APRENDIZAGEM COMPUTACIONAL

Diogo Filipe (201806280)
Ricardo Nunes (201706860)
José Macedo (201705226)

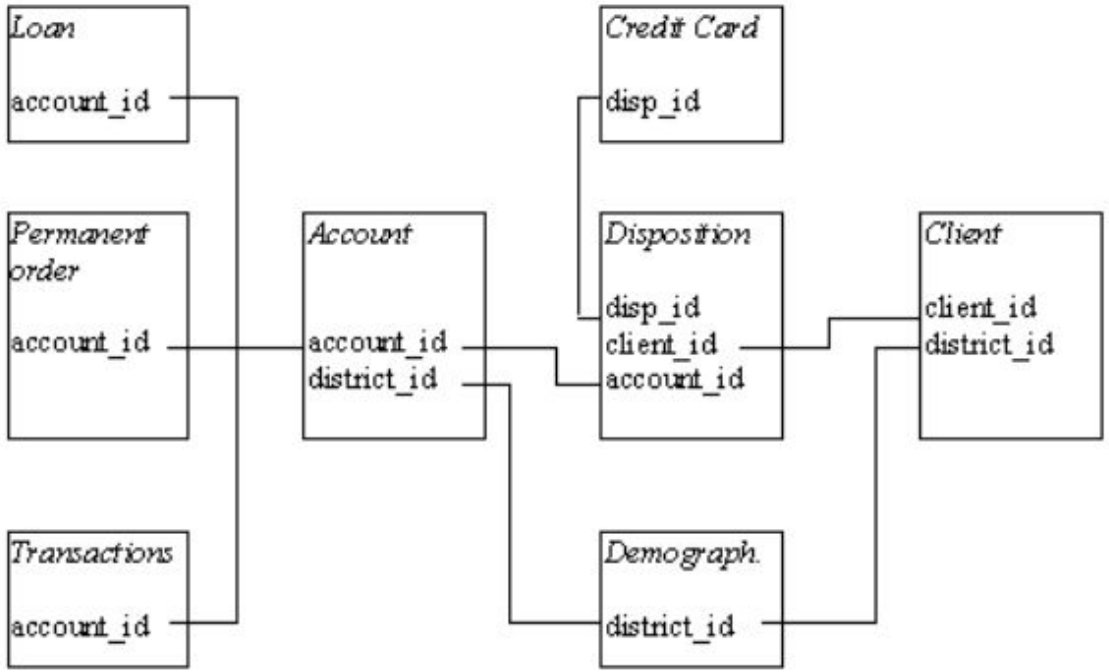
Domain

To loan money to someone, we must trust that person.

That confidence is gained by getting to know the person, knowing the person's data and analyzing it, we decide whether the person is trustworthy or not.

This project aims to do just that, with a dataset of a Czech bank, analyze and predict whether a person will pay the loan to the bank or not.

Thus, the bank improves its services.



Business Understanding

Analysis of requirements

- The project needs to be done before the deadline (13 of december), along with the presentation and at least two Kaggle submissions

- R was used for data visualization and validation. To develop the predictive models, we used Python. For the descriptive analysis, we used RapidMiner

Business goals

- Our goal was to analyze the dataset and predict whether a person will pay the loan to the bank or default.

- We will consider that the business goal has been achieved if our model correctly predict at least 90% of the cases.

Data Mining goals

- The predictions should be as correct as possible, giving an high area under the receiver operating characteristic curve (AUC). We aimed for a AUC greater than 0.9.

Exploratory Data Analysis

- **The data is imbalanced**: around 86% of the loans have class 1, and 14% have class -1 (loan default)
- Loan amount has a high standard variation.
- **Loan amount doesn't influence the payment of the loan. People default on low loan amount as well in the high ones.**
- Loan amount and duration are correlated (pearson coefficient: 0.593)
- **Duration is actually a categorical variable**, with values 12, 24, 36, 48 and 60 weeks.
- Most common type of credit card is *classic* (73%), followed *Junior* (22%) and *Gold* (5%).
- **There is contradictory data**: some clients belong to more then one district.
- There is no correlation between loan amount, average salary, n^a of entrepreneurs and unemployment rate and the status of the loan.
- There are some outliers in the data. However, we didn't remove them, since our data is imbalanced and the outliers can be the differentiator factor.

Exploratory Data Analysis

- The amount of transactions of an account does not influence the status of the loan.
- **All missing values were from the transactions data on the columns account**: *bank*, *k_symbol* and *operation*. However, **the missing values were not accidental**: they were purposely left blank.
- **Each account had only one loan and, at most, one credit card**. An account can be connected to at most 2 clients: one owner and one disponent.
- **There are bank clients of age less than 18 years that asked for a loan**. Since there was no rule in the specification against it, we decided to keep the data.
- There are two missing values on table district in columns *unemployment rate '95* and *no. of committed crimes '95*. We imputed those values with the mean value of those columns, since there were only two.
- There are 3 transactions with amount less than or equal to 0, since that is invalid data, we ignore them.
- There are 597 transactions with non-positive balance.

