Using Multivariate Analysis and Machine Learning to detect Defaulting Payments of Credit Card Clients in Taiwan in 2005

Group 11

João Novo 90113, Alexandre Ferreira 87144, André Roque 86694

Instituto Superior Técnico - Universidade de Lisboa

Abstract

The main goal of this project is to predict if a customer defaults on their credit card payment in the following month. Methods: We aim to solve this binary classification problem using supervised learning methods such as Naive Bayes, Decision Trees, Random Forests, Support Vector Machine and Linear Discriminant Analysis and unsupervised learning methods such as Hierarchical Clustering, K-means and K-Medoids.

For supervised learning methods, the best model was Random Forest with 93.28% accuracy. For unsupervised learning, the best model was a standardized K-Means model with the Gower distance.

None of the utilized models reached a level of accuracy that would give strong a strong incentive to be used with almost complete certainty. However, supervised methods tended to be quite better than unsupervised methods, so with more data and subsequently more finely tuned supervised models, it would be foreseeable that accuracy could increase to more satisfactory levels.

Keywords: Credit, Default-payment, Dimensionality Reduction, Naive Bayes, Decision Trees, Random Forests, Support Vector Machine, Linear Discriminant Analysis, Principal Component Analysis, Minimum Redudancy Maximum Relevance, Hierarchical Clustering, K-means and K-Medoids

1 Introduction

In modern society the use of credit cards is a common practice and one of the main business of banks. For banks, it helps the bank to generate interest revenue but at the same time, it raises the liquidity risk and credit risk to the bank. In order to both control the cash flow and risk, detecting the customers with default payment next month could play an important role on estimating the cash flow and assessing the risk. The dataset under study gathers data collected in 2005 in Taiwan. The goal of this work is to perform a classification task to predict whether or not a customer is said to default or not in the following month. Since from the perspective of risk management, the predictive accuracy of the binary result of classification is of value. After a preliminary exploratory data analysis, which allowed for an initial insight on which features could be more useful for the prediction of payment default in the next month, the dataset was analysed by means of dimensionality reduction methods, supervised learning methods and unsupervised learning methods.

1.1 The Original Dataset

The original dataset has 30 000 observations (rows) and 25 variables: 1 'ID' column, 14 numerical variables, 9 categorical variables and one binary target variable (the binary classification of defaulting payment next month (dpnm)). A more detailed explanation of the features is presented in Table 6. The categorical variables have different levels and numeric variables include variables with negative values and others without, plus a wide range of magnitude.

Making use of the correlation heatmap found in the appendix under 16 it is also possible to infer a great correlation between the variables corresponding to the amount of bill statements across all months.

2 Preprocessing and Exploratory Data Analysis

In order to cleanup the data and prepare it for analysis some steps were taken. The column 'ID' was removed since it didn't provide any insight as each row represented a different client.

Next the columns were renamed for better readability, according to the following:

- credit_limit = LIMIT_BAL
- gender = SEX
- education = EDUCATION
- marital_status = MARRIAGE
- age = AGE
- payment_status_sept = PAY_0
- payment_status_aug = PAY_2
- payment_status_jul = PAY_3
- payment_status_jun = PAY_4
- payment_status_may = PAY_5
- payment_status_apr = PAY_6
- bill_sept = BILL_AMT1
- bill_aug = BILL_AMT2
- bill_jul = BILL_AMT3
- bill_jun = BILL_AMT4
- bill_may = BILL_AMT5
- bill_apr = BILL_AMT6
- payment_amount_sept = PAY_AMT1
- payment_amount_aug = PAY_AMT2
- payment_amount_jul = PAY_AMT3
- payment_amount_jun = PAY_AMT4
- payment_amount_may = PAY_AMT5

- payment_amount_apr = PAY_AMT6
- default_payment = ‘default payment next month’

In addition, to have a better grasp of the range of the input values the table 7 (in the appendix) presents the min, max, mean and standard deviations for each feature of the dataset.

The following problem was spotted on the original dataset: clients with no amount to be settled in columns ’Bill_Amt1’ to ’Bill_Amt6’ had default payment flagged 1. This is illogical, since clients that have no credit bill to repay the bank can’t default on their payment. Therefore, observations matching this case were corrected.

In the feature ’education’ the categories 4, 5 and 6 were merged into the category ’0’ meaning ’others’, since unknown and others add no value to the understanding of the feature.

By the use of plots we can also see that around 78% of customers are not defaulted, while only 22%p of customers have a default payment next month, Figure 1. Therefore showing that the data is unbalanced.

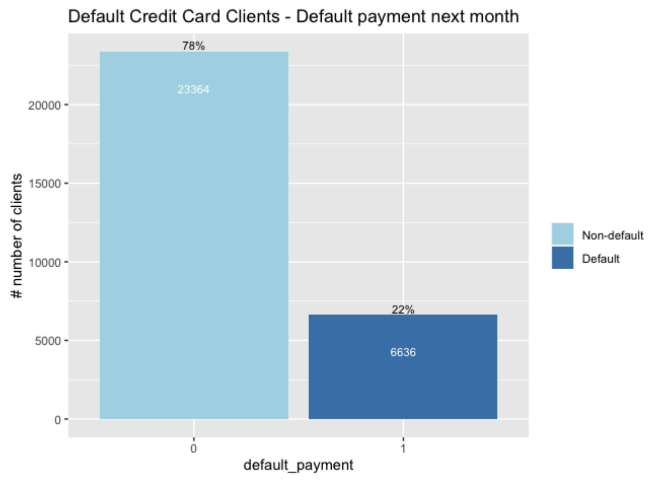


Figure 1: Default payment distribution in the dataset.

The most frequent amounts of credit loaned are in the interval of 5k to 50k NT$ and that’s the where most percentage of people default, Figure 13. The loans are given mostly to customers aging from 20 to 40, maybe due to the lower risk of defaulting payment (likely to live till old age, working class) or the amount of loans requested by this age range, Figure 2.

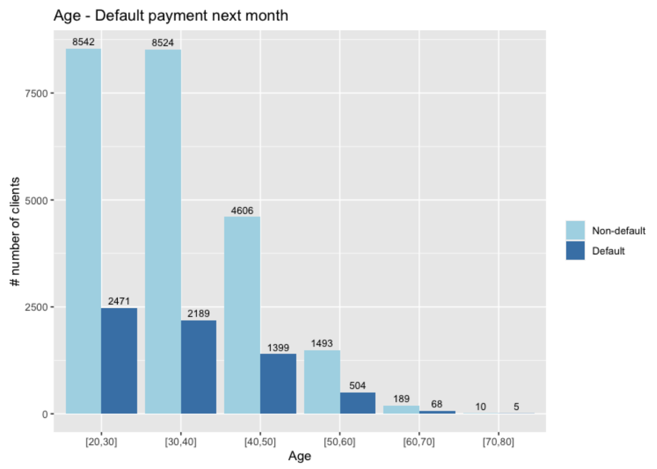


Figure 2: Default or not distribution by age intervals.

As well as some other insights in figures: 14 13 15.

At last, we draw a correlation plot of the input variables, Figure 16. High correlations can easily be found by the color legend. We spot high positive correlations for: bill_april to bill_sept since the amount of the bill from month to month is logically related.

