelling the Customer Intelligence in the Bank. The goal is to, given a set of unknown random variables, predict if a client is active or non-active, in order to prevent the clients from leaving the bank. It is clear that the results of the study are of extreme relevance for the bank, as promoting active clients in an efficient way directly influences the profit of the organisation.

Throughout the report all the methodologies used will be presented and explained, from the Explanatory Data Analysis, through the training of the Classifiers and ending with the conclusions taken.

Keywords: Data Mining, Unknown Variable Name, Banking, R, Machine Learning, Classification

1 Exploratory Data Analysis

In the context of Data Mining, exploratory data analysis (EDA) is the process of analysing a dataset in order to summarize its main characteristics and evaluate the impact of them. For example, it's important to identify missing data, the presence of outliers, and perform tests for the assumptions underlying the multivariate techniques that will be applied. The objective of this examination is "as much to revel what is not apparent as it is to portray the actual data" [1].

This important initial step is often overlooked due to the fact that is time-consuming even though it is necessary.

The overall characterization of the dataset is achieved using both empirical tools, such as those seen in the Section 1.3, and graphical techniques. Their job is complementary and one is not supposed to replace the other.

1.1 Dataset Description

The dataset is readily available through the course web page [2]. The data was already split in two different files: the first, intended to train the models, and the second, to test it. The test set is marked with the true classes and so it's possible to evaluate the performance of the classifier with it. Both files have 12000 instances, which means that each file comprises half of the 24000 total instances. Of these, half correspond to *active* clients and half to *non-active*. See Appendix A.1.

It's possible to conclude that both datasets are balanced in the sense that the proportion of class 0 and class 1 is the same. Since this is the case, there is no need to use special re-sampling techniques to make the training set balanced and so avoid a bias of the model towards one class.

It's important to note that, in order to protect the client's identity, the meaning of each feature is not revealed, which limits the insight one can bring into analysing the problem. Particularly, the codification of the label as being 0 or 1 is unknown and so it limits the flexibility to favour a given class over the other.

There are 36 input attributes, 6 of which are categorical and 30 real-valued. They appear in the dataset following this order with the output class as the last column. Variables were named $V1$-$V36$ in ascending order as the columns appear. The label is hereafter named Y.

There are no missing values in either set. If there were, an analysis of the rows and/or variables affected would have to be performed and decisions about the action to take (either removal or insertion of a given value) had to be made.

The following sections are based on the training data only.

1.2 Descriptive Statistics

In order to analyse the continuous variables, a table was made containing statistics that were considered to be relevant. The table is available in Appendix A.2.

A summary containing the relevant aspects of the analysis follows:

- Every variable has a mean close to 0 and small variance (between 10^{-3} and 10^{-73}). Nevertheless, usually the range, difference between the maximum and minimum values, is close to the unit.
- Variables $V7$, $V9$, $V10$, $V17$ to $V19$ and $V25$ have an interquartile range of exactly 0 which means that at least 50% of the data corresponding to that variable takes exactly the same value. Some of these cases go even further and have the same value repeated more than 75% of the time having the minimum equal $Q3$.
- Every variable has an accentuated heavy-tailed distribution, which means that they have a far heavier tail than the exponential distribution. This conclusion is not as strong regarding the variables $V8$ and $V10$. This implies that a big part of the information is on the tails. For example, a Gaussian distribution has 0 excess *kurtosis*. A heavy-tailed distribution has positive excess *kurtosis* as is the case for every variable in the dataset.
- The previous points may imply that even though the variances are small, there is variability and the small variance may be due to the scale of each variable.
- With the exception of the variables $V8$, $V10$, $V20$, $V21$ and $V30$, which have a skewness close to zero, and therefore have a mean coinciding with the median, are either left-skewed, meaning that the mean value is less than the median, and get a negative skewness, or right-skewed, meaning that the mean value is bigger than the median, and get a positive skewness.
- The previous statements are enough to conclude that none of these variables follows a Gaussian distribution as will be demonstrated in Section 1.3.2.

More conclusions will be drawn with use of graphical methods, complementing the ones already made.

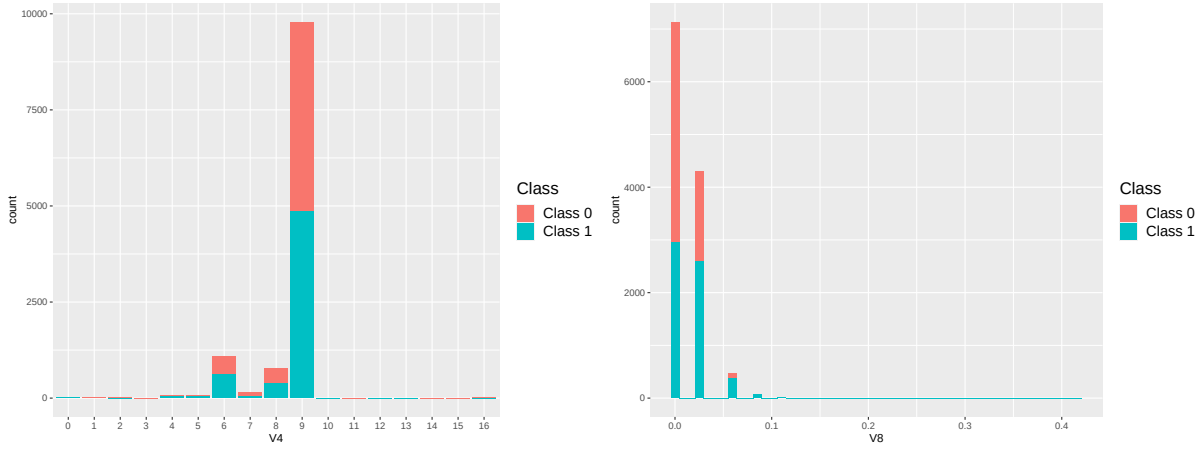
1.3 Variable Behaviour

1.3.1 Bar Plots and Histograms

An histogram is an approximate representation of the distribution of numerical or categorical data [3]. To assess the behaviour of the variables, an histogram was made for each of them, using color to represent the data belonging to a given class. An analysis follows for the histograms (bar plots for the discrete variables) considered more relevant and grouped by variables that hold a similar representation. The bar plots and histograms for all the variables can be found in Appendices A.3 and A.4, respectively.

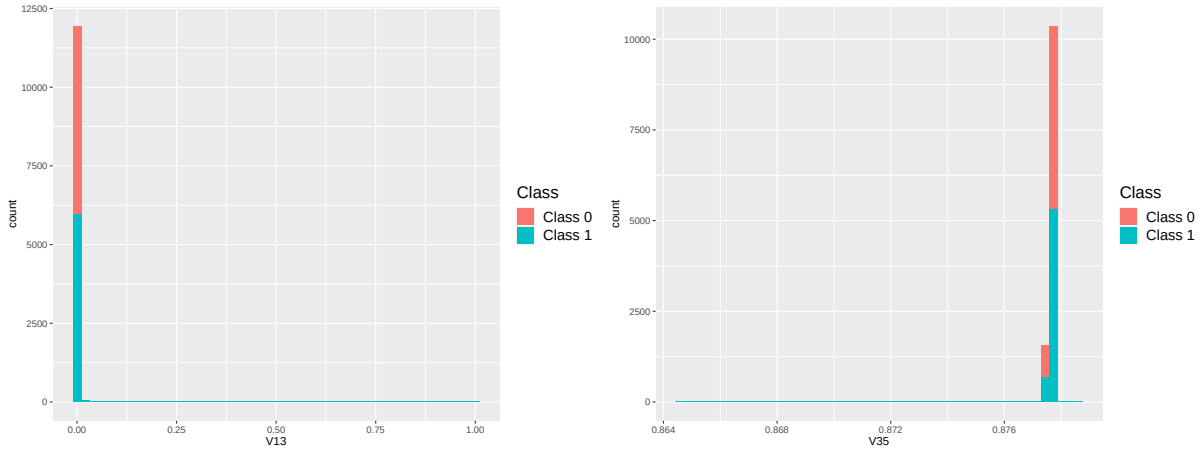
For the discrete variables, $V1$-$V6$, it's possible to verify that the mode of the distribution has a concentration of at least 80% of the data. For the majority of the categories

the data is balanced among the two classes and only for some of those that have a small number of instances do they fall mostly in one class (see Figure 1a).



(a) Bar Plot of variable $V4$ (b) Histogram of variable $V8$

Figure 1: Bar plot for discrete variable and histogram for continuous variable showing discrete set of values.



(a) Histogram of variable $V13$ (b) Histogram of variable $V35$

Figure 2: Two histograms for continuous variables showing small variability.

For the continuous variables, there are 3 types of behavior in the histograms. The first type corresponds to the ones that take only a discrete set of values, namely V7-V10. Note that these 4 variables all seem well distributed among the two classes and that the majority of the data has the same value, except for variable 8, in each the majority of the data is distributed among two values. Figure 1b shows an example of this behavior.

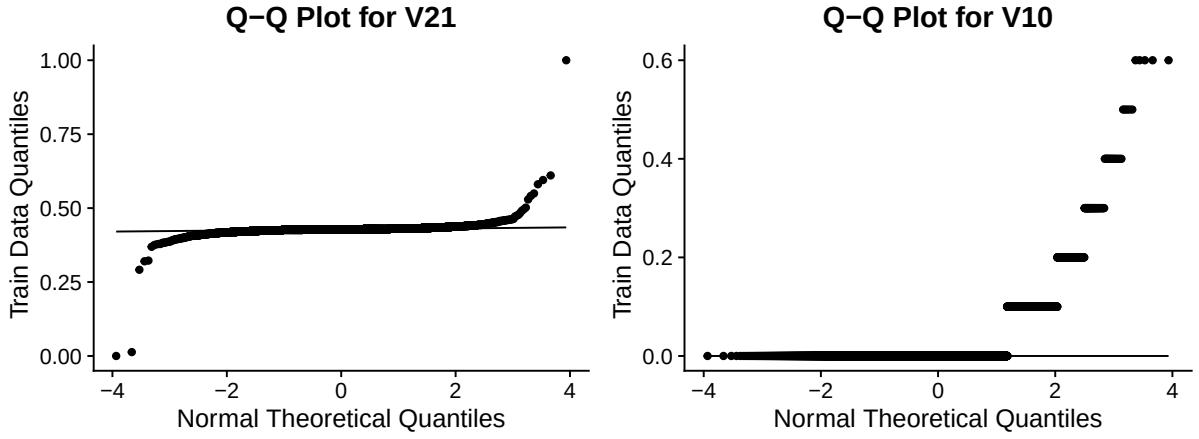
The second type corresponds to the continuous variables in which all data points seem to be concentrated around a specific value, having just some points away from it. Those points can be considered the outliers. These are the variables $V11$-$V26$, $V28$-$V32$, $V34$ and $V36$. Still, they also seem well distributed among the two classes, as seen in Figure 2a.