Exploratory Data Analysis

Data quality dimensions:

- **Accuracy:** The data is in its majority correct, with a few exceptions discussed previously
- **Completeness:** There are a lot of missing values, but they are located mostly on the transactions. The other tables are completed, with the exception of districts, that have two missing values.
- **Consistency:** There are some clients with more than one district. The reason for this is that the client and account tables both have the district ID, and sometimes they mismatch. We assumed the source of truth to be the account district ID, and dropped the client one.
- **Timeliness:** All data of loans are in the 90's decade.
- **Believability**: The data was given by the professor, and its' source isn't specified, so we assume that it's trustworthy.
- **Interpretability:** The data is very well divided, according to the UML, so interpretability wasn't an issue

Exploratory Data Analysis

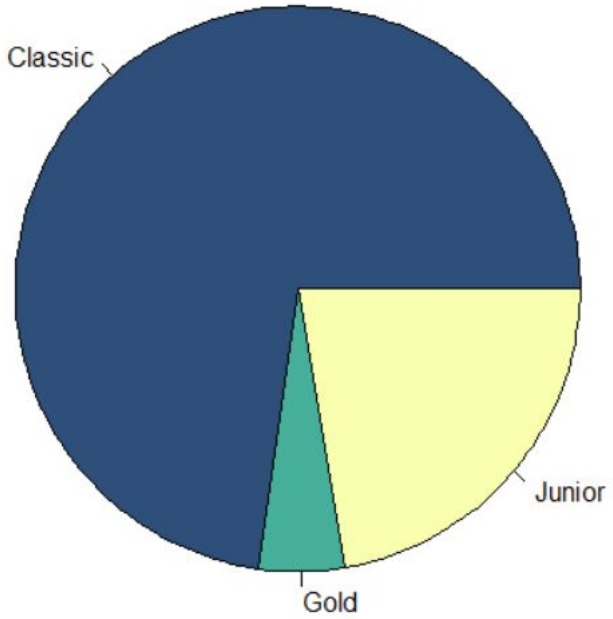


Figure 1: Pie Chart of credit card types

As we can see, transaction with type "classic" is the most frequency value.

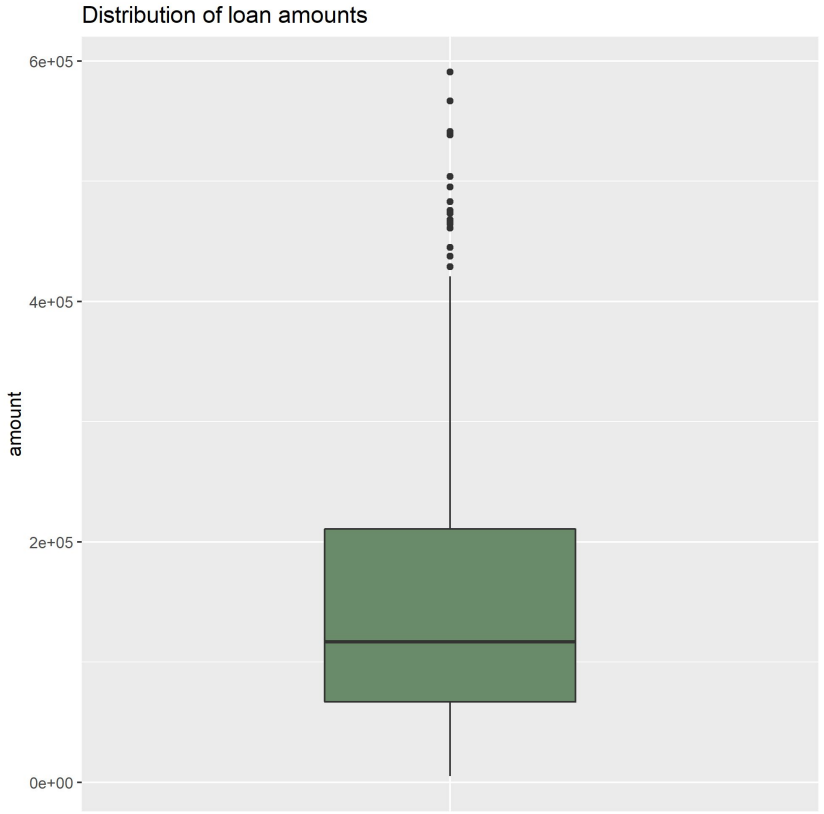


Figure 2: Boxplot of Distribution of Loan Amounts

The average of the average sale by district is concentrated between the values 8500 and 9500. We can also see outliers in the boxplot which represent the district that have the bigger values regarding the average sale.