3 Dimensionality Reduction

Due to the presence of a considerable amount of ordinal, numeric and binary features on our initial dataset, we decided to apply dummyfication to the categorical ones, obtaining a total of 42 features. After trying some models with the original dataset, the models weren’t performing as expected mainly because of two reasons:

1. The presence of binary features that contained only a very small amount of positives

2. The presence of highly correlated features among the whole set of new features

Due to this reasons, some of our models weren’t even generating a output. Hence we decided to perform a a feature selection based on two unsupervised methods: Principal Component Analysis (PCA) and minimum Redudancy Maximum Relevance (mRMR).

3.1 Mixed PCA

PCA avaliates the covariance matrix chooses a linear combination of features in such a way that each PC explains the maximum variance possible. Thus, by performing a transformation of the space of our observations it generates a new space of features with the same dimension as the initial one. Given the characteristics of our dataset, we decided to use the R library PCAmixdata[1], that combines the use of the PCA for numerical features and Multiple Component Analysis (MCA), a similar method to PCA, but used mainly for binary variables. After applying this method, we obtained the screen plot on figure 3 , assignig to each number of of PC’s the cumulative variance explained.

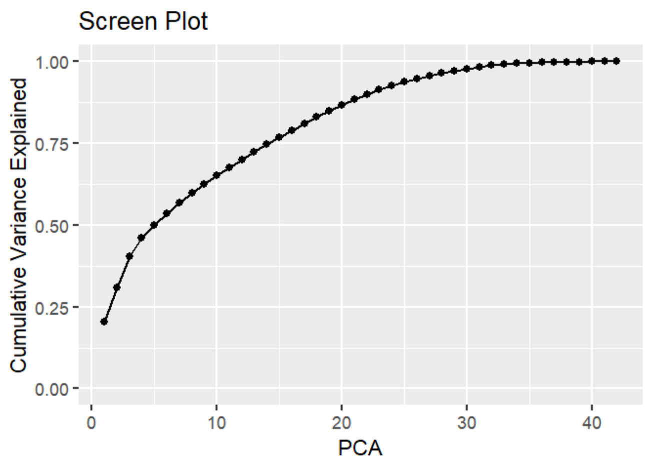


Figure 3: Cumulative variance explained for each number of PC’s

We considered to retain 17 PC’s since they are enough to explain 81.6% of the variance. Also, by colsely analysing the PC's loadings, we observed that every feature was present in a relevant proportion in at least one of the 17's PC's, confirming the choice to keep these 17 new features.

3.2 mRMR

For the mRMR, we used the package mRMRe [2]. Given a pre-choosen number n of features, the algorithm uses the Mutual Information Matrix to select a predetermined number of features. 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 [3]. Since we obtained 17 feaures using the PCAmix method, we decided to set the algorithm to choose n = 17 in order to have a more fair comparison between the results of each method for each type of feature selection.

This is the list of the features it chose: *payment_amount_jun, education.0 , payment_amount_may , payment_amount_apr, payment_status_jul.1, marital_status.2, payment_status_may.2, payment_status_jul.2, payment_status_sept.2, bill_sept, payment_status_apr.2, education.1, payment_status_aug.2, payment_status_jun.2, marital_status.3, payment_status_aug.3, payment_status_sept.3.*

4 Dataset Balancing

Our dataset was imbalanced, since default payment, i.e., our response variable, was identified on less than 22% of the clients. 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 for that reason will not be able to accurately predict new data during the application of the trained model. Therefore, the train set must have a more satisfactory representation of the classification problem with a wider range of examples of all classes.

Additionally, we decided to use the entire dataset for training and testing, in order to achieve a more fair comparison between each model. Thus, we splitted the datasets obtained from the dimensionality reduction process into two different datasets: the train set, used to train the classifiers, and the test set, used to assess the performance of the classifiers.

Since our data is imbalanced, we must ensure a fairer splitting, in other words, that we have a more 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 [4].

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 (that is, default_payment =0).

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 [5] and [6], 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 [7].

Moreover, while our main goal was to balance the dataset, we still wanted it to have as many true observations as possible, without biasing one of the classes. 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. Since we used a dataset containing a very high number of observations (30 0000), we expected that by applying this technique the results obtained wouldn't be biased and the loss of information to be minimal. After applying this method, the percentage of positive observations (i.e, default_payment=1) raised to nearly 30%.

5 Performance Measures

When evaluating the performance of the tested classifiers, we took into consideration the fact that the test sets was 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 for evaluating the 2 classes of our target variable.

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 client would be in a default payment situation. This means that, in our context, a true positive represents a client having a default payment that is identified as such by our model. Conversely, a false negative represents a client which, despite having a default payment, our model classifies as not. The latter case is known as an error of type II.

Since our objective is to predict a default payment, in order to provide the information for an eventual prevention of liquidity problem to the credit provider , we claim that it is preferable that the client is wrongly identified to be in debt rather than having an error of type II, since the latter can have serious consequences in assessing potential losses. 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.

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 specificity [1], was also taken in consideration.

At last, we also decided to take in account the Area Under the (ROC- Receiver Operating Caracheteristics) Curve (AUC), that measures how much the model is capable of distinguishing between classes. The higher the AUC is, the better the model is at predicting 0 classes as 0 and 1 classes as 1. By measuring the area under the ROC curve, an evaluation metric for binary classification problems that represents the probability curve that plots the True Positive Rates against False Positive Rates at various threshold values and essentially separates the 'signal' from the 'noise'. This measure is very specially useful when evaluating models over unbalanced datasets.

To conclude the performance metrics used are: Accuracy, Balanced Accuracy, Precision, Sensitivity, Specificity, AUC.

6 Supervised Learning

Supervised learning methods consist of predicting responses given a known data set, knowing a priori the classes/groups, in this case: default (1) or no-default (0). For this, we need the dataset was divided in training and test set, as previously explained in 4.

Both the PCA dataset and the mRMR dataset were testes with the following classifiers. In addition, we computed extra results for the cases where it was considered that 2 classes with probability 0.4 would imply a correct choice by the model.

The results in full extent are present in the Appendix in tables 8, 9, 10, 11.

6.1 Naive Bayes (NB)

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`. The main downside of this algorithm is that it assumes conditional independence among the features and if that is not true, the accuracy of the classification decreases. Similar to the other models, this classifier was trained using the selected features for algorithms tested in Section 3.

Naive Bayes model's perfomance is not good in this case as our case cannot satisfy one of the assumption of Naive Bayes model: assumption of independent predictors. The model implicitly assumes that all the attributes are mutually independent but our payment status are positively correlated to each other. On the other hand, Zero Frequency may happen when categorical variable has a category is not observed in training data set but in the test set, then the model will assign a zero probability and will be unable to make a prediction. In short, Naive Bayes does not work well with categorical variables.

6.2 Decision Tree (DT)

A decision tree is a graphical representation of possible solutions to a decision based on certain conditions. It is called a decision tree because it starts with a single variable, which then branches off into a number of solutions, just like a tree. A decision tree has three main components :

- **Root Node :** The top most node is called Root Node. It implies the best predictor (independent variable).

- **Decision / Internal Node :** The nodes in which predictors (independent variables) are tested and each branch represents an outcome of the test.

- **Leaf / Terminal Node :** It holds a class label (category) - Yes or No (Final Classification Outcome).

In this case a decision tree is computed with pruning to correct overfitting stopping the tree growth when there is no statistically significant association between any attribute and the class at a particular node using chi-squared test to check statistically significant association.