The third type is very similar to the previous, the difference being that these variables are more disperse around the median value, having an heavier tail. These are the variables $V27$, $V33$ and $V35$. An example can be found o Figure 2b.

Regarding these histograms, one can conclude that for each variable the data is mostly distributed among the two classes, which leads to the conclusion that no variable alone may be used to find active or non-active clients. It is also possible to confirm that the data has a low variability being concentrated on or around a specific value.

1.3.2 Q-Q Plots

A Q-Q Plot is a graphical method that allows to compare probability distributions by plotting their quantiles against each other. Figure 3a shows an example of a Q-Q Plot considering a theoretical normal distribution. If the data follows a similar distribution apart for some linear transformation of variables the points will follow a line [4].



(a) Heavy-tails.

(b) Q-Q Plot showing a discrete set of values

Figure 3: Q-Q Plots for two variables.

Figure 3a resembles a flipped S-shape, which means that the real distribution is heavy-tailed and thus not well modeled by a Gaussian. The same analysis was performed for every explanatory variable and the results were similar. This supports the conclusions made during the analysis of the descriptive statistics. It is also interesting to notice that by analysing the Q-Q Plot for $V10$ that this variable takes only a discrete set of values instead of a continuous, as observed in Figure 3b. This could mean that there is a limitation in the method used to measure this variable.

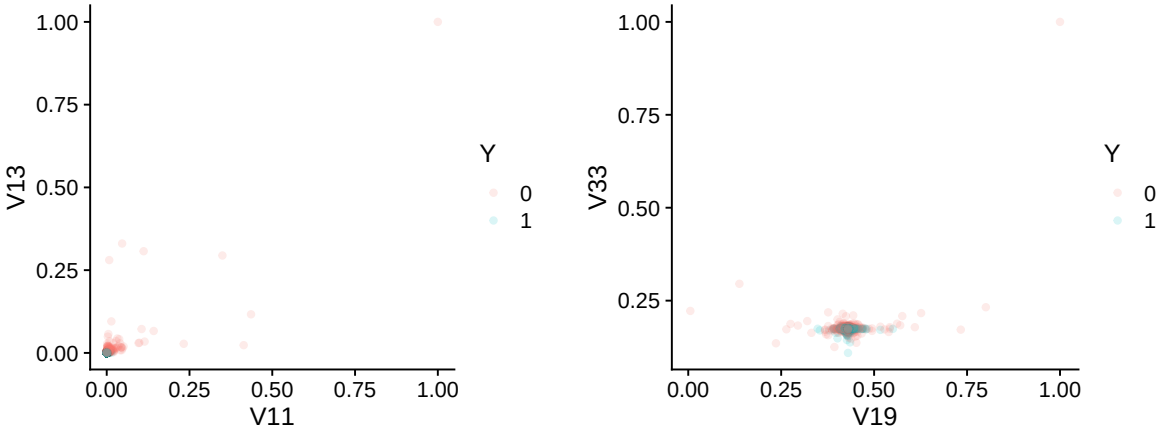
1.4 Feature Engineering

In order to enable more information to be extracted from the dataset there was an effort to develop a set of new features that may allow the statistical methods to obtain a better performance.

One possible approach would be to apply a logarithmic or square root transformation to a variable that is right skewed to try and make it centered. For the left skewed, the variable should first be "reversed", apply one of the referred transformations, and then "reverse" it again. The aim of this would be to build features that are more Gaussian-like.

Other approach, and the one that was taken, was to create new features based on interesting visual characteristics of the data.

The first step was to visualize all the combinations of features using a scatter plot for each, using color to differentiate between target class. As the number of possible combinations between continuous features is $\binom{30}{2} = 435$ combinations, the use of the software Weka was imperative in order to make this an easy process. Since the dataset doesn't have an header row to name the variables, one had to be created otherwise Weka couldn't load the data.



(a) Distance to the origin.

(b) Distance to the mean point.

Figure 4: Scatter plot for pairs of variables.

There are two situations to take into consideration:

1. By observing the scatter plot in Figure 4a it's seen that further from the origin there are only points that belong to class 0. That is, it exists a relation between the distance to the origin and the label. In order to take advantage of this fact, a new variable was created as $V11^2 + V13^2$.

2. From the scatter plot in Figure 4b it is seen that there is also a relationship between the two variables but this time the distance should be measured relatively to the mean of the joint distribution.

It's important to note that a distance to a given point is a function with circular level curves, not elliptical. So there is a need to transform the data to allow for a circular boundary, in some transformed space, to be equivalent to an elliptical boundary in the input space (made of the two variables of interest).

If there is a space (x, y), the equation of an axis align ellipse will be $(\frac{x}{a})^2 + (\frac{y}{b})^2 = r^2$ when a circle is given by $x^2 + y^2 = d^2$. Multiplying the equation for the ellipse by b^2 and making $x' = \frac{x}{a}$ results in $x'^2 + y^2 = r'^2$ which is the equation of a circle. The conclusion is that it is possible to transform an elliptical boundary in a circular boundary by scaling one of the variables by a factor α.

The problem with this approach is that it is difficult to determine the value for α analytically, but a practical approach could be to try different values of α and choose the one that makes the best separation between the two classes.

1.5 Correlation Measures

To assess the dependence between categorical variables the mutual information measure was used. The values were computed using the *method mutual_info_score* from the *sklearn.metrics* Python's package. The corresponding *heatmap* can be found on Figure 5. Note that, on the legend, the numbers from 0 to 5 correspond to the discrete variables $V1$ to $V6$ and the number 6 corresponds to Y. The Pearson correlation implemented in the R library *corrplot* [5] was used to construct a correlogram, as found on Appendix A.5, of the original numerical and engineered features in the 1.4 Section.

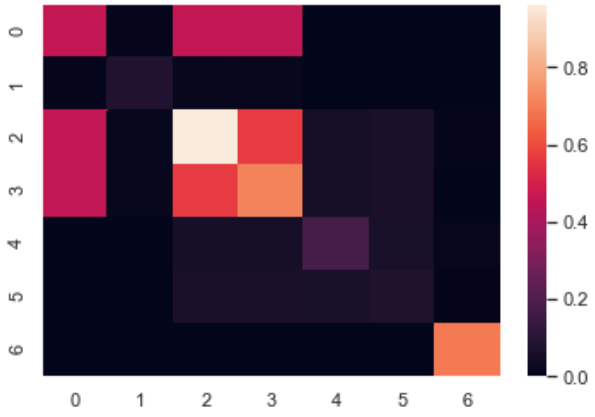


Figure 5: Mutual Information Heatmap for the categorical variables

1.6 Feature Selection

The results from Section 1.5 were used to perform feature selection jointly with a function that performs a Minimum Redundancy, Maximum Relevance feature selection from [6].

From Figure 5 it's possible to observe that variables $V3$ and $V4$ are highly related. According to the results of the mRMR procedure, $V4$ is more relevant than $V3$ and so the latter was removed.

Regarding the continuous variables a correlation threshold of 0.75 was used to select the variables to keep based on the information of the correlogram on Appendix A.5. The information from the mRMR procedure was used as a tie breaker. For example, the variable $V39$ is highly correlated with variables $V11$, $V13$, $V31$, $V33$, $V37$, $V38$, $V40$, $V41$ and $V44 - V50$. Since the procedure returns that the most important variable is variable 39 all the others were removed. The procedure was repeated for the remaining, leading to variables $V22$, $V23$, $V26$, $V29$, $V35$ and $V42$ being removed.

To study the effects of the decisions made during the exploratory data analysis two datasets will be considered, the initial and the processed that underwent the referred transformations.

1.7 Outliers Detection

No outlier detection or removal methods were implemented based on the fact that, since the data has unknown meaning, an important part of the data could be "masked" as an outlier. Also, as the data for each explanatory variable is concentrated around a single value as was seen in the previous analysis, eliminating the points that most methods consider outliers would lead to variables taking a single value, rending them useless.

2 Dimensionality Reduction

Taking into consideration that the dataset has a considerable number of features, applying a dimensionality reduction technique may have a positive impact on some of the classifiers, namely improving their performance, decreasing computational cost and avoiding overfitting the training set.

From the many dimensionality reduction algorithms available, Principal Components Analysis (PCA) was the one selected, since it doesn't require the normality assumption, like Linear Discriminant Analysis (LDA) would have.

2.1 Principal Components Analysis

The PCA creates components (principal components) that are linear combinations of the original variables and are orthogonal between themselves. In addition, they're created in order to preserve the maximum amount of variability of the original data. That is, the variables will be projected into a new feature space with less dimensions, while keeping as much information as possible. These new dimensions will then be used to create a new dataset to be used on the classifiers.

Considering that the variables are measured in different scales and the fact that PCA isn't invariant to scaling, standardized PCA was the algorithm applied through the

use of the function, available on the R's library *stats*, *prcomp* with the parameter *Scale* set to "TRUE" [7].

In order to choose the number of principal components to use, an analysis was conducted on each training set and three criteria were taken into consideration:

1. Kaiser - keep all the components that have a variance bigger than one;
2. Pearson - keep all the components until the cumulative variance reaches 80%;
3. Scree plot - plotted through the function *fvizeig* available on the library *factoextra*.

By looking at the *rotation* component, given by *prcomp*, the conclusions taken for the first and second criteria indicated that 9 and 10 principal components should be kept, respectively. Analysing the scree plot, the indication was to also keep 9 principal components and so it was the value chosen.

Finally, the new dataset was created by multiplying the continuous features with the corresponding index at each principal component, for all the observations on the training and testing datasets. It's relevant to emphasize that PCA was only applied to the continuous variables which were then merged with the categorical variables since this algorithm is not recommended for categorical variables [8].

The same process was applied to the processed dataset using 9 principal components, even though the scree plot suggested the use of only 6.

3 Classifiers

From the analysis from Section 1, two datasets resulted, the *Initial Dataset* and the *Processed Dataset*. In order to evaluate the effect of the dimensionality reduction methods, according to Section 2, two new datasets were generated, namely *Initial Dataset w/ PCA* and *Processed Dataset w/ PCA*.

This section uses the aforementioned datasets to develop models that will solve the proposed problem.

3.1 Metrics

In order to compare classifier models, *score estimation metrics* were used. Therefore, taking into consideration the properties of the Initial Dataset, the chosen metrics were: *confusion matrix*, *accuracy*, *sensitivity*, *specificity* and *precision*.