Exploratory Data Analysis

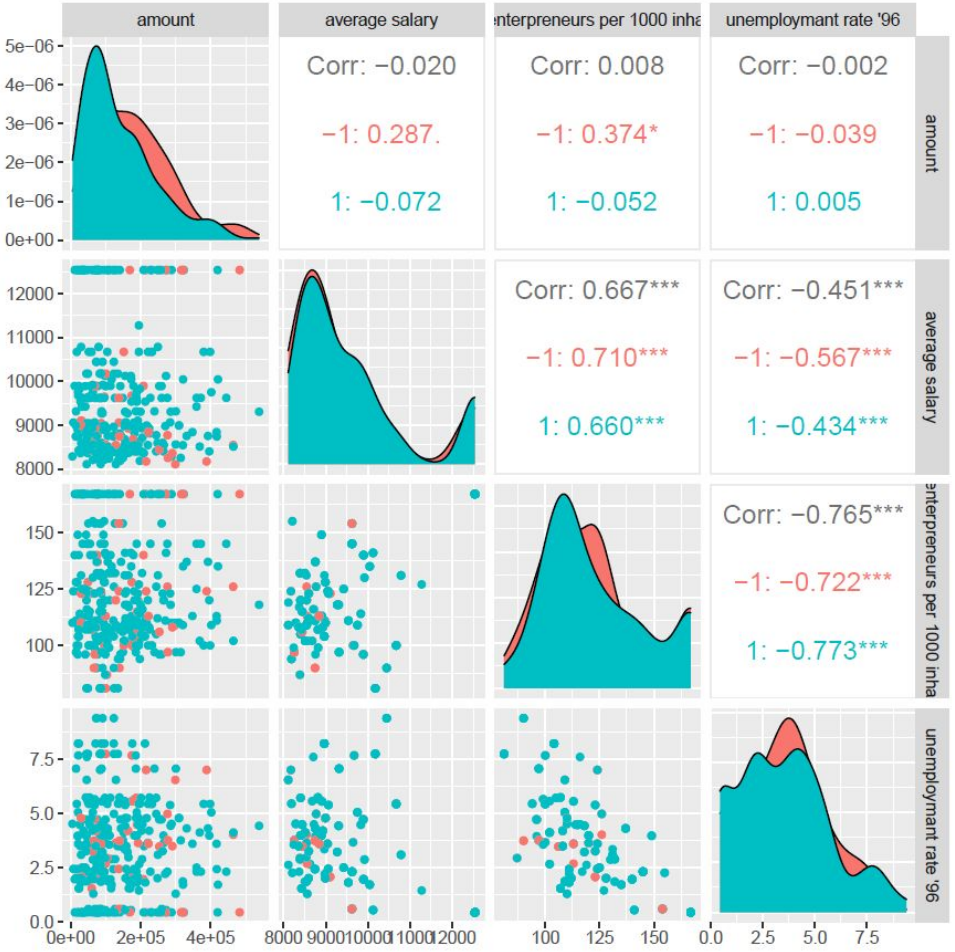


Figure 3: Multi-scatterplot of loan amount, average salary, nr of entrepreneurs and unemployment rate

We wanted to see if the status of a loan is influenced by the amount of a loan, the average salary in a district, number of entrepreneurs per 1000 inhabitants and unemployment rate '96 there.

As we can see, from these 4 variables, the average salary and the number of entrepreneurs per 1000 inhabitants have a correlation of 0.667 between each other, being this the most relevant one, while the others are too low.
From the analysis of the graphs, it looks like the average salary is the one that is more likely to influence the status of loan, even though the correlation is basically insignificant.

Exploratory Data Analysis

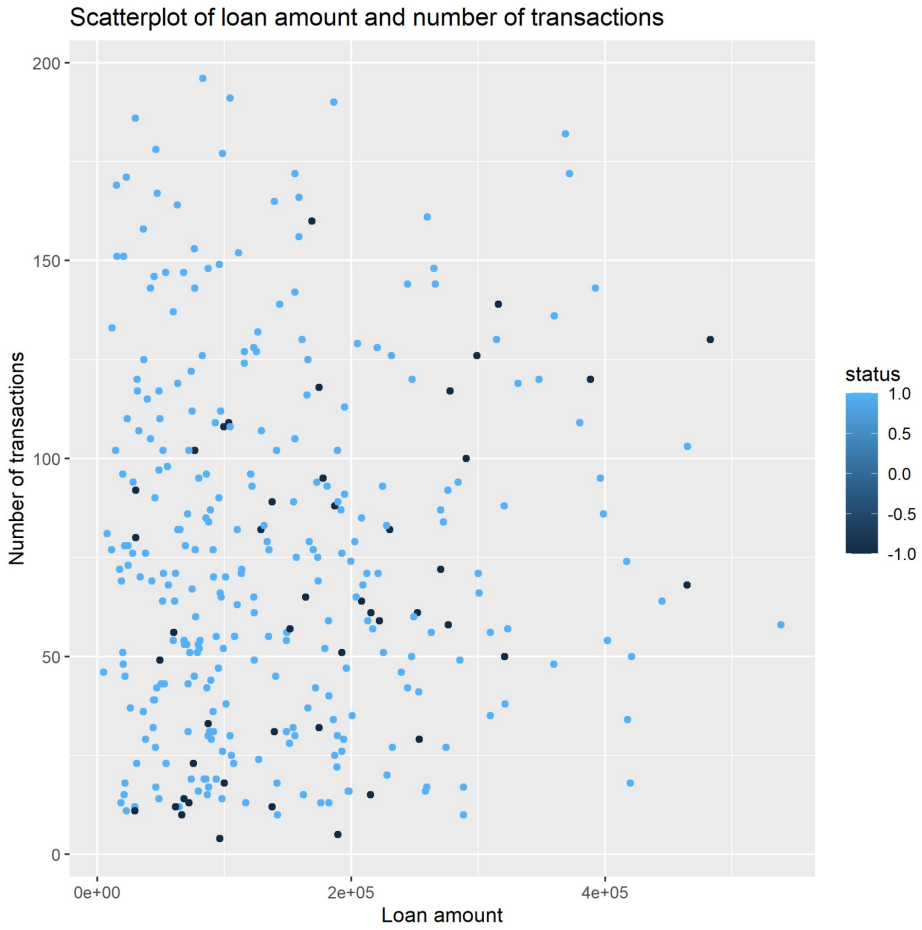


Figure 4: Scatterplot of Loan Amount and Number of Transactions

We wanted to see if the number of transactions and the loan amount influenced the status of the loan. For example, our hypothesis if a person has a high loan and does a lot of transactions, it may not be able to pay that loan.

Turns out that our how hypothesis was wrong: there is no correlation between this variables, has you can see by the plot above.