The cost complexity was the chosen post-pruning method. It is measured by the Number of leaves in the tree and the misclassification rate or Sum of Squared Error. The 'CP' stands for Complexity Parameter of the tree. We want the cp value of the smallest tree that has smallest cross validation error. In regression, this means that the overall R-squared must increase by cp at each step.

In this case, we pick the tree having CP = 0.01 as it has least cross validation error (xerror) as seen by the table bellow:

CP	nsplit	rel	error	xerror	xstd
1	0.171162	0	1.00000	1.00000	0.0099778
2	0.022878	1	0.82884	0.82898	0.0094041
3	0.010168	2	0.80596	0.80610	0.0093148
4	0.010000	3	0.79579	0.80215	0.0092990

The relative error of each iteration of the tree is the fraction of misclassified cases in the iteration relative to the fraction of misclassified cases in the root node.

The final decision tree takes the shape and values shown:

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

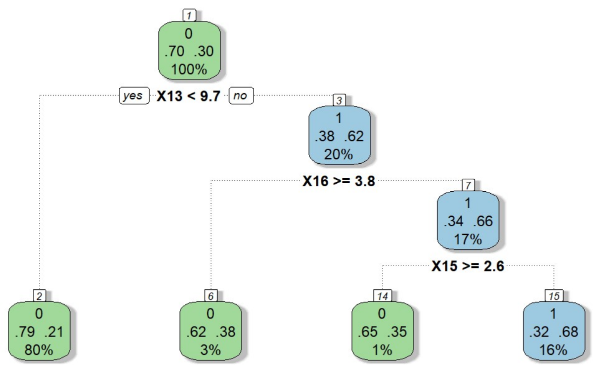


Figure 4: Decision tree

The variables X13, X15 and X16 ended up being the variables used in the tree construction and the obtained confusion matrix indicates an accuracy of 76.2%. By studying more in depth the squared loadings of the PCA we can associate the X13 variable with the original variable payment_status_aug2. By doing the same thing for the other variables X15 associates mainly with payment_status_may.1 and the X16 variable with payment_status_jun.0.

6.3 Random Forest (RF)

A random forest consists of aggregating the results of various decision trees. On Random Forest the individual trees work with a random subset of all the features available contrasting with normal decision trees where while splitting a node, all the possible features are considered and is chosen the one that produces the better separation between the left and right nodes created, which makes the trees to be less correlated among themselves. That avoids a common risk in decision trees, error propagation. The ability of random forests to limit over-fitting without substantially increasing error due to bias partially explains why they are such powerful models. Another benefit is not being sensitive to outliers. Essential to an accurate model is the tuning which is based in finding the ideal number of trees and the number of random variables used in each tree.

A drawback Random forests have is a bias towards the categorical variable having many categories. This problem is avoided when using a redimensioning process with PCA since the final dataset only has numerical variables. The correlation between any two trees in the forest is also important because increasing the correlation increases the forest error rate. Moreover the strength of each individual tree in the forest impacts the final result in a way that a tree with a low error rate is a strong classifier. Proof of this superior potential of the random forests method is reflected on the obtained results with 93% accuracy 8.

We also used a specific random forest library [8] in order to tune it more in depth.

6.4 Support Vector Machine (SVM)

The SVM algorithm essentially finds an hyperplane which divides data into two classes. It looks for points, called support vectors, that are located directly to the dividing line closest. Then, the algorithm calculates the gap between the reference vectors and the dividing plane. The main goal here is to maximize the clearance distance and make the gap as large as possible.

Looking at the final results SVM performs in line with other models but at the same time performs poorly in terms of sensitivity and highly in terms of specificity, suggesting a high number of false negatives and a low number of false positives. This means that SVM tended to classify the majority of the observations as not defaults (0). As this is the majority value of the target value the accuracy is good, but obviously not reliable for the evaluation of the model.

6.5 K-Nearest Neighbors (KNN)

The KNN algorithm assumes that similar things exist in close proximity. KNN classifies objects based on the objects that are most similar to it. It uses test data to make an "educated guess" on what an unclassified object should be classified as.

We applied the k-Nearest Neighbors algorithm classifier to our data by using the R library `caret` ([9]). The implementations of the algorithm selected a number of neighbors(k) equal to 25 as the best option, which can be explained by the imbalance of our data set. Given that we have more negative than positive values, the probability of being near negative class neighbors is higher. Therefore, it is preferred a smaller number of nearest neighbors in order to increase probability of being close to positive class neighbors.

6.6 Linear Discriminant Analysis (LDA)

Linear discriminant analysis is used as a tool for classification, dimension reduction, and data visualization. It aims to maximize the distance between the mean of each class and minimize the spreading within the class itself. LDA uses therefore within classes and between classes as measures. This is a good choice because maximizing the distance between the means of each class when projecting the data in a lower-dimensional space can lead to better classification results. While using LDA, it is assumed that the dataset follows a Gaussian Distribution.

6.7 Results

It can be seen that for all the machine learning models trained for both the PCA and mRMR input datasets when it is considered that when there are 2 classes with probability higher than 0.4, the model improves.

Firstly, analysing the performance, the classifiers that achieved the greater accuracy were Random Forests and LDA. Since the PCA analysis allowed to reduce the complexity of the data we can say that, doing the PCA is a good approach for the problem. It can even be seen that the mRMR dataset achieved a slight better result than the PCA dataset despite both having 17 features in consideration.

In the tables below we can analyse the accuracy of every model for both the PCA and mRMR datasets. Nevertheless, it should not be taken as the only factor of how better a model is compared to another, therefore for a further and better analysis one is advised to look into tables 8, 9, 10, 11..

	Accuracy - PCAmixdata	
Classifier	Normal	p > 0,4
Naive Bayes	0.7971865	0.8070447
SVM	0.766615	0.7934205
LDA	0.8114754	0.8373948
KNN	0.8025033	0.8443731
DT	0.7617299	0.8067125
RF	0.932855	0.8495791

Table 1: Accuracy for each classifier with the PCA dataset as input

	Accuracy - mRm	
Classifier	Normal	p > 0,4
Naive Bayes	0.7988333	0.8073333
SVM	0.7815	0.8071667
LDA	0.8333333	0.8521667
KNN	0.8185	0.8626667
DT	0.77525	0.8625
RF	0.9235757	0.8511667

Table 2: Accuracy for each model with the mRm dataset as input

7 Unsupervised Learning

In this section unsupervised learning methods were applied to the data. The goal is to find in the clustering algorithms some correlations indicating wether a customer is due to default or not. Firstly, it was used a hierarchical method. Then, partition methods such as K-Means and K-Medoids were also applied.

The dataset used was reduced to 3000 observations instead of the original 30 000, keeping the original balancing (78-22) and once more both the PCA and mRMR datasets were tested. This was do to computing capabilities.

For the computation of the similarly matrices, we used the Euclidean distance for the features created by the PCA, while for the ones choosen by the mRMR algorithm, since some of them were binary, we preferred to use the Gower distance. This latter distance is the average of partial dissimilarities among observations, ranging between 0 and 1. The partial dissimilarities are calculated accordingly to the type of variable: for numerical features, the partial dissimilarity is the ratio between the absolute difference of observations and the maximum range observed considering all individuals; for a qualitative feature, the partial dissimilarity is equal to 1 if two observations have a different value, and 0 otherwise.[10]

7.1 Hierarchical Clustering

Hierarchical clustering is a methods that groups similar objects into groups called clusters. It does so by analysing the distance if each point, according to a certain proximity criterion, and grouping them one by one by assigning each object to the nearest cluster. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, but where the objects within each cluster are broadly similar to each other.