3.1.1 Confusion Matrix

The confusion matrix is a bidimensional table that represents the number of cases for a given real value *versus* the one predicted by the classifier. Table 1 illustrates the binary

case, given the current problem. A matrix allows to easily check where missclassifications occurred. Its diagonal displays the correct classifications for each class.

		Predicted Value	
		Positive	**Negative**
Real Value	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

Table 1: Confusion Matrix for the binary case.

3.1.2 Accuracy

The *accuracy*, also known as classification error, is the ratio between the number of true cases (positive and negative) and the total number of cases, given by Equation 1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

3.1.3 Sensitivity

The *sensitivity*, also known as *recall*, is the proportion of actual positives that are correctly identified as such and it's given according to the Equation 2.

$$Sensitivity = \frac{TP}{TP + FN} \tag{2}$$

3.1.4 Specificity

Specificity measures the proportion of actual negatives that are identified as being negatives. It is also called *True Negative Rate* and is given according to the Equation 3.

$$Specificity = \frac{TN}{TN + FP} \tag{3}$$

3.1.5 Precision

The *precision*, also known as *Positive Predictive Value*, represents the true positive events, among all the events classified as being positive. The ideal value for this measure is 1 (or 100%) and it is computed by Equation 4.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

3.2 Score Estimation Methods

In order to estimate a model's performance, several methods can be applied such as *Resubstitution*, *Hold-out*, *K-fold Cross Validation*, *Leave One Out Cross Validation* and *Bootstrap*. Some of this methods will be applied to compare and illustrate the difference in the results. Given the Resubstitution's high bias (training and testing the model with the same dataset), this method will not be used.

3.2.1 Hold-out

Hold-out is a simple method that uses a train set to train the model and a test set to estimate how good the model is. Has generally more bias than the following methods.

3.2.2 K-Fold Cross Validation

The *K-Fold Cross Validation* is model validation technique used to assess how the model will generalize when working with independent data. With *Cross Validation*, and comparing its several test errors, it's possible to select the best model for a problem. The idea is to split the data into k folds (usually 5 or 10), with equal sizes, train the model with k-1 parts, and the k^{th} is used to test it. Repeat for the remaining parts.

3.2.3 Leave One Out Cross Validation - LOOCV

Leave One Out Cross Validation is a special case of *Cross Validation* that improves the bias with respect to the latter at the expense of increasing the variance (it's more unstable) and has a higher computational cost. For this case it took approximately 15 times more time to run *LOOCV* compared with *Cross Validation*.

3.2.4 Bootstrap

Bootstrap is a metric that relies on random sampling with replacement. This method generates B datasets, each of size N, by sampling uniformly from the Initial Dataset, and so considering that it represents the population. Then B classifiers will be trained by using the B bootstrap samples. It allows to derive an estimate in a straight foward way, in this case the accuracy.

3.3 Classification Methods

3.3.1 K-Nearest Neighbours (k-NN)

The K-Nearest Neighbours algorithm is a supervised learning and non-parametric method used for classification problems. Its output is computed based on the *k nearest points/training samples* in the feature space according to a specific *similarity* measure between them, namely Euclidean Distance, Manhattan Distance, Jaccard similarity and Gower's similarity, *et cetera*, which should be chosen according to the dataset's proprieties.

Finally, the new observation is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k-nearest neighbors.

On the one hand, this method is easy to understand because of its simplicity, doesn't require that the points from the different classes are linearly separable, and it's easy to generalize. On the other hand, k-NN doesn't have a model, it's only influenced by the current structure of all the points - *lazy learning* - and it's sensitive to noisy, irrelevant and unbalanced data.

The *initial dataset*, used to train the model, has a feature space with much higher dimensionality, one of the cons that deteriorates the performance of k-NN in consequence of the curse of dimensionality [9], and some irrelevant and noisy features that weren't removed. This condition will be reflected on the result, when comparing with the *processed dataset*, that resulted from the data analysis and PCA.

The function used was *kknn* with the *train* method from *carret* [10]. In order to estimate the best value for the several hyperparameters *k*, *kernel* and *distance*, that represent the number of neighbors, the type of kernel and the similarity function, respectively, some score estimation methods were used to estimate the hyperpameters, with $kernel \in \{optimal, rectangular, gaussian\}$, where rectangular is standard unweighted k-NN, $kmax \in \{1, 3, ..., 25\}$, and $distance \in \{1, 2\}$. Given that the function *kknn* only allows to use the Minkowski distance, Gower distance (the most appropriate metric in this case as there is both categorical and numerical data) wasn't used because of the library limitations. The score estimation method used was Cross Validation, due to computational costs, and some of the results from *parameter tuning* are in Table 2.

kmax	kernel	distance	Cross Validation
1	gaussian	1	0.6243
5	optimal	2	0.6354
15	rectangular	1	0.6524
19	optimal	1	0.6880
25	gaussian	1	0.6739

Table 2: Manhattan distance = 1 ; Euclidean distance = 2

Once the model was trained, the predictions were made as in Tables 3 and 4.

	Accuracy	**Recall**	**Specificity**	**Precision**
Initial Dataset	0.6278	0.4901	0.7653	0.6762
Processed Dataset	0.7817	0.7475	0.8160	0.8024
Initial Dataset w/ PCA	0.7840	0.7640	0.8040	0.7958
Processed Dataset w/ PCA	0.6685	0.6455	0.6915	0.6766

Table 3: Results for all dataset with k-NN Classifier.

		Predicted Value	
		Class 0	**Class 1**
Real Value	**Class 0**	4824	1176
	Class 1	1416	4584

Table 4: Confusion Matrix for the k-NN Classifier, for the Initial Dataset w/ PCA.

Taking into consideration the results from the chosen metrics on each dataset, the Initial Dataset w/ PCA achieved the best results on this particular classifier, having an *accuracy* of 78.40%, although it was able to estimate slightly better *class 0* than *class 1*, which can be seen by the *recall* and *specificity* numbers, being the first one of 76.40% and the second 80.40%. This results were expected since the Initial Dataset w/ PCA is comprised with less variables although with a higher contribution in explaining the variability, which leads to a simpler model and therefore a better estimation capacity.

Finally, it's also important to take notice of the Processed Dataset's results since they were quite close from the ones obtained with the *Initial Dataset w/ PCA*, having an *accuracy* of 78.17%. But just like the latter, it was able to estimate with higher rate *class 0* than *class 1*, however the difference is smaller.

3.3.2 Logistic Regression *Boosted*

One of the most widely used algorithms for classification problems is Logistic Regression. This model is a statistical method that allows to model the probability of a given event, in which $y \in \{0, 1\}$ (binary case). Can be extended to the multiclass case.

The model can be written as follows:

$$P(y = 1|x) = g(x^T\beta) \quad P(y = 0|x) = 1 - g(x^T\beta) \tag{5}$$

where

$$g(s) = \frac{1}{1 + e^{-s}} \tag{6}$$

The coefficients of the model are given by:

$$\hat{\beta} = \arg\max\ l(\beta) \tag{7}$$

where l is the *conditional log-likelihood* function

$$l(\beta) = log\ P(y^{(1)}, ..., y^{(n)}|x^{(1)}, ..., x^{(n)}; \beta) \tag{8}$$

In order to improve the performance of the model, a variant of logistic regression by the name *LogitBoost* [11] was used. The method employed comes from the library *carret* and is adapted to deal with big datasets faster and improve the classifier.

This boosted algorithm is an ensemble meta-algorithm (meta-algorithm because it wraps and executes other algorithms) for primarily reducing both the bias and variance, capable of converting a weak learner into a strong one (one that is arbitrarily well-correlated with the true classification) [12].

This allowed for better results when comparing with simple Logistic Regression. The classifier relies on a voting scheme where "nIter" classifiers are applied to each sample. The results are used as "votes" to make the final classification.

Grid-searching was applied in order to compute the best value for the hyperparameter *nIter* (boosting iterations). Given the fact that the classification is performed by a majority vote rule, the number of iterations should not be even or ties will occur, and so the search for *nIter* was restricted to the odd integers in $\{1, ..., 49\}$. Table 5 summarizes the results.

Due to time limitation this method was only applied to the Initial and Processed Dataset where the results were similar. Based on the results from 5 the value chosen for *nIter* was 15. Finally the model was trained with the complete training set for each dataset.

nIter	**Bootstrap**	**Leave One Out Cross Validation**	**Cross Validation**	**Repeated K-fold Cross Validation**
1	0.6239	0.6315	0.6461	0.6276
5	0.6491	0.6583	0.6521	0.6494
15	0.7003	0.6992	0.7075	0.7044
35	0.6931	0.7013	0.7049	0.7011
49	0.6819	0.6992	0.7024	0.7001

Table 5: Results from parameter tuning for LogitBoost.

Once the model was trained, its performance was assessed using the corresponding test set from all four datasets, obtaining the results in Table 6.

	Accuracy	**Recall**	**Specificity**	**Precision**
Initial Dataset	0.6899	0.7074	0.6751	0.6476
Processed Dataset	0.6418	0.6553	0.6303	0.5978
Initial Dataset w/ PCA	0.6125	0.6750	0.5828	0.4338
Processed Dataset w/ PCA	0.6178	0.6332	0.6055	0.5598

Table 6: Results for all datasets with LogitBoost Classifier.

Table 7 shows the confusion matrix for the predictions on the dataset that shows an overrall better performance.

		Predicted Value	
		Class 0	**Class 1**
Real Value	**Class 0**	4393	1607
	Class 1	2114	3886

Table 7: Confusion Matrix for the LogitBoost Classifier, with the Initial Dataset.

Considering the results displayed in Table 6, by comparing the numbers achieved by each dataset on the chosen metrics, it's clear that the *Initial Dataset* has the best results, with an *accuracy* of 68.99%. Also, looking closely at the confusion matrix in Table 7, the classifier can estimate better *class 0* than *class 1*, which is reflected on the higher value for *recall* than *specificity*, whose values are 70.74% and 67.51%, respectively.

Given that the estimate of the *hyperparameter* was for the *Initial* and *Processed Dataset*, the fact that both show better accuracy may be a consequence of that.

Finally, the poor overall results can be explained by the fact that the two classes are mixed with each other, making it extremely difficult for a linear decision method to make the separation.