Exploratory Data Analysis

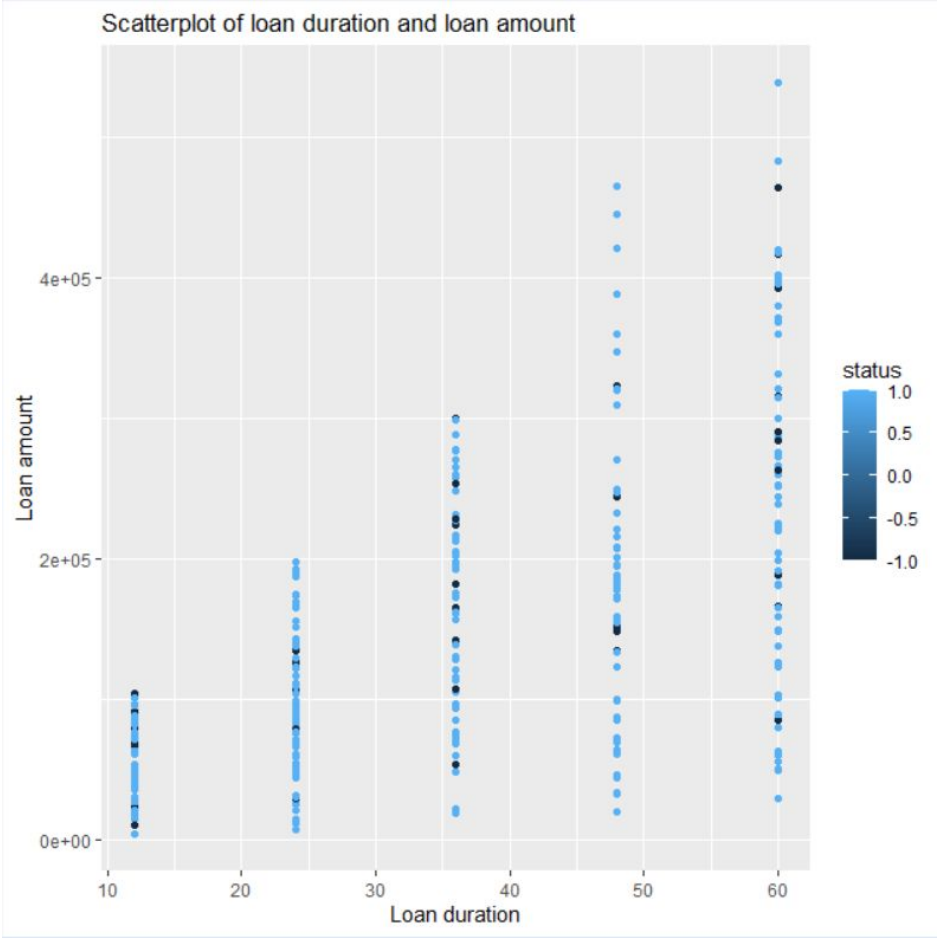
Figure 5: Scatterplot of Loan Duration and Loan Amount

We also wanted to find out if the loan duration and the loan amount influenced the status of the loan.For example, our hypothesis if a person has a high loan and has a small duration to pay, it may not be able to pay that loan.

In this graphic we can see some correlation between the variables loan duration and loan amount, the duration seems to be longer for a higher amount,as expected.

Nevertheless, it has nothing to do with the status.

As we can see, the loan duration takes only 5 different values (12,24,36,48 and 60), so we will treat it as a categorical variable.

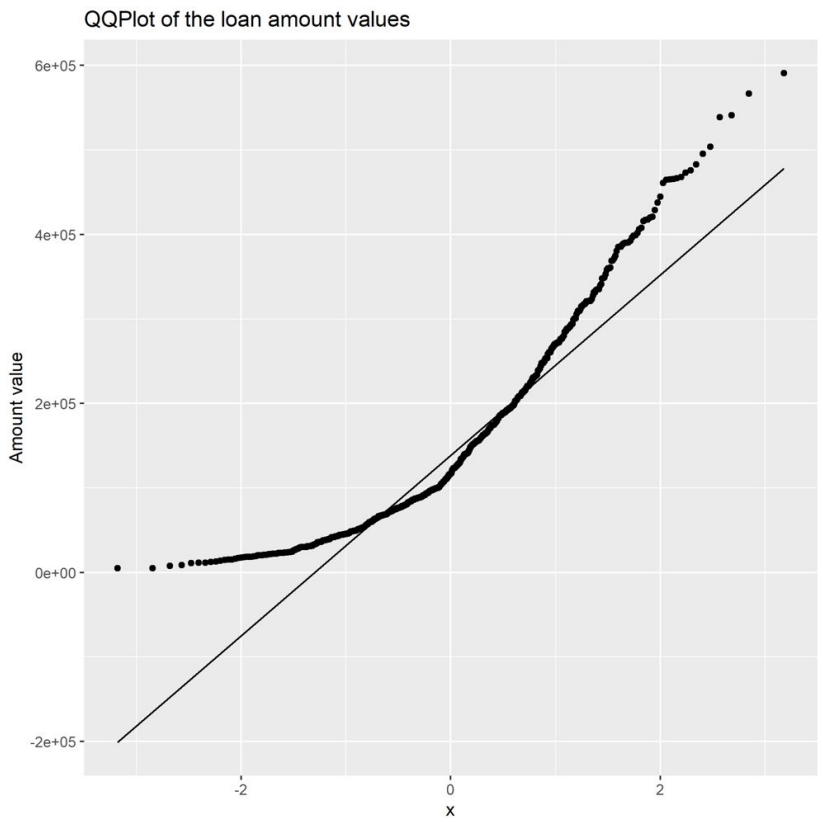


Exploratory Data Analysis

Figure 5: QQPlot of the Loan Amount Values

We tried to verify if the amount followed a normal distribution using a QQplot.

Turns how that the distribution is right skewed. This means most amount values are below average.