In this context a hierarchical clustering model should be computed by unlabelling the dataset (removing the target label). Then, after the model clusters the data with the remaining features, the classification feature should again be reintroduced to then test the accuracy of the model with respects to its cluster output. Additionally, since we want the output to be comprised of a binary result (default or no-default), we can set the clustering model into giving us two final clusters.

The methods used for hierarchical clustering were Single Linkage, Complete Linkage, Average Linkage, and Ward's Method. An euclidean distance was used for the PCA dataset and a gower distance for the mRMR dataset. This method was chosen based on the comparison of results obtained and the fact that the existance of binary variables in the mRMR dataset is more adequately measure using this method. The dendogram of the method which produced the best results, according to tables 12, 13, for the mRm prepocessed dataset is shown below:

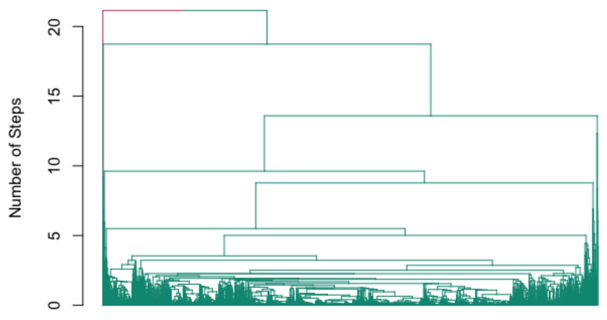


Figure 5: Hierarchical Clustering with 'Average' method for mRMR dataset

7.2 Non-Hierarchical Clustering

7.2.1 K-Means

While using K-means before the actual clustering is done the optimal number of clusters (k) for the data set needs to be identified. The used method for determining the ideal number of clusters were the elbow technique, the gap statistics technique and the silhouette technique, both for the PCA dataset and the mRMR dataset. The used distance to compute the distance between points was the Gower distance, due to the logic explained previously.

The output of the silhouette method for the PCA dataset is shown bellow since it was the one producing better results:

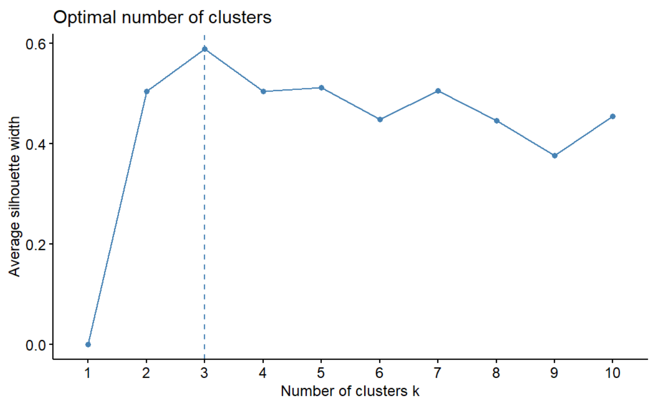


Figure 6: K-Means silhouette for getting optimal number of clusters with the PCA dataset

It is easy to identify the optimal number of clusters will be 3. It might sound irrational to apply 3 clusters for a dataset with a binary output but it can prove very effective. By labelling an intermediate cluster of observations as being in a grey area in terms of classification it can be extrapolated into a group of clients whose default probability is not negligible but at the same time is not high enough to be labelled as one to default.

The next step was to apply the actual clustering by initializing the centroids (center-point) of each cluster randomly and assign each observation to the closest centroid by calculating least squared euclidean distance between centroids and observations. In each iteration new means are calculated as centroids for new clusters. The process is repeated until convergence or maximum iteration is reached.

In this case the plot obtained does not show clear and easy spotting of clusters in a two dimensional representation. Although such a plot can be misleading for a dataset with 17 dimensions the results obtained when comparing the true labels of each observation with the one from the clustering do not show encouraging results as suspected.

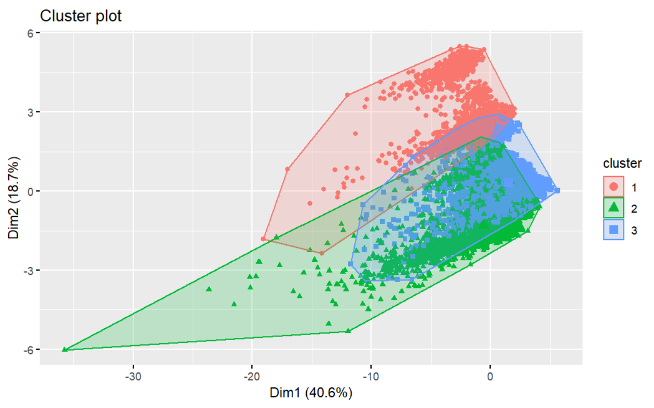


Figure 7: K-Means with 3 centroids on the PCA dataset

Next, the same methods for discovering the optimal k are applied to the mRMR dataset. This time it is presented the gap statistics obtained which indicates towards a sole cluster as the better option to study. Since this has no interest for the study being performed and by comparing the frequency of observations with two or three clusters we opted for a two cluster approach because the numbers could clearly be more rationally applied than a three cluster approach in this case.

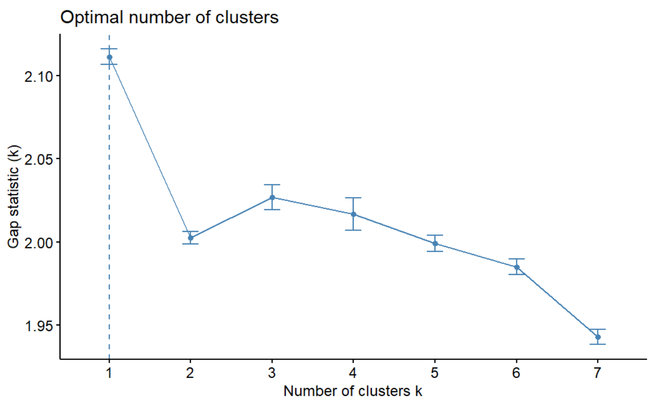


Figure 8: K-Means gap statistics for getting optimal number of clusters with the mRMR dataset

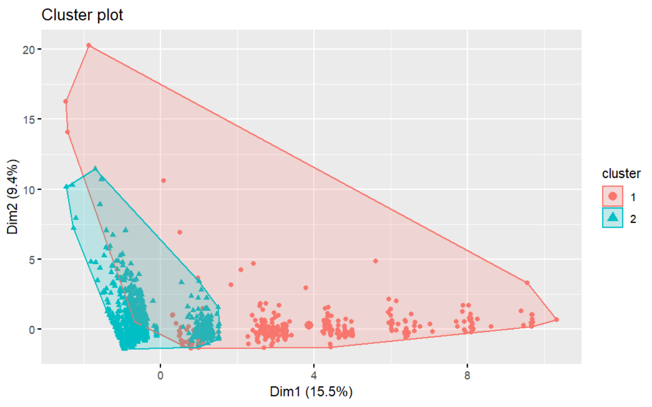


Figure 9: K-Means with 2 centroids on the mRmR dataset

The frequency of the results for each dataset clusters is presented in the following table:

	K - means	
Clusters	PCA	MRmR
Cluster 1	0.144	0.107
Cluster 2	0.375	0.893
Cluster 3	0.481	—-

Table 3: Frequency percentage of each cluster using both datasets

By briefly analyzing these results and comparing with the frequencies of the target variable in both datasets it is easy to associate the value 0 with cluster 2 and value 1 with cluster 1 in the mRMR dataset. Making the same analysis on the cluster for the PCA dataset it is deductible that cluster will associate with no default, cluster 2 falls in a grey area and cluster 1 is clearly associated with clients going to default.