3.3.3 Random Forest

Random Forest is a very simple but very powerful classifier, frequently allowing for good results. The algorithm is based on an ensemble of decision tree classifiers trained with *bagging*. Unlike other classifiers, Random Forests provide a reliable feature importance estimate and doesn't need lots of data to be trained, working well with datasets with high dimensionality. One con of this algorithm that was especially felt is that training a large number of deep trees can have high computational costs and high memory use. The algorithm to train a *Random Forest* is as follows:

1. generate B training sets $T^{(i)}$ by bootstrap;
2. train a tree classifier from each set $T^{(i)}$ with the *random subspace* method;
3. compute the a posteriori distributions for each tree;
4. aggregate all the a posteriori distributions using Equation 9.

$$\hat{P}(y = k|x) = \frac{1}{B} \sum_{i=1}^{B} P^{(i)}(y = k|x) \tag{9}$$

The parameter that was tuned is *mtry*, which represents the number of variables randomly sampled as candidates at each split according to point 2. The default value for number of trees and maximum depth of each were used. Considering $mtry \in \{1, ..., 20\}$, the results are in Table 8.

mtry	Bootstrap	Leave One Out Cross Validation	Cross Validation	Repeated K-fold Cross Validation
1	*	*	0.7628	0.7601
3	*	*	0.7745	0.7708
12	*	*	0.7590	0.7521
20	*	*	0.7420	0.7304

Table 8: **Due to computational costs, the score methods weren't all used.*

After training the model considering the best value for *mtry* as 3, the labels for the test set were calculated, obtaining the results in Tables 9 and 10.

	Accuracy	Recall	Specificity	Precision
Initial Dataset	0.7464	0.6841	0.8133	0.7856
Processed Dataset	0.9648	0.9371	0.9908	0.9903
Initial Dataset w/ PCA	0.5114	0.9893	0.0335	0.5058
Processed Dataset w/ PCA	0.9633	0.9358	0.9908	0.9902

Table 9: Results for all dataset with Random Forest Classifier.

		Predicted Value	
		Class 0	**Class 1**
Real Value	**Class 0**	5945	55
	Class 1	377	5623

Table 10: Confusion Matrix with the Random Forest Classifier, for the processed dataset.

Looking at the overall results in Table 9, both *Processed Datasets* have an upper hand by a large margin, achieving (without PCA) an accuracy of 96.48% as well as 93.71% and 99.08% for the *recall* and *specificity*, respectively. It is also important to take note that just like in the two previous classifiers, the Random Forest could estimate better class 0 than class 1, which is reflected on the *recall* and *specificity* numbers.

As to be expected, the results can be explained by the fact that the *Random Forest* is in fact an *ensemble learning model*, i.e., it uses multiple learning algorithms, introducing variability, to obtain better estimation performance.

Even though the use of PCA would not be necessary since each decision tree does feature selection by itself, the bad results from the *Initial Dataset w/ PCA* are inconclusive as almost all test examples were predicted as class 1.

3.3.4 Naive Bayes

Naive Bayes is an algorithm that solves a classification problem through a probabilistic approach. It does this by maximizing the posterior probability, assuming that all

explanatory variables X_i are conditionally independent given Y. Note that in Equation 10, $P(y|X)$ is given by the Bayes' theorem.

$$\hat{y} = \arg\max P(y|X) = \arg\max \frac{P(X|y) \cdot P(y)}{P(X)} \tag{10}$$

Although there is loss of information compared to the optimal solution, Naive Bayes may be preferable as it reduces the number of parameters to estimate. Intuitively the classifier chooses the class that has the highest probability of happening given the knowledge it has.

Even considering that the variables are not independent, Naive Bayes still presented as a good algorithm to try. It's a classifier with proven good performance in different machine learning problems, especially when the number of features is large, and it is easy to implement for a large dataset as its training time is lower than other algorithms.

The application of this classifier was made through the R function *train* available on the library *caret* [10]. The parameter *method* was set as "nb", while the parameter *trControl* had different settings for different runs, namely, "cv" (cross validation, with 10 folds), "boot" (bootstrap) and "repeatedcv" (repeated cross validation, with 10 folds and 5 repeats). Finally, the parameter *tuneGrid* was included to consider the use of normal and kernel density function and with or without smoothing (parameter *fL* could be zero or one, assuring that no probability would be equal to zero). Regarding the sub-parameter *adjust*, which allows to adjust the bandwidth of the kernel density, it was kept at one.

After performing the algorithm the best results indicated that the probability density function should be estimated through a kernel and the smoothing improved the results on the runs marked with * in Table 11.

	Bootstrap	**10 Fold Cross Validation**	**Repeated 10 Fold Cross Validation**
Initial Dataset	0.5867	0.5972	0.5956*
Processed Dataset	0.5313	0.5295	0.5256
Initial Dataset w/ PCA	0.5861*	0.5839	0.5853
Processed Dataset w/ PCA	0.5303	0.5183*	0.5229

Table 11: Results on the training set for the Naive Bayes classifier.

Considering that 10-fold cross validation on the initial dataset had the best performance it was chosen to predict the test set, through the function *predict*. This yielded an accuracy of around 62.22%.

By analysing Table 13 it is clear that the classifier's mistakes happened mostly when predicting class 1. This is backed by the *recall* and *specificity* results, being the second much lower than the second (see Table 12).

Accuracy	Recall	Specificity	Precision
0.6222	0.8810	0.3635	0.5806

Table 12: Performance Measures for Naive Bayes on the initial dataset.

		Predicted Value	
		Class 0	**Class 1**
Real Value	**Class 0**	2181	3819
	Class 1	714	5286

Table 13: Confusion Matrix for Naive Bayes on the Initial Dataset.

The final step was to see how each predictor variable was independently responsible for predicting the outcome. That being said, the function *varImp* available on the library *caret* was applied, plotting the absolute value for the t-statistic for each feature [13]. This led to the conclusion that the variables $V8$, $V13$ and $V11$ were the most important, while $V30$, $V18$ and $V10$ were practically irrelevant.

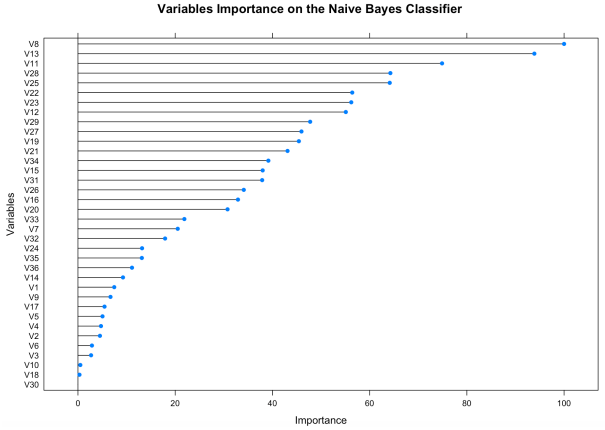


Figure 6: Variable Importance for Naive Bayes on the Initial Dataset.

The Naive Bayes has a very strong premise, which is the independence of each explanatory variable, taken the response variable. Since the overall results are far from satisfactory, one is able to deduce that the premise was violated. In other words, there should be explanatory variables that are correlated with each other, interfering with the well functioning and consequently the model's capacity to make accurate estimations. This issue can be seen in Figure A.5, where it's well represented the high correlation between a large number of variables, in the initial dataset, as previously stated.

3.3.5 Support Vector Machines

Support Vector Machines (SVM) are a decision machine and so don't provide a posterior probability like most methods used until now. Nevertheless it is popular algorithm in the machine learning community.

One of the reasons for this is because they make use of the *kernel trick*, which allows to capture complex relationships between the data points, while keeping the computations in the lower dimensional space.

The idea behind SVM is to build a hyperplane that will separate the classes by maximizing the margin between them. It does this through the use of data points called *support vectors* that result from a constrained optimization problem using Lagrange multipliers. This algorithm was chosen because it is a common method applied to moderate sized datasets in binary classification problems.

Since in some cases the classes are not linearly separable in the feature space, two approaches were developed. Either assign a cost to the missclassified observations and/or apply the *kernel trick* and project the data into a higher dimensional space where there is a higher chance to separate the classes.

This could seem to be a problem as working in high dimensional spaces comes with high computational costs. This is where the "trick" occurs, as the kernel methods represent the data only through a set of pairwise similarity comparisons between the original data (on the lower dimensional space), without having to explicitly apply the transformations and represent the data in the higher dimensional feature space [14], significantly reducing the computational costs.

Considering that the classes aren't linearly separable, the first step was choosing the type of kernel. To do so the training dataset was split in 70% for training and 30% for validation. The algorithm was ran for a linear, polynomial and radial kernels and for a range of hyperparameters, namely, the degree for the polynomial (1 to 5), gamma for all the kernels except the linear (in the range 10^{-7} to 10^{3}) and cost for all (in the range 1 to 10^{20}). This was defined through the R function *svm* available on the library *e1071*. The parameters mentioned are *kernel*, *degree*, *gamma* and *cost*. In addition, the variables were automatically scaled by the algorithm and there was no need to convert the categorical variables into one-hot encoding since they are saved as "factor" and so they are correctly identified by the function through the use of the formula interface [15].

As mentioned, the cost represents how much the classifier will penalize wrongly classified observations, and so it is a regularization parameter. On the other hand, gamma will influence the different kernels on different ways. Finally, the step described above can be seen as pre-training and led to the conclusion that the best kernel was the *rbf*.

The results were then applied to 10-fold cross validation to find the best configuration of hyperparameters. Although this algorithm was applied to all the available datasets, the results for the Processed Dataset w/ PCA aren't presented as they never went over 0.5203 in accuracy. However, it is relevant to mention that the computation cost was lower than on the remaining datasets.

C=100 G= 0.1	C = 1000 G = 0.01	C = 1000 G = 0.1	C = 10000 G = 0.001	C = 10000 G = 0.1
0.6648	0.6739	0.6846	0.6697	0.6821

Table 14: Results on the pre-training for the Initial Dataset and radial kernel.

C=10 G= 1	C = 100 G = 1	C = 1000 G = 1	C = 10000 G = 0.1	C = 100000 G = 0.1
0.6478	0.6494	0.6474	0.6503	0.6495

Table 15: Results on the pre-training for the Processed Dataset and radial kernel.

C = 100 G = 10	C = 1000 G = 10	C = 10000 G = 1	C = 10000 G = 10	C = 100000 G = 1
0.6589	0.6536	0.6593	0.6456	0.6602

Table 16: Results on the pre-training for the Initial Dataset w/ PCA and radial kernel.

Notice that there's a slightly better performance on the *Initial Dataset*. Taking this into consideration, the SVM was then applied to predict the initial test set using a radial kernel, cost 1000 and gamma 0.1. The predictions are shown in Tables 17 and 18.

Accuracy	Recall	Specificity	Precision
0.5417	0.2223	0.8610	0.6153

Table 17: Performance measures for the SVM classifier on the Initial Dataset.

		Predicted Value	
		Class 0	**Class 1**
Real Value	**Class 0**	5166	834
	Class 1	4666	1334

Table 18: Confusion Matrix for the SVM classifier on the Initial Dataset.