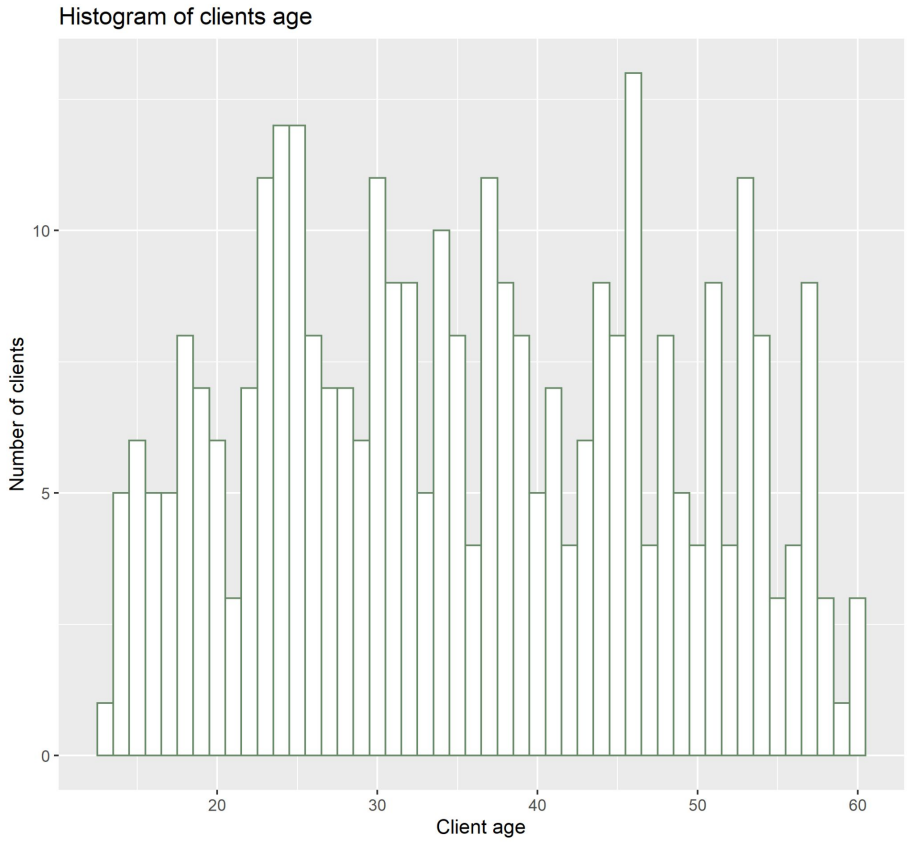


Figure 6: Histogram of Clients Age

Exploratory Data Analysis

Average	Standard Variation	IQR
151410	113372	143922

Table 1: Simple statistics for loan data

The following table presents some common statistics of the column amount. We selected the average, standard deviation and IQR so that we could see how disperse is the data.

The amount values are very disperse, with a standard variation close the mean value.

Average	Standard Variation	IQR
15 678.0	9 190.0	6 372.0

Table 2: Simple statistics for transaction data

We calculated the same attributes of the Amount column, since it can give us an idea of how much money one has in its account in average, information that could be directly proportional to who pays the loans and who doesn't.

Data Preparation

- Categorical variables:
  - Loan duration needs no preparation: value is expressed in weeks (absolute scale).
  - **One-Hot encoding was applied** to the credit card type and the account frequency type, since there were few possible values (three in each case).
  - Since not all accounts have credit cards, the **missing values were imputed with 0**.
- All dates were parsed to the correct object type.
- Redundant/Useless features:
  - **All ID's were removed,** since they do not contain relevant information (they only serve to connect the data).
  - After extracting all information from the dates, they were removed as well.