7.2.2 K-Medoids

The idea of K-Medoids clustering is to make the final centroids as actual data-points. This result to make the centroids interpretable. The algorithm of K-Medoids clustering is called Partitioning Around Medoids (PAM). K-Medoids computing method is very similar to K-Means except that with K-Means we were computing mean of all points present in the cluster. But for the PAM algorithm updation of the centroid is different. If there are m-point in a cluster, swap the previous centroid with all other (m-1) points from the cluster and finalize the point as new centroid that have a minimum loss.

K-Medoid Algorithm is fast and converges in a fixed number of steps and PAM is less sensitive to outliers than other partitioning algorithms. However, K-Medoid algorithm is not suitable for clustering non-spherical (arbitrary shaped) groups of objects. This is because it relies on minimizing the distances between the non-medoid objects and the medoid (the cluster centre) – briefly, it uses compactness as clustering criteria instead of connectivity. Moreover, it may obtain different results for different runs on the same dataset because the first k medoids are chosen randomly. The obtained results using the euclidean distance to construct the clusters, has two distinct clusters for the mRMR dataset and three for the PCA dataset are shown in the following plots:

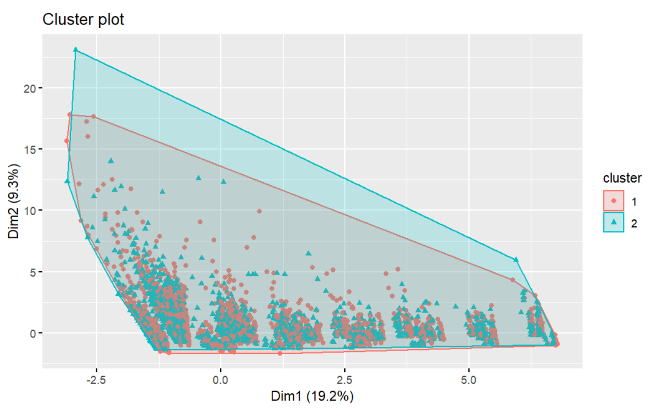


Figure 10: K-Medoids with 2 centroids on the mRmR dataset

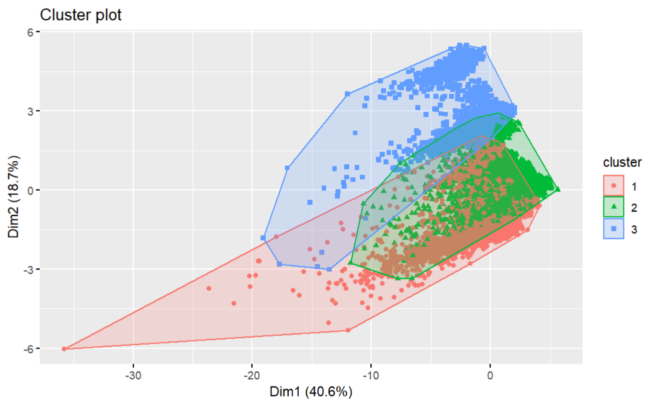


Figure 11: K-Means with 3 centroids on the PCA dataset

	K - medoids	
Clusters	PCA	MRmR
Cluster 1	0.468	0.468
Cluster 2	0.379	0.5324
Cluster 3	0.153	—

Table 4: Frequency percentage of each cluster using both datasets

The results in this case are similar of those obtained in K-Means in the PCA dataset, only with a change of order between clusters 1 and 3. The results with mRMR on the other side were poor with the clusters comprehending almost a 50/50 ratio of observations.

8 Classification on clusters labels

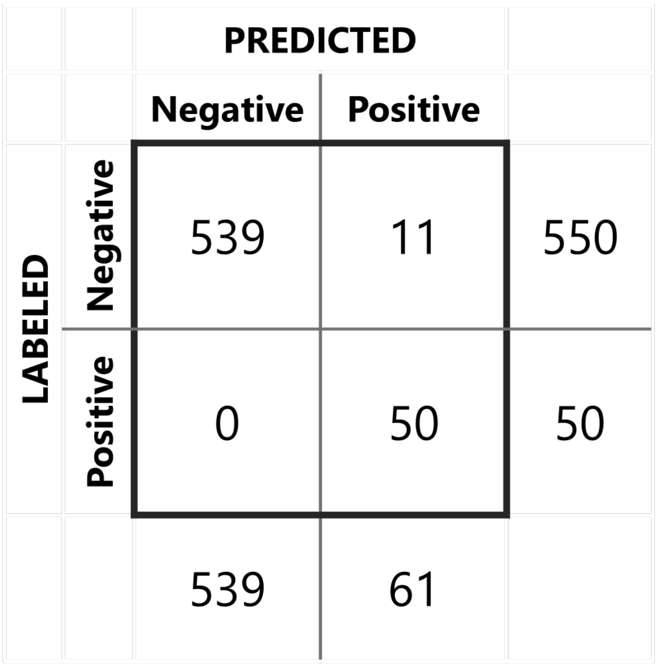
The last point of our analysis consists in getting the cluster labels in the partioning models obtained and use them as a target variable for the train datasets we had unlabeled. After that the goal was to choose a method of supervised learning that had produced good results with the original default payment variable and train it with the new labels. The goal here was to compare the results produced between this combination of clustering and supervised learning and a more traditional approach.

To start it was chosen the K-Means labels obtained with the mRMR model and applied the KNN supervised learning method, given this was one of the best performing methods in our study. The number of observations chosen was 3000 for the traning and 600 for testing, for computability reasons. The results obtained are in the table bellow:

	— K – means
Metrics	PCA
Accuracy	0.98
Precision	1
Sensitivity	0.82
Specificity	1
AUC	0.51

Table 5: Frequency percentage of each cluster using the PCA dataset

The confusion matrix obtained is the following:



		PREDICTED		
		Negative	Positive	
LABELED	Negative	539	11	550
	Positive	0	50	50
		539	61	

Figure 12: Confusion matrix of the KNN model applied to K-Means clusters labels

The results obtained seem very encouraging with the final model getting right all the labels predicted as negative. As we have been seeing throughout our work the biggest difficulty is indeed predicting the clients that will default on their payments correctly, predicting the target label '1'.

Unfortunately, the application of this process in the PCA dataset was not possible since the number of clusters exceeded the binary options of the original target variable, making it impossible to know which clusters to assign to default or no default in the payments.