Finally, looking at Table 17 the classifier's bad performance is reflected on the *accuracy* value of 54.17% as well as not being able to estimate both classes evenly, since the *specificity* value of 86.10% is much higher than the *recall* value of 22.23%, which means the classifier is able to estimate better *class 0* than *class 1*.

4 Conclusion

The *Initial Dataset* had three major limitations, the first one being the unknown nature/name of each variable, the second the fact that each variable had extremely low variances and third the fact that the dataset was not well suited for a Gaussian distribution approximation, which would limit the performance of the models that could be used. For the first limitation, there were no tools to deal with the problem, but on the other side, for the second limitation 3 additional datasets were created on which were used different data manipulation strategies that could provide an increase in the classifiers performance.

Since the Initial Dataset is balanced, it was chosen as metrics the *Confusion Matrix, Accuracy, Recall, Specificity* and *Precision* which can be good choices to analyse the results for the problem.

Taking a closer look at each dataset, the classifiers where the best performance was achieved are as follows,

- Initial Dataset: Random Forest;
- Processed Dataset: Random Forest;
- Initial Dataset w/ PCA: k-NN;
- Processed Dataset w/ PCA: Random Forest.

On the other hand, regarding the methodology that could lead to a higher performance, by looking at the overall results, it would be to start by processing the dataset as explained in Section 1.4 then from here either apply PCA to the dataset and use the Random Forest classifier by following the steps addressed in Section 3.3.3 or immediately make use of the Random Forest classifier without PCA, since the results according to the chosen metrics were practically the same. It's highly important to take note that by this being the best methodology, it doesn't mean the metrics scores will be the same on a new test dataset, but rather a good approach in order to achieve the desired estimation of the classes.

Finally, regarding future work, further explanatory analysis should be performed, and an investigation conducted in order to assess the best models to apply to variables that behave in the way these do. If more computational power was available, state-of-the-art algorithms like deep neural networks could be tested and further parameter tuning applied to the models already used. If possible, getting some key take ways behind each explanatory variable and insight over the data from a specialist could improve the overall performance of the classifiers.

References

[1] Joseph F. Hair Jr. William C. Black Barry J. Babin Rolph E. Anderson. *Multivariate Data Analysis*.

[2] Conceição Amado. *Project 1*. URL: http://web.tecnico.ulisboa.pt/~ist13493/MEDM2020/Project1/.

[3] Wikipedia. *Histogram*. URL: https://en.wikipedia.org/wiki/Histogram.

[4] Wikipedia. *Q-Q Plot*. URL: https://en.wikipedia.org/wiki/Q-Q_plot.

[5] Taiyun Wei and Viliam Simko. *Corrplot Package*. URL: https://cran.r-roject.org/web/packages/corrplot/corrplot.pdf.

[6] Nicolas De Jay et al. *mRMRe Package*. URL: https://cran.r-roject.org/web/packages/mRMRe/mRMRe.pdf.

[7] R-core. *prcomp*. URL: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/prcomp.

[8] Gustavo Angeles and Stanislav Kolenikov. *PCA*. URL: http://staskolenikov.net/talks/Gustavo-Stas-PCA-generic.pdf.

[9] Wikipedia. *Curse of Dimensionality*. URL: https://en.wikipedia.org/wiki/Curse_of_dimensionality.

[10] Max Kuhn. *train*. URL: https://www.rdocumentation.org/packages/caret/versions/4.47/topics/train.

[11] Jerome Friedman. *LogitBoost*. URL: https://logitboost.readthedocs.io.

[12] Wikipedia. *Boosting*. URL: https://en.wikipedia.org/wiki/Boosting_(machine_learning).

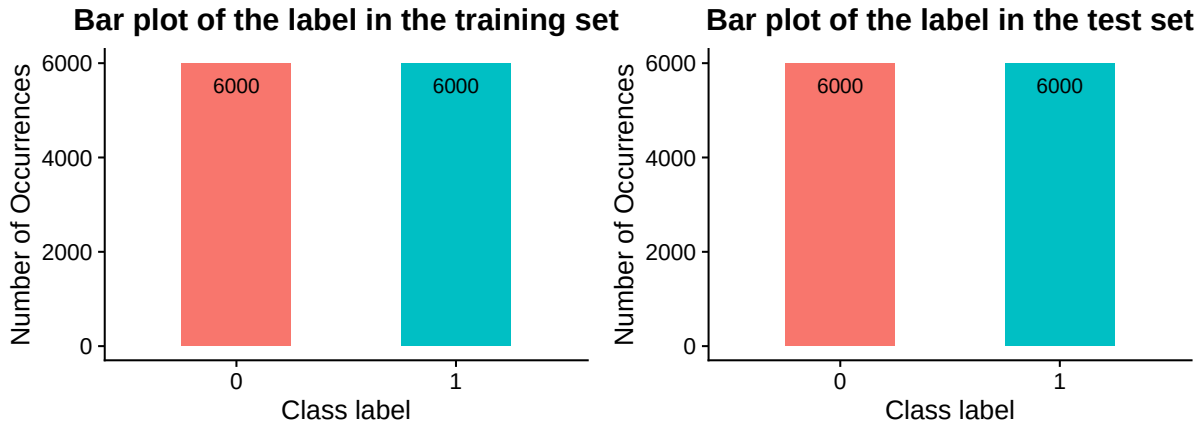
[13] Max Kuhn. *varImp*. URL: https://www.rdocumentation.org/packages/caret/versions/6.0-86/topics/varImp.

[14] Drew Wilimitis. *The Kernel Trick*. URL: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f.

[15] David Meyer. *svm*. URL: https://www.rdocumentation.org/packages/e1071/versions/1.7-3/topics/svm.

A Exploratory Analysis

A.1 Bar plots of the label for both training and test set

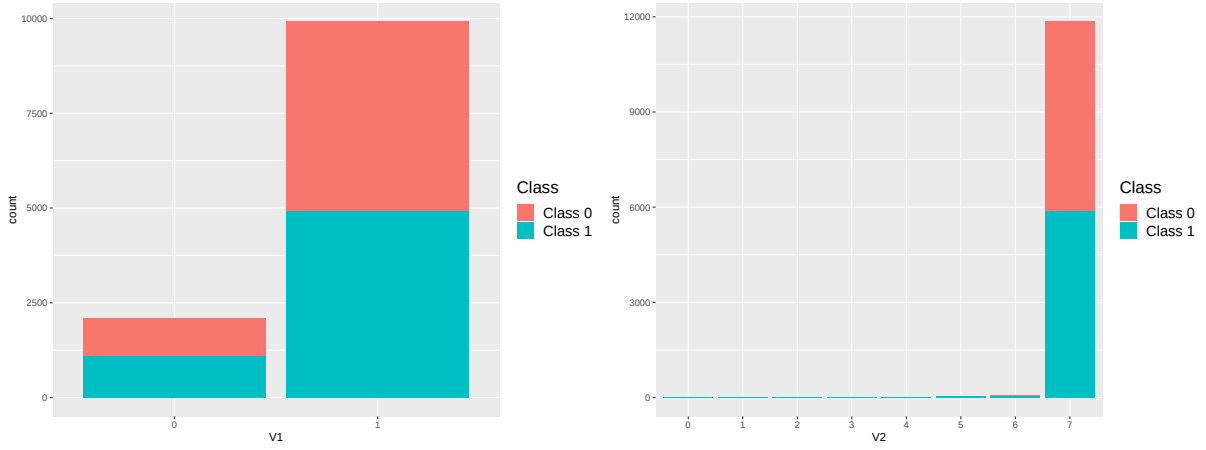


A.2 Table of Descriptive Statistics for the Training Set

	mean	variance	minimum	Q1	median	Q3	max
V7	1.776e-02	8.225e-05	0.000e+00	1.493e-02	1.493e-02	1.493e-02	4.478e-01
V8	1.289e-02	3.232e-04	0.000e+00	0.000e+00	0.000e+00	2.778e-02	4.167e-01
V9	1.576e-03	3.349e-04	0.000e+00	0.000e+00	0.000e+00	0.000e+00	1.000e+00
V10	1.494e-02	2.183e-03	0.000e+00	0.000e+00	0.000e+00	0.000e+00	6.000e-01
V11	6.151e-04	1.372e-04	0.000e+00	0.000e+00	2.388e-04	2.388e-04	1.000e+00
V12	2.251e-08	5.404e-13	0.000e+00	0.000e+00	3.496e-10	5.638e-09	6.480e-05
V13	8.630e-04	1.196e-04	0.000e+00	0.000e+00	1.593e-04	6.370e-04	1.000e+00
V14	2.224e-08	5.278e-13	0.000e+00	0.000e+00	1.049e-09	7.080e-09	6.479e-05
V15	9.950e-05	8.647e-05	0.000e+00	2.122e-34	2.122e-34	2.122e-34	1.000e+00
V16	8.026e-36	5.792e-73	3.543e-36	7.988e-36	7.991e-36	8.005e-36	8.630e-35
V17	9.999e-01	8.471e-05	0.000e+00	1.000e+00	1.000e+00	1.000e+00	1.000e+00
V18	1.250e-03	1.919e-04	0.000e+00	0.000e+00	0.000e+00	0.000e+00	8.371e-01
V19	4.299e-01	1.176e-04	5.618e-03	4.298e-01	4.298e-01	4.298e-01	1.000e+00
V20	3.184e-05	2.498e-13	0.000e+00	3.184e-05	3.184e-05	3.184e-05	5.718e-05
V21	4.279e-01	9.343e-05	0.000e+00	4.265e-01	4.277e-01	4.289e-01	1.000e+00
V22	1.289e-04	2.276e-13	9.351e-05	1.289e-04	1.289e-04	1.289e-04	1.542e-04
V23	9.240e-01	1.449e-06	9.093e-01	9.240e-01	9.240e-01	9.240e-01	1.000e+00
V24	9.578e-01	5.332e-07	9.316e-01	9.578e-01	9.578e-01	9.578e-01	1.000e+00
V25	2.368e-01	6.754e-05	0.000e+00	2.368e-01	2.368e-01	2.368e-01	1.000e+00
V26	1.034e-01	1.892e-13	1.034e-01	1.034e-01	1.034e-01	1.034e-01	1.034e-01
V27	2.657e-01	9.014e-05	1.521e-01	2.655e-01	2.655e-01	2.662e-01	1.000e+00
V28	1.034e-01	2.094e-13	1.034e-01	1.034e-01	1.034e-01	1.034e-01	1.034e-01
V29	9.724e-01	2.224e-07	9.615e-01	9.724e-01	9.724e-01	9.724e-01	1.000e+00
V30	9.823e-01	1.079e-07	9.614e-01	9.823e-01	9.823e-01	9.823e-01	1.000e+00
V31	9.506e-02	1.000e-04	2.090e-02	9.478e-02	9.492e-02	9.507e-02	1.000e+00
V32	5.826e-03	2.820e-11	5.480e-03	5.826e-03	5.826e-03	5.826e-03	5.869e-03
V33	1.744e-01	6.280e-05	1.093e-01	1.741e-01	1.742e-01	1.747e-01	1.000e+00
V34	4.706e-03	2.641e-11	4.399e-03	4.706e-03	4.706e-03	4.706e-03	4.731e-03
V35	8.776e-01	2.875e-08	8.645e-01	8.776e-01	8.776e-01	8.776e-01	8.786e-01
V36	9.215e-01	5.646e-08	9.047e-01	9.215e-01	9.215e-01	9.215e-01	9.224e-01