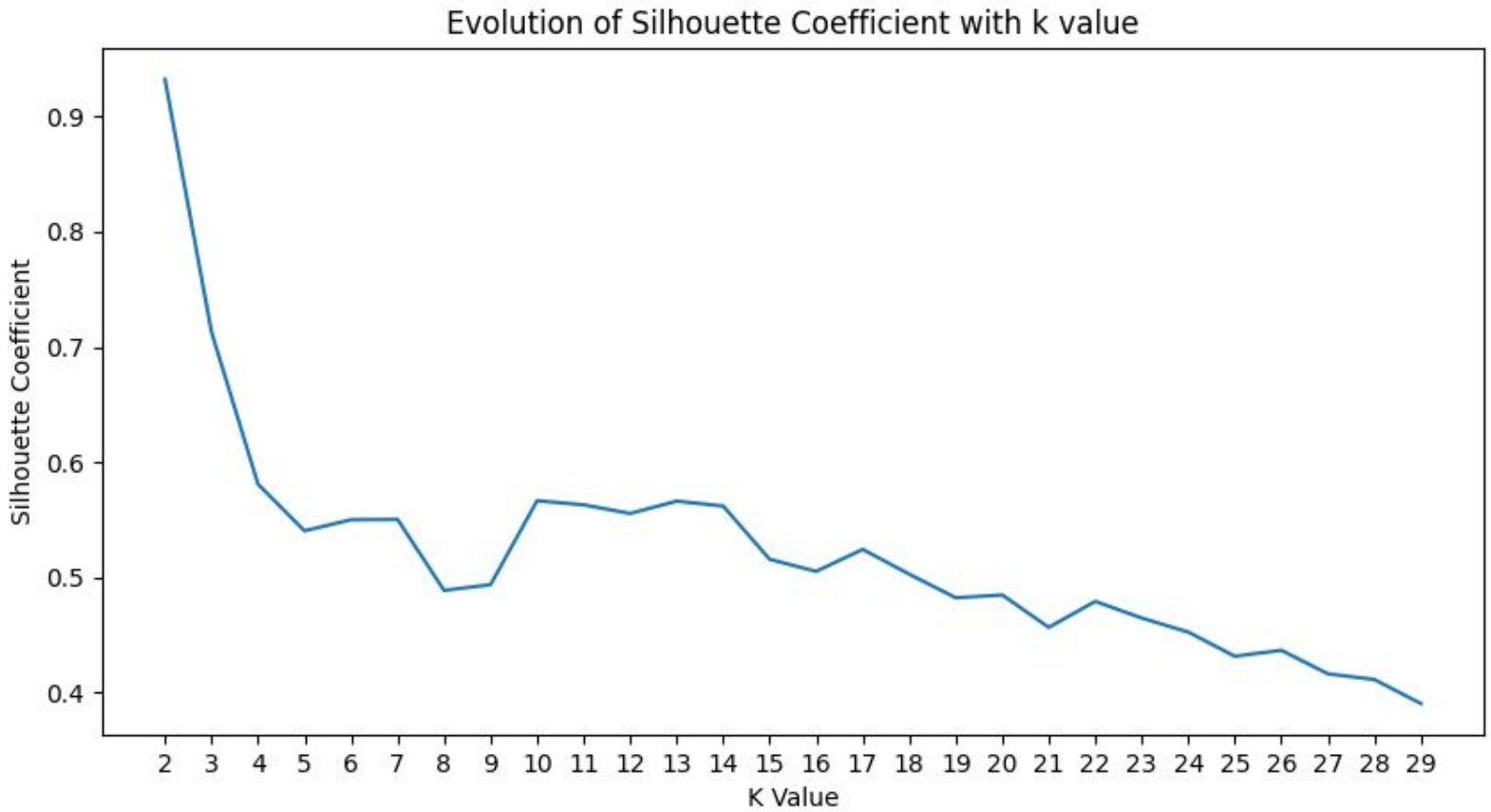
Data Preparation

- **Normalized all numeric features using zscore.**
- Missing table permanent order, specified on the UML. Ignored this missing table.
- **To treat imbalanced data, the SMOTE technique was applied to training data (leaving the testing data intact).** Since we have categorical data, a variation of the algorithm (SMOTENC) that handles this type of data was picked.
- Corrected miss labeled data (“withdrawal in cash” transformed to “withdrawal” in transaction).
- Transformed transaction amount to a negative number when its type was “withdrawal”.

Data Preparation

- Feature Engineering:
  - Removed payment column, since it derived from duration and amount (amount = duration * payments).
  - **Extracted sex and age from birth date.**
  - **Replaced all district information with a cluster** (K Means Cluster).
  - **Grouped all transaction table by account, and extracted statistical features**: nr of each type of operations and percentage, nr of each type of k_symbol and percentages, nr of transactions types (withdrawal/credit), mean and standard deviation of amount and balance.
  - Group owner and disponents of account by creating a new column called NumberOfPeople.

Data Preparation

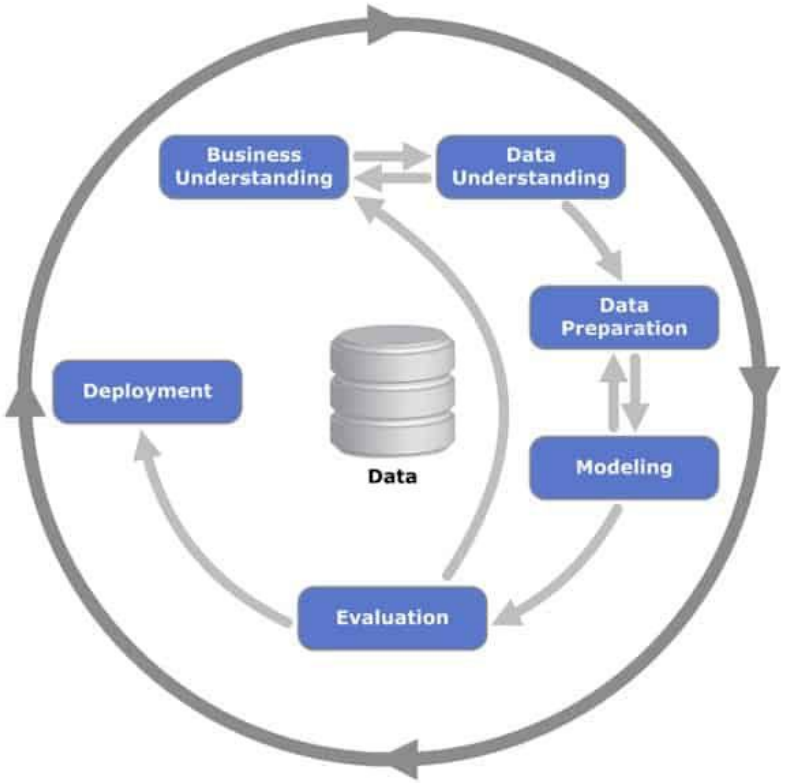


Problem Definition

Our problem is to determine if a loan will be paid or not, based on the given dataset.
In order to do that, we explored, analise and prepared the data, using models in python to make predictions.

The methodology we followed was *Cross Industry Standard Process for Data Mining*.

The evaluation was made with several fitted models.



Project Management

CRISP-DM

In order to approach the predictive task, we opted to use *Cross Industry Standard Process for Data Mining* methodology, as we said previously.
This way, in the earliest stage of our project we took our time to assure that the Business Understanding was well defined so that we could start the Data Understanding phase, and posteriorly, the Data Preparation phase. We programmed in R, Python and RapidMiner in order to do that and analysed the .csv files given, focusing on the ones we felt were more relevant like *loan.csv*, *trans.csv* or even *district.csv*.
With the continuous enhancement of that part, we proceeded with the creation of a pipeline and election and implementation of models in order to test the classifiers e verify if the results were the ones we intended them to be.

Project Management

Used Tools:

- **GitHub**: It was essential to manage the code and made possible uploads by all members of the group without compromising each others' work by using diverse branches. Github also has a tool named **Github Projects,** which allowed us to list all the tasks to be done, assign them to the members of the group and understand the state of our project (if it was advanced or behindhand).
- **HackMD**: It was used so that all members of the group could write the various points of the final report in markdown. This way, at the end, we had all the necessary information in one place and available to use.
- **Whatsapp and Discord**: Even though we, as a group, met in person to discuss and organize the progress of our project sometimes, we mostly reunited via **Discord** and, to share some quick ideas, we messaged each other via **Whatsapp**.