9 Conclusions and Future Work

This project aimed to study what at first seemed a simple, straightforward dataset. The reality came to prove very different as the absence of missing values and plentitude of observations became something of a burden as we tried to train our models with as many observations as possible. This stance led to long periods of computation and many R Studio sessions aborted and periods where our computers simply ran out of available memory, with some data frames reaching as much as 280 million elements. However, this was a choice made by our group who saw in this the opportunity to work with a rich dataset as a duty to deliver results. A second point that surprised was the fact that the high dimensionality of the data, together with varied scales and magnitudes and binary and categorical features of multiple levels meant hard work if we wanted good results in the classifiers we had to excel at the pre-processing stage. A lot was learned in this part as the area of feature engineering is full of in depth solutions, often complex to implement but not all worthy of good results. In the end, we believe the two methods chosen were adequate for this project. The PCA implementation particularly proved itself really tricky to cope with due to the variety of categorical features. We ended up learning a lot while trying to grasp where to normalize data, where to scale, where not scale, what dummy variables to create and the techniques used to stratify it.

Moving on to the actual implementation of the classification methods we tried to potentiate our results by covering a wide range of supervised methods and evaluate them with a good number of performance metrics, allowing for a deep comparison between models and techniques. Overall, the results were much better than anticipated for most models, possibly due to the essential pre-processing and dimensionality reduction of the data.

After the supervised models however, the results obtained with the unsupervised clustering techniques seemed a bit disappointing and showed the difficulty of clustering datasets with so many variables using these methods.

Although we tried to use several techniques to discover the best number of centroids to use and after experimenting different distances such as Euclidean, Manhattan and Gower the results did not get much better. It was also here where the computational power showed itself more critic. For example, we tried to add to this report a spectral clustering method but we were unable to run the code completely, despite having no apparent bug. The computation of the dissimilarity matrices used in the Gower distance proved really heavy too.

As for the classification of cluster labels it became a bit conditioned with the results obtained in the unsupervised models. It was however a big surprise when the results obtained with this solution, not only the accuracy, but all the other metrics excelled. It is undoubtedly a technique we want to explore further in the future.

In sum, this project proved challenging but at the same time enriching as a way to explore a lot of different methods techniques and applications. It was also really important for us to understand the importance and structure of all different phases of multivariate analysis and what is a project in this field all the way from beginning to end.

ID
ID of each client
LIMIT_BAL
Amount of given credit in NT dollars (includes individual and family/supplementary credit)
SEX
Gender (1=male, 2=female)
EDUCATION
1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown
MARRIAGE
Marital status (1 = married; 2 = single; 3 = divorce; 0=others)
AGE
Age in years
PAY_0
Repayment status in September 2005 (-2: No consumption; -1: Paid in full; 0: The use of revolving credit; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 9 = payment delay for nine months and above)
PAY_2
Repayment status in August 2005 (equal scale)
PAY_3
Repayment status in July 2005 (equal scale)
PAY_4
Repayment status in June 2005 (equal scale)
PAY_5
Repayment status in May 2005 (equal scale)
PAY_6
Repayment status in April 2005 (equal scale)
BILL_AMT1
Amount of bill statement in September 2005 (NT$)
BILL_AMT2
Amount of bill statement in August 2005 (NT$)
BILL_AMT3
Amount of bill statement in July 2005 (NT$)
BILL_AMT4
Amount of bill statement in June 2005 (NT$)
BILL_AMT5
Amount of bill statement in May 2005 (NT$)
BILL_AMT6
Amount of bill statement in April 2005 (NT$)
PAY_AMT1
Amount of prior payment in September 2005 (NT$)
PAY_AMT2
Amount of prior payment in August 2005 (NT$)
PAY_AMT3
Amount of prior payment in July 2005 (NT$)
PAY_AMT4
Amount of prior payment in June 2005 (NT$)
PAY_AMT5
Amount of prior payment in May 2005 (NT$)
PAY_AMT6
Amount of prior payment in April 2005 (NT$)
default payment next month (dpmn)
Default payment (1=yes, 0=no)

Table 6: Explanation of each feature of the dataset

	min	max	median	mean	sd
credit_limit	10000	1E+06	140000	167484.32	129747.66
gender	1	2	1	1.40	0.49
education	1	7	2	1.74	0.82
marital_status	1	4	2	1.56	0.53
age	21	79	34	35.49	9.22
payment_status_sept	1	11	3	3.03	1.17
payment_status_aug	1	11	2	2.39	1.00
payment_status_jul	1	11	2	2.25	1.01
payment_status_jun	1	11	2	2.21	0.96
payment_status_may	1	10	2	2.26	0.91
payment_status_apr	1	10	3	2.80	1.01
bill_sept	-165580	964511	22381.5	51223.33	73635.86
bill_aug	-69777	983931	21200	49179.08	71173.77
bill_jul	-157264	1664089	20088.5	47013.15	69349.39
bill_jun	-170000	891586	19052	43262.95	64332.86
bill_may	-81334	927171	18104.5	40311.40	60797.16
bill_apr	-339603	961664	17071	38871.76	59554.11
payment_amount_sept	0	873552	2100	5663.58	16563.28
payment_amount_aug	0	1684259	2009	5921.16	23040.87
payment_amount_jul	0	896040	1800	5225.68	17606.96
payment_amount_jun	0	621000	1500	4826.08	15666.16
payment_amount_may	0	426529	1500	4799.39	15278.31
payment_amount_apr	1	6939	156	1231.83	1859.33
default_payment	0	1	0	0.22	0.42

Table 7: Statistics of eache feature of the dataset

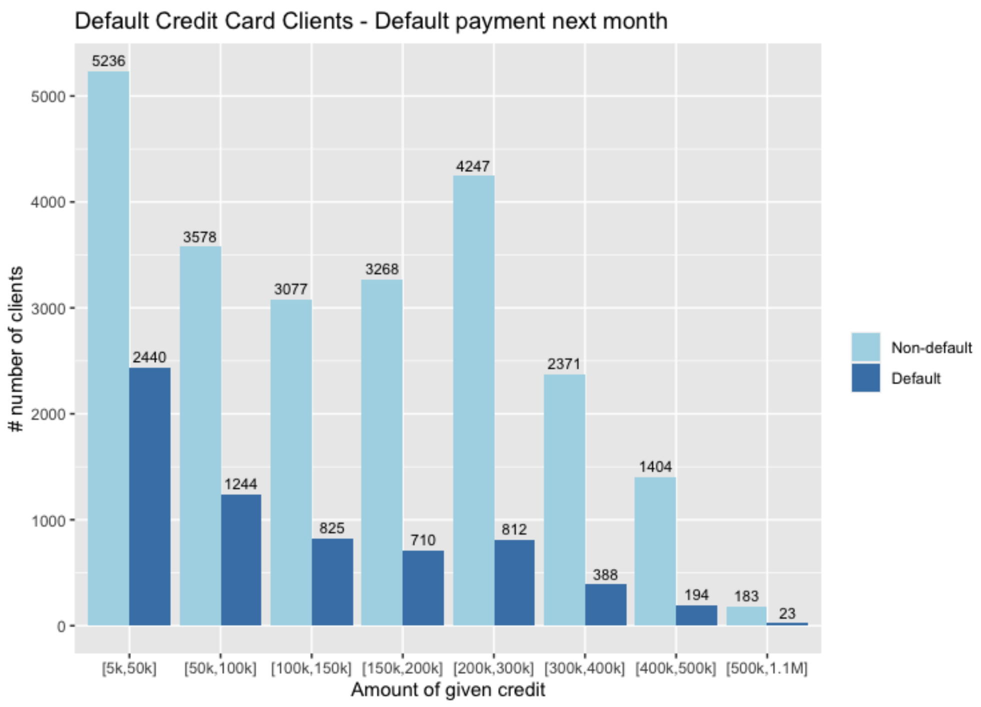


Figure 13: Default credit card clients by amount of credit given.