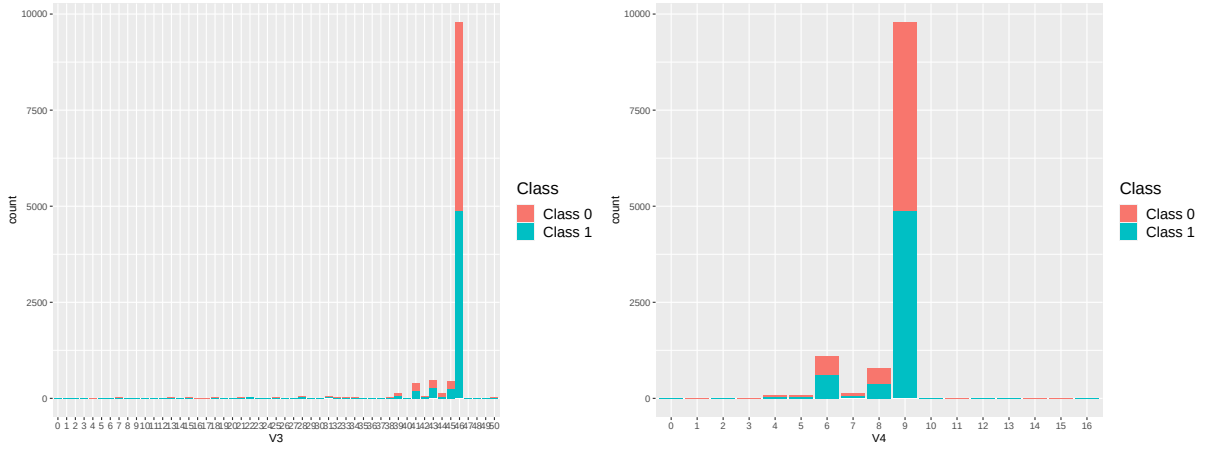
	IQR	skewness	kurtosis
V7	0.000e+00	1.136e+01	4.319e+02
V8	2.778e-02	3.098e+00	4.484e+01
V9	0.000e+00	3.136e+01	1.443e+03
V10	0.000e+00	4.459e+00	2.886e+01
V11	2.388e-04	6.307e+01	4.786e+03
V12	5.638e-09	7.411e+01	5.966e+03
V13	6.370e-04	7.087e+01	5.999e+03
V14	7.080e-09	7.648e+01	6.250e+03
V15	2.200e-38	1.044e+02	1.115e+04
V16	1.764e-38	9.251e+01	9.370e+03
V17	0.000e+00	-1.070e+02	1.160e+04
V18	0.000e+00	3.605e+01	1.675e+03
V19	0.000e+00	1.114e+01	1.087e+03
V20	1.700e-09	-3.469e+00	2.198e+03
V21	2.389e-03	4.719e+00	1.658e+03
V22	4.000e-09	-2.032e+01	3.263e+03
V23	3.255e-06	5.705e+01	3.425e+03
V24	7.815e-06	3.837e+01	2.541e+03
V25	0.000e+00	6.551e+01	6.309e+03
V26	2.000e-09	-2.542e+01	2.602e+03
V27	7.924e-04	5.258e+01	3.785e+03
V28	2.000e-09	-2.438e+01	2.215e+03
V29	3.011e-06	4.360e+01	2.589e+03
V30	5.994e-06	-6.276e+00	2.386e+03
V31	2.943e-04	7.923e+01	6.625e+03
V32	1.900e-08	-4.543e+01	2.451e+03
V33	6.108e-04	9.453e+01	9.821e+03
V34	2.100e-08	-4.450e+01	2.253e+03
V35	1.925e-06	-4.884e+01	3.211e+03
V36	3.672e-06	-4.382e+01	2.533e+03