Descriptive Task

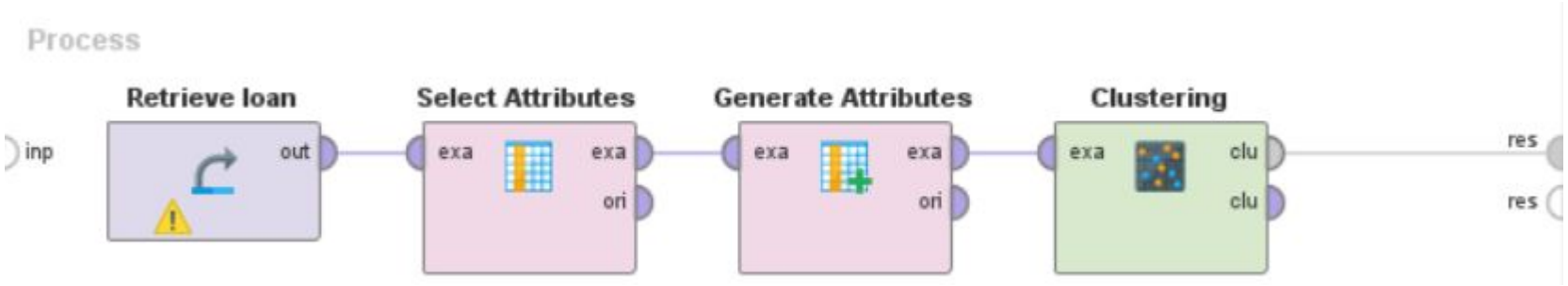
Goal

Group the dataset to find some behavior patterns that help us understand the groups that pays or not the loan.

Algorithms used

k-means

DBScan



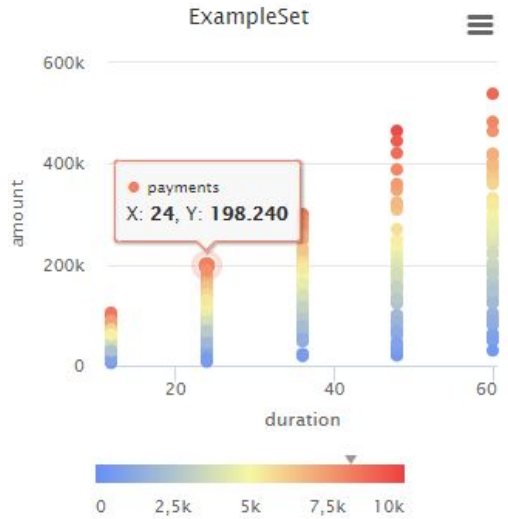
Descriptive Task

Several clustering operations were performed with different attributes.

As expected, the clusters are similar to the conclusions we make from the data analysis.

This cluster shows that clients that make bigger loans, they have a bigger salary and the balance of the transactions is larger. And the bigger the loan amount, the bigger duration and number of payments, as we have seen before.

Attribute	cluster_0	cluster_1
amount	210432	119534
duration	56	26
payments	3757	4597
average_salary	7698	7430
average_balance	53289,713	28674,123
average_amount	10643,129	4780,912



Experimental Setup

Explored Models

- K Nearest Neighbour
- Multi-Layer Perceptron
- Random Forest Classifier
- Support Vector Machine
- Gaussian Naive Bayes

- We used Cross Validation with a n=10 (90% training, 10% testing) to reduce the overfitting of the data.
- The metric used to evaluate the results was AUC .
- We used random search to fine tune the parameters of our models. The random search normally outperformes the grid search, since the grid search brute-forces all parameters, hence our choice.
- In order to implement the algorithms, we resorted to the *Python* library *Scikit-learn*.

Experimental Setup

K Nearest Neighbour:

KNN estimates the probability of a data point to be a member of one group or the other depending on which one the data points that are nearest to it are in.

Classifier:

Initially, we used the classifier ***KNeighborsClassifier*** and as a parameter ***n_neighbors=30***.

Parameters chosen for hypertuning and value range:

- *n_neighbors* = ***(5,10, 15, 20)***
- *leaf_size* = ***(10,20,30, 40, 50)***
- *p* = ***(1,2)***
- *weights* = ***('uniform', 'distance')***
- *metric* = ***('minkowski', 'chebyshev')***

Experimental Setup

Random Forest Classifier:

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Parameters chosen for hypertuning and value range (Random Forest Classifier):

- *n_estimators = [10, 100,200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]*
- *criterion = ['gini', 'entropy']*
- *max_depth = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None]*
- *min_samples_split = [2, 5, 10]*
- *min_samples_leaf = [1, 2, 4]*
- *max_features = ['auto', 'sqrt', 'log2']*
- *bootstrap=[True,False]*
- *class_weight = ['balanced', 'balanced_subsample']*

Experimental Setup

Neural Networks (Multi-Layer Perceptron):

This model optimizes the log-loss function using LBFGS or stochastic gradient descent.

Classifier:

Initially, we used the classifier ***MLPClassifier*** and as parameters:

- *alpha=1e-5*
- *hidden_layer_sizes=(5,)*
- *solver='lbfgs'*
- *random_state=10*

Experimental Setup

Parameters chosen for hypertuning and value range for NN:

- *hidden_layer_sizes=[(100,), (100, 50), (100, 50, 20), (100, 50, 20, 10), (100, 50, 20, 10, 5)]*
- *activation=['logistic', 'tanh', 'relu']*
- *solver=['lbfgs', 'adam']*
- *alpha=stats.uniform(loc=0.0001, scale=0.001)*
- *learning_rate_init=stats.uniform(loc=0, scale=0.01)*

Experimental Setup

Support Vector Machine:

Supervised learning models with associated learning algorithms that analyze data or classification and regression analysis. Since it is based on statistical learning frameworks or VC theory, Support Vector Machines are very robust.