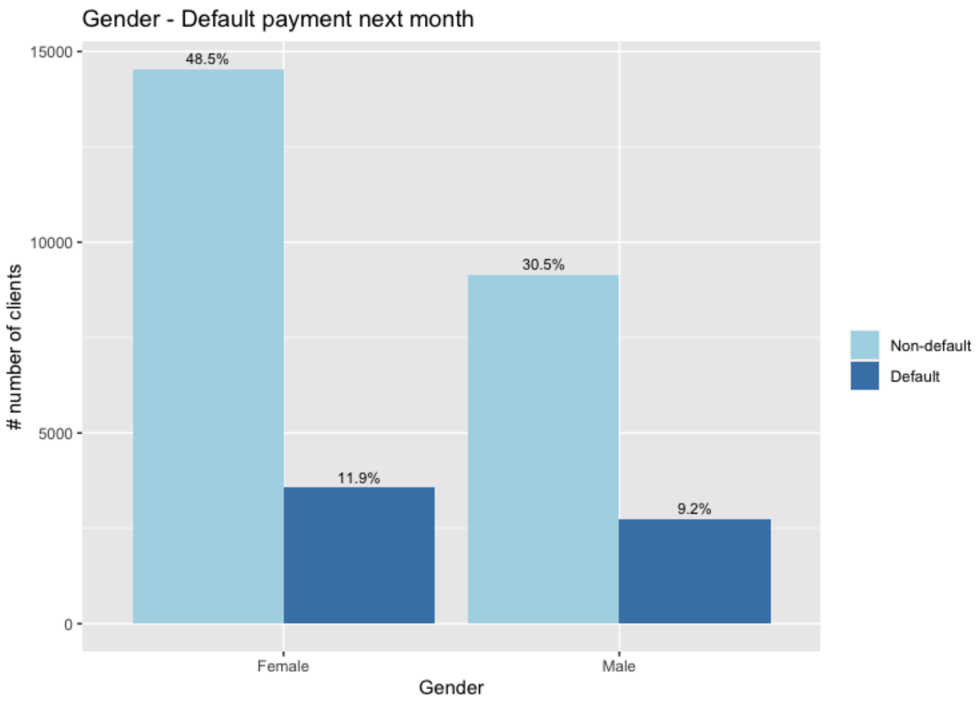


Figure 14: Default or not distribution by customer gender.

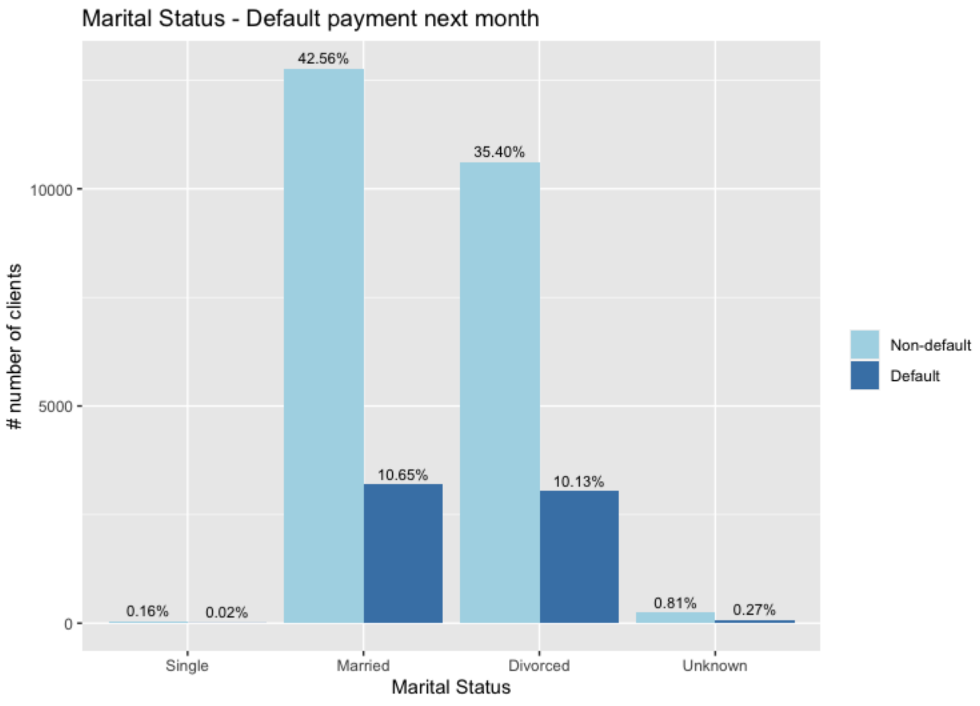


Figure 15: Default or not distribution by marital status.