A.3 Bar Plots for the Discrete Variables



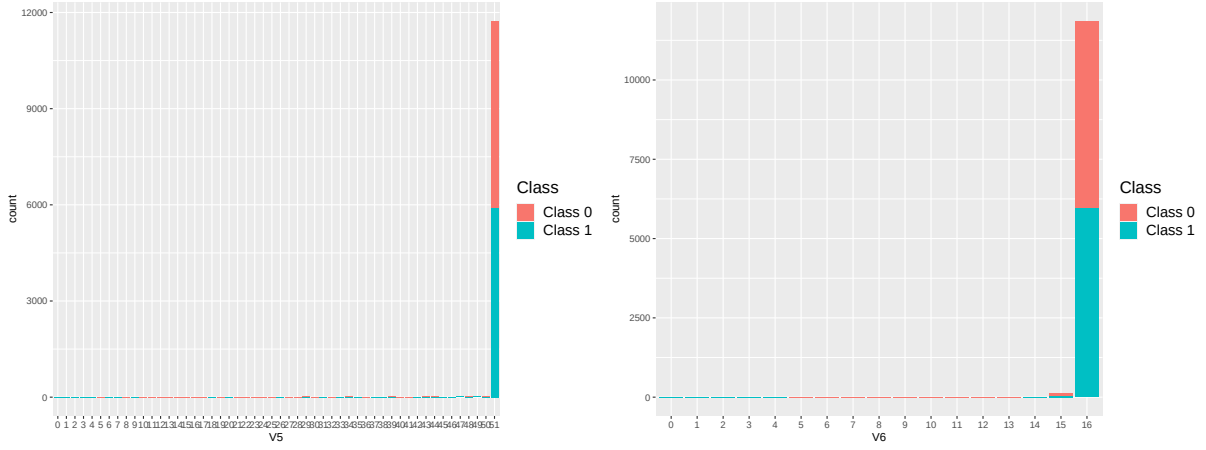
(a) Bar Plot of Variable $V1$

(b) Bar Plot of variable $V2$



(c) Bar Plot of Variable $V3$

(d) Bar Plot of variable $V4$

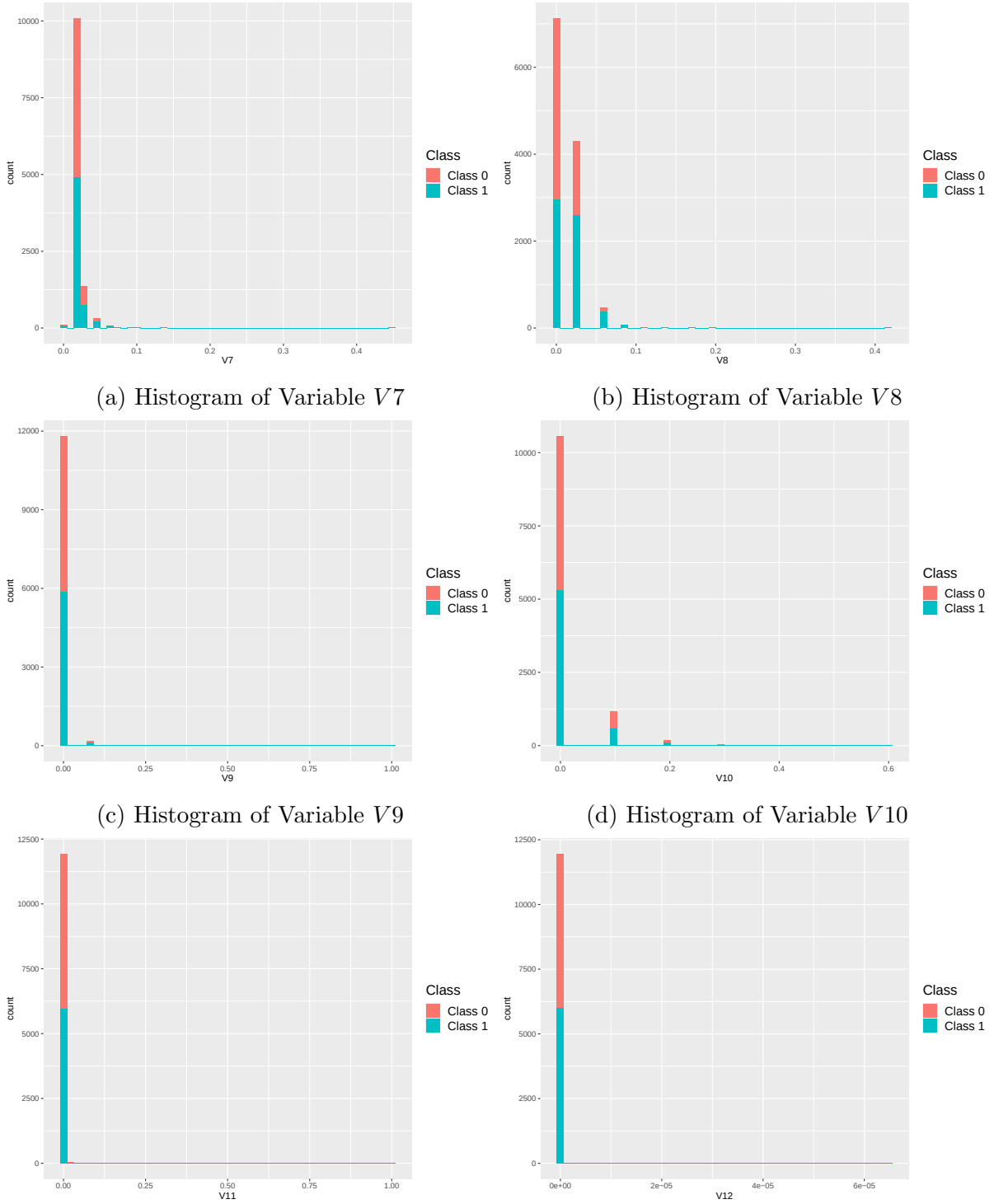


(e) Bar Plot of Variable $V5$

(f) Bar Plot of variable $V6$

Figure 7: Bar Plots for the Discrete Variables

A.4 Histograms for the Continuous Variables



(a) Histogram of Variable $V7$

(b) Histogram of Variable $V8$

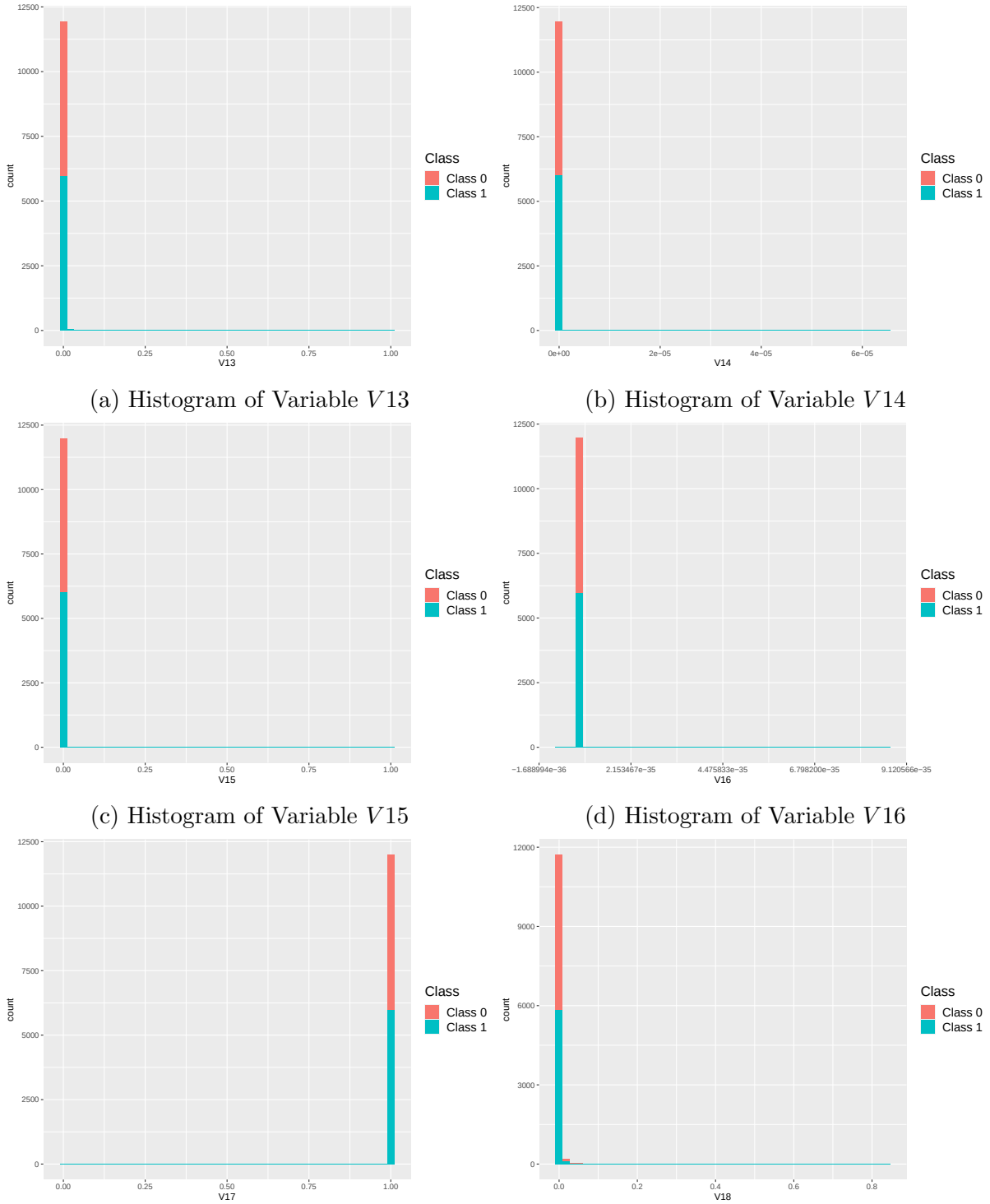
(c) Histogram of Variable $V9$

(d) Histogram of Variable $V10$

(e) Histogram of Variable $V11$

(f) Histogram of Variable $V12$

Figure 8: Histograms for the Continuous Variables $V7$ to $V12$



(a) Histogram of Variable $V13$

(b) Histogram of Variable $V14$

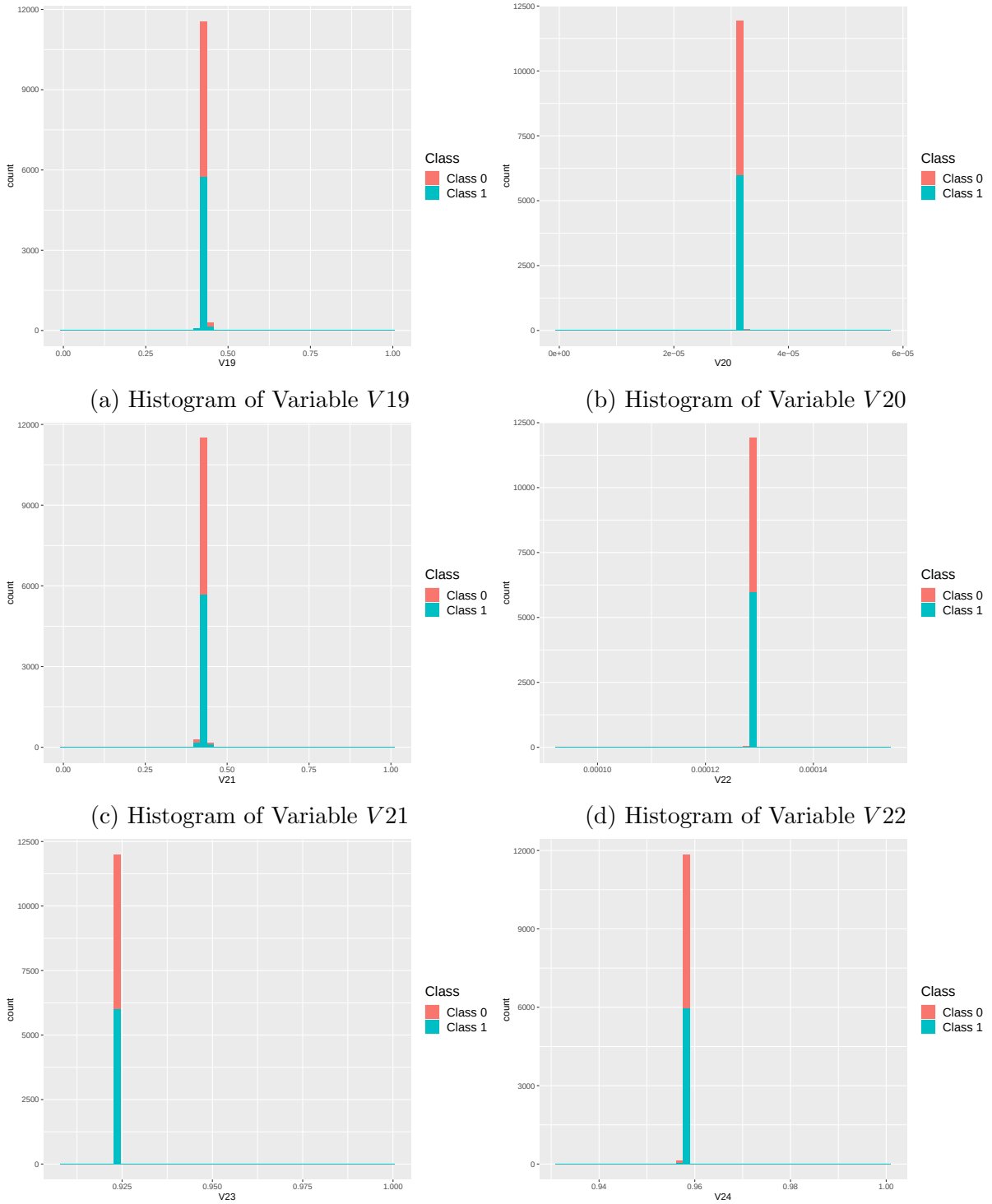
(c) Histogram of Variable $V15$

(d) Histogram of Variable $V16$

(e) Histogram of Variable $V17$

(f) Histogram of Variable $V18$

Figure 9: Histograms for the Continuous Variables $V13$ to $V18$



(a) Histogram of Variable $V19$

(b) Histogram of Variable $V20$

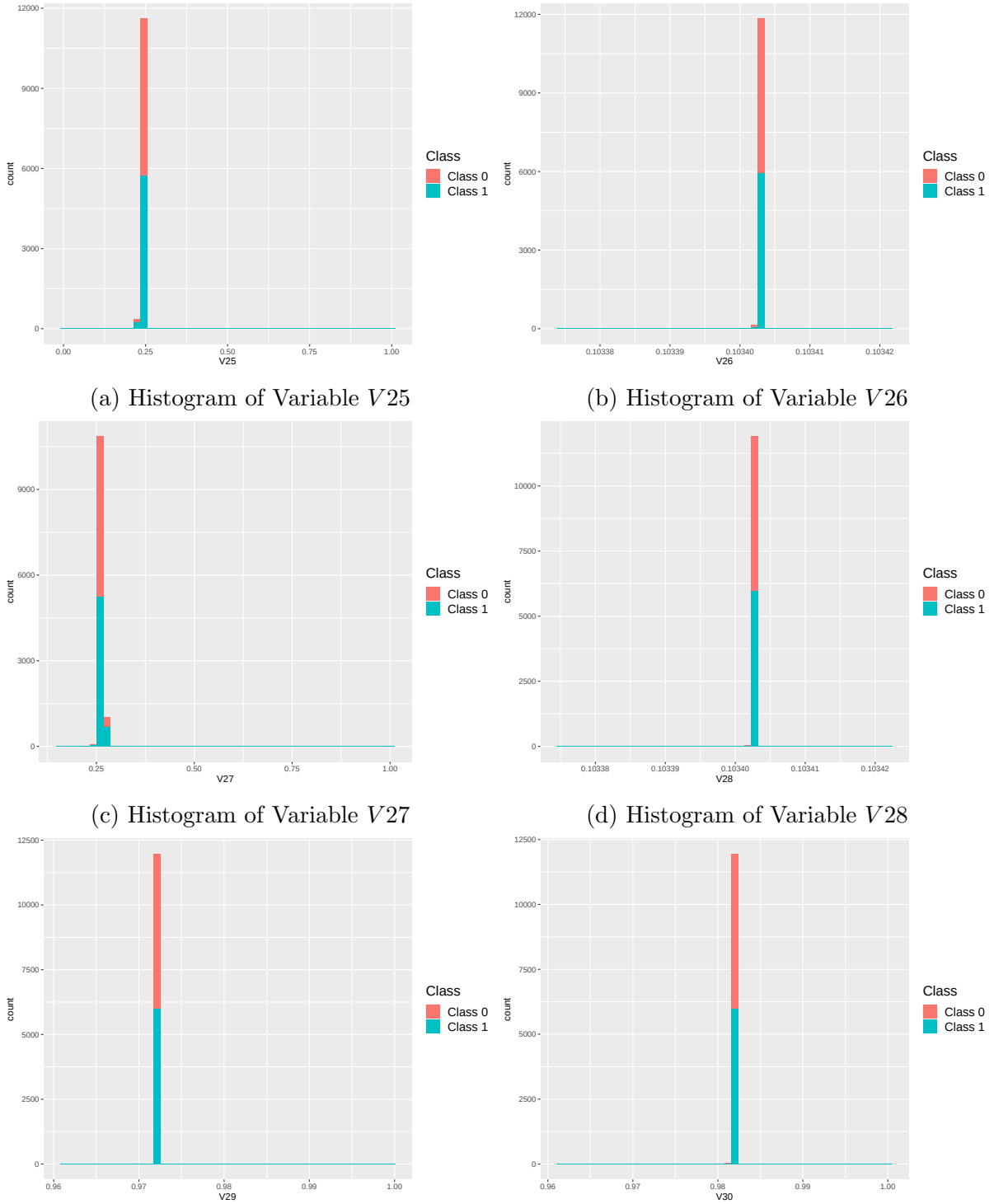
(c) Histogram of Variable $V21$

(d) Histogram of Variable $V22$

(e) Histogram of Variable $V23$

(f) Histogram of Variable $V24$

Figure 10: Histograms for the Continuous Variables $V19$ to $V24$



(a) Histogram of Variable $V25$

(b) Histogram of Variable $V26$

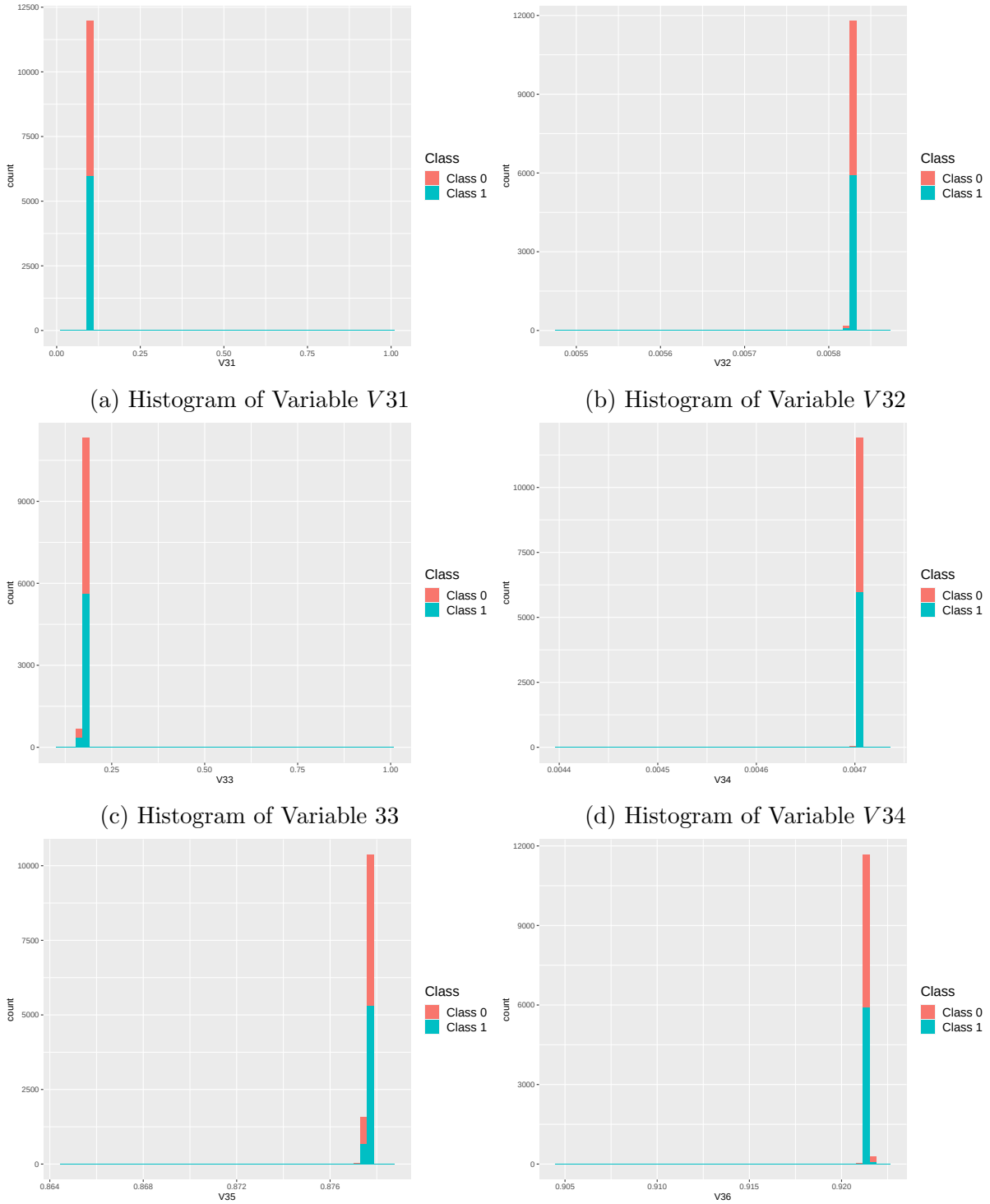
(c) Histogram of Variable $V27$

(d) Histogram of Variable $V28$

(e) Histogram of Variable $V29$

(f) Histogram of Variable $V30$

Figure 11: Histograms for the Continuous Variables $V25$ to $V30$



(a) Histogram of Variable $V31$

(b) Histogram of Variable $V32$

(c) Histogram of Variable 33

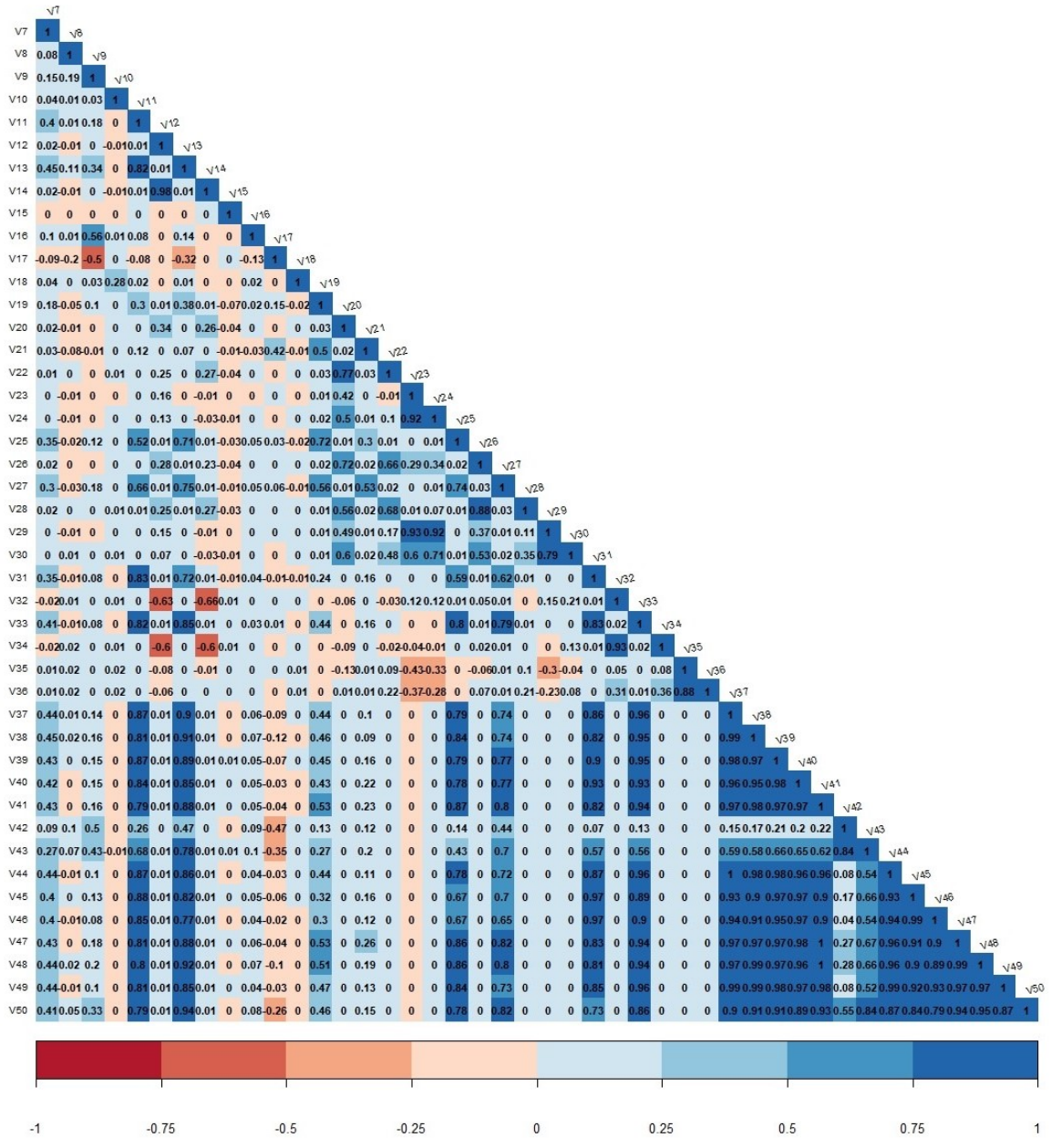
(d) Histogram of Variable $V34$

(e) Histogram of Variable $V35$

(f) Histogram of Variable $V36$

Figure 12: Histograms for the Continuous Variables $V31$ to $V36$

A.5 Correlogram for the continuous variables



B Attachments

The following files contain the code used for the described stages.

1. **Exploratory_train.R** - file containing all the R code used for the Exploratory Analysis.
2. **mi.py** - file with the Python code used to obtain the mutual information heatmap.
3. **Classifiers_RF_LRB_KNN.R** - file that contains the R code to the Logistic Regression Boosted, kNN and Random Forest classifiers. It has information about *how to run it*.
4. **pca_medm.R** - file that contains the R code to the Principal Component Analysis;
5. **svm_medm.R** - file that contains the R code to the Support Vector Machine classifier;
6. **bayes_medm.R** - file that contains the R code to the Naive Bayes classifier.