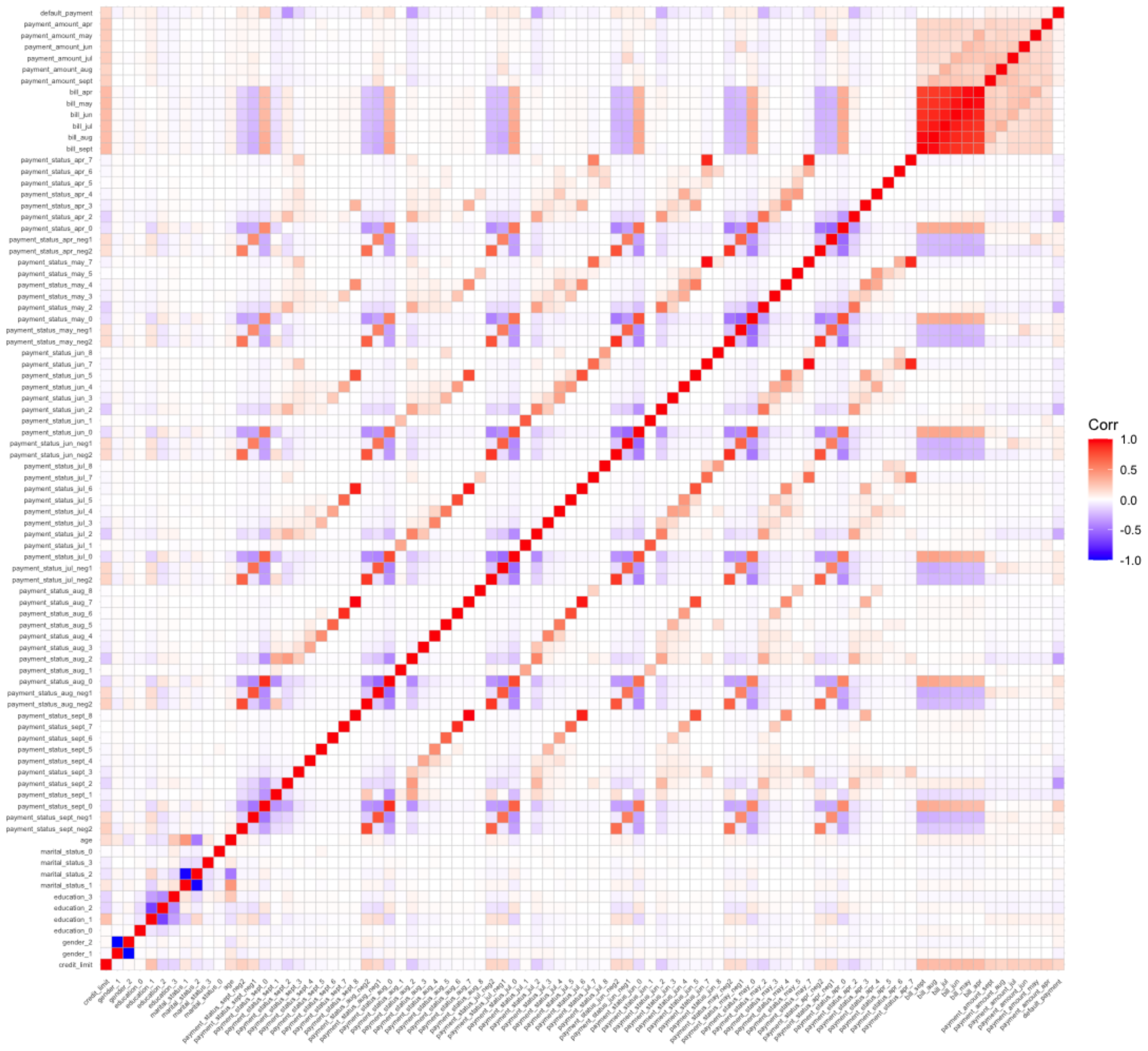


Figure 16: Correlation matrix.

	PCA - Normal				
Classifier	Accuracy	Precision	Sensitivity	Specificity	AUC
Naive Bayes	0.7971865	0.52666	0.41841	0.89896	0.659
SVM	0.766615	0.4102	0.2233	0.9137	0.569
LDA	0.8114754	0.58281	0.38651	0.92566	0.656
KNN	0.8025033	0.67563	0.44665	0.94238	0.677
DT	0.7617299	0.56227	0.39435	0.91751	0.656
RF	0.932855	0.56227	0.39435	0.91751	0.663

Table 8: Results for each supervised classifier with PCA features as input

	PCA - p > 0,4				
Classifier	Accuracy	Precision	Sensitivity	Specificity	AUC
Naive Bayes	0.8070447	0.55674	0.43619	0.90669	0.671
SVM	0.7938635	0.52573	0.27249	0.93395	0.603
LDA	0.8373948	0.67563	0.44665	0.94238	0.695
KNN	0.8443731	0.6764	0.5314	0.9317	0.732
DT	0.8067125	0.56227	0.39435	0.91751	0.656
RF	0.8495791	0.7028	0.5021	0.9429	0.723

Table 9: Results for each supervised classifier with PCA features as input where $p > 0{,}4$

	mRMR - Normal				
Classifier	Accuracy	Precision	Sensitivity	Specificity	AUC
Naive Bayes	0.7988(3)	0.5219	0.5372	0.8687	0.703
SVM	0.7815	0.47145	0.30696	0.90815	0.608
LDA	0.8(3)	0.66880	0.41377	0.94531	0.680
KNN	0.8185	0.74641	0.45174	0.95904	0.678
DT	0.77525	0.6069	0.4763	0.9177	0.697
RF	0.9235757	0.7869	0.4763	0.9656	0.668

Table 10: Results for each supervised classifier with mRMR as input

	mRMR - p > 0,4				
Classifier	Accuracy	Precision	Sensitivity	Specificity	AUC
Naive Bayes	0.8073333	0.5429	0.5411	0.8784	0.710
SVM	0.8071667	0.56713	0.35759	0.92715	0.642
LDA	0.8521667	0.74641	0.45174	0.95904	0.705
KNN	0.8626667	0.7523	0.5190	0.9544	0.737
DT	0.8625	0.7869	0.4763	0.9656	0.721
RF	0.8511667	0.6988	0.5158	0.9407	0.728

Table 11: Results for each supervised classifier with mRMR as input where p > 0,4

	Hierarchical Clustering - PCA					
Method	Accuracy	Balanced Accuracy	Precision	Sensitivity	Specificity	AUC
Single	0.804	0.49984	0	0	0.99969	0.5
Complete	0.804	0.49984	0	0	0.99969	0.5
Average	0.804	0.49984	0	0	0.99969	0.5
Ward	0.4275	0.39574	0.13154	0.34355	0.44793	0.604

Table 12: Results for each Hierarchical Clustering method with the PCAmixdata dataset

	Hierarchical Clustering - mRMR					
Method	Accuracy	Balanced Accuracy	Precision	Sensitivity	Specificity	AUC
Single	0.7238	0.49983	0	0	0.99965	0.5
Complete	0.7625	0.60197	0.70052	0.24366	0.96029	0.60197
Average	0.7805	0.62618	0.78535	0.28170	0.97065	0.626
Ward	0.7485	0.6245	0.5731	0.3478	0.9012	0.625

Table 13: Results for each Hierarchical Clustering method with the mRMR dataset

9.1 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.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{1}$$

$$Sensitivity = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2}$$

$$Specificity = \frac{TrueNegatives}{TrueNegatives + FalsePositives} \tag{3}$$

$$Accuracy = \frac{TruePositives + TrueNegatives}{TruePositives + TrueNegatives + FalsePositives + FalseNegatives} \tag{4}$$

References

[1] Marie Chavent, Vanessa Kuentz-Simonet, Amaury Labenne, and Jérôme Saracco. Multivariate analysis of mixed data: The r package pcamixdata. *arXiv preprint arXiv:1411.4911*, 2014.

[2] Nicolas De Jay, Simon Papillon-Cavanagh, Catharina Olsen, Nehme El-Hachem, Gianluca Bontempi, and Benjamin Haibe-Kains. mrmre: an r package for parallelized mrmr ensemble feature selection. *Bioinformatics*, 29 (18):2365–2368, 2013.

[3] Nicolas De Jay, Catharina Olsen Cavanagh, Gianluca Bontempi, Benjamin Haibe-Kains, and Maintainer Benjamin Haibe-Kains. Package 'mRMRe'. 2020.

[4] Ananda Mahto. splitstackshape: Stack and Reshape Datasets After Splitting Concatenated Values. `https://github.com/mrdwab/splitstackshape`, 2019. [Last access 14/04/2020].

[5] Chris Seiffert, Taghi M Khoshgoftaar, and Jason Van Hulse. Hybrid sampling for imbalanced data. *Integrated Computer-Aided Engineering*, 16(3):193–210, 2009.

[6] Silvia Cateni, Valentina Colla, and Marco Vannucci. A Method for Resampling Imbalanced Datasets in Binary Classification Tasks for Real-World Problems. *Neurocomputing*, 135:32–41, 2014.

[7] Nicola Lunardon, Giovanna Menardi, and Nicola Torelli. ROSE: A Package for Binary Imbalanced Learning. *R Journal*, 6(1), 2014.

[8] Fortran original by Leo Breiman, R port by Andy Liaw Adele Cutler, and Matthew Wiener. Breiman and cutler's random forests for classification and regression, 2018. URL `https://cran.r-project.org/web/packages/randomForest/randomForest.pdf`.

[9] Max Kuhn. The caret Package. 2019.

[10] Thomas Filaire. Clustering on Mixed Type Data. `https://towardsdatascience.com/clustering-on-mixed-type-data-8bbd0a2569c3`, 2018. [Last access 28/05/2020].