Classifier:

Initially, we used the classifier ***SVC*** with no parameters.

Parameters chosen for hypertuning and value range:

- ***C=stats.uniform(loc=1, scale=6)***
- ***kernel=['rbf', 'sigmoid']***
- ***gamma=['scale', 'auto']***
- ***coef0=stats.uniform(loc=0, scale=6)***

Experimental Setup

Gaussian Naive Bayes:

The naive Bayes classifiers are a family of simple “probabilistic classifiers” based on applying the Bayes’ Theorem with strong independence assumptions between the features. They are highly scalable.
In the Gaussian Naive Bayes classifier specifically the continuous values associated with each class are distributed according to a normal distribution.

Classifier:

We used the classifier ***GaussianNB*** with no parameters. The classifier didn’t have any parameters that could be fine-tuned (it had none), so regarding hypertuning and value range we couldn’t approach the classifier the same way we approached the others.

Experimental Setup

To run our project, any user can run this two scripts:

explore_models.py
In this file we explore the models using RandomSearch and find the best parameters to tune.

main.py
The main file loads the csv data, does the preprocessing and trains the model with it.

The project also contains a configuration file called **config.ini** that has the best parameters to use in the algorithms (drawn from the fine tuning process).

Database Tools

- All data was provided to us in CSV format. Since this data format can be easily imported and manipulated and there isn't a lot of data, we felt no need to convert to other formats. Moreover, the library *Pandas* has built-in functions for importing and exporting data in CSV

Results

We obtained better results with the Random Forest Classifier, followed by Gaussian Naive Bayes and MLP.

Gaussian Naive Bayes, MLP and Support Vector had lower but reasonable results, even though Gaussian Naive Bayes is a simple classifier and didn't have parameters that could be tuned as we previously mentioned.

In the Kaggle Competition, the Decision Tree results we got was obtained in a very initial phase of our project. Later, with the application of SMOTE and fine tuning, we got better results, mainly using the Random Forest Classifier.

	Scores
Random Forest	0.93903
K Nearest Neighbour	0.64191
Gaussian Naive Bayes	0.81294
MLP	0.80924
Support Vector	0.78705

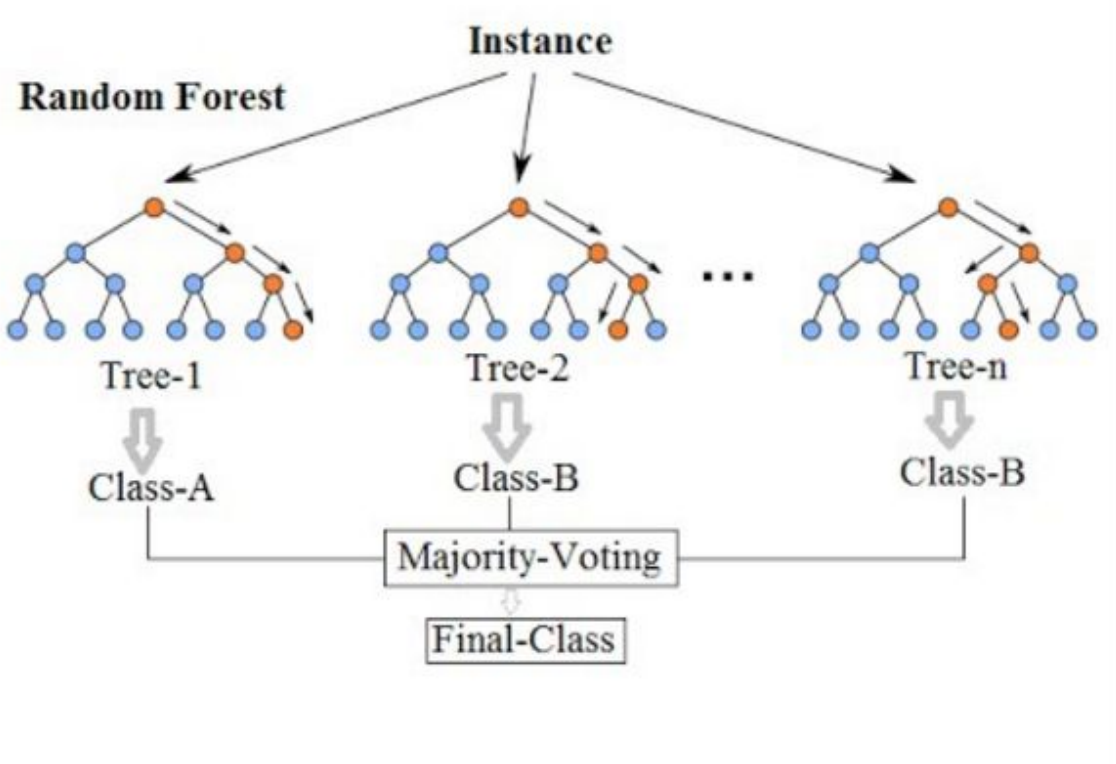
	Private Score	Public Score
Random Forest	0.92098	0.94691
Decision Tree	0.54444	0.63024

Results

RFC final tunned parameters:

- *random_state*=*11*
- *n_estimators* = *1600*
- *min_samples_split* = *2*
- *min_samples_leaf* = *2*
- *max_features* = *'sqrt'*
- *max_depth* = *100*
- *criterion* = *'entropy'*
- *class_weight* = *'balanced_subsample'*
- *bootstrap* = *False*

Best Score: 0.93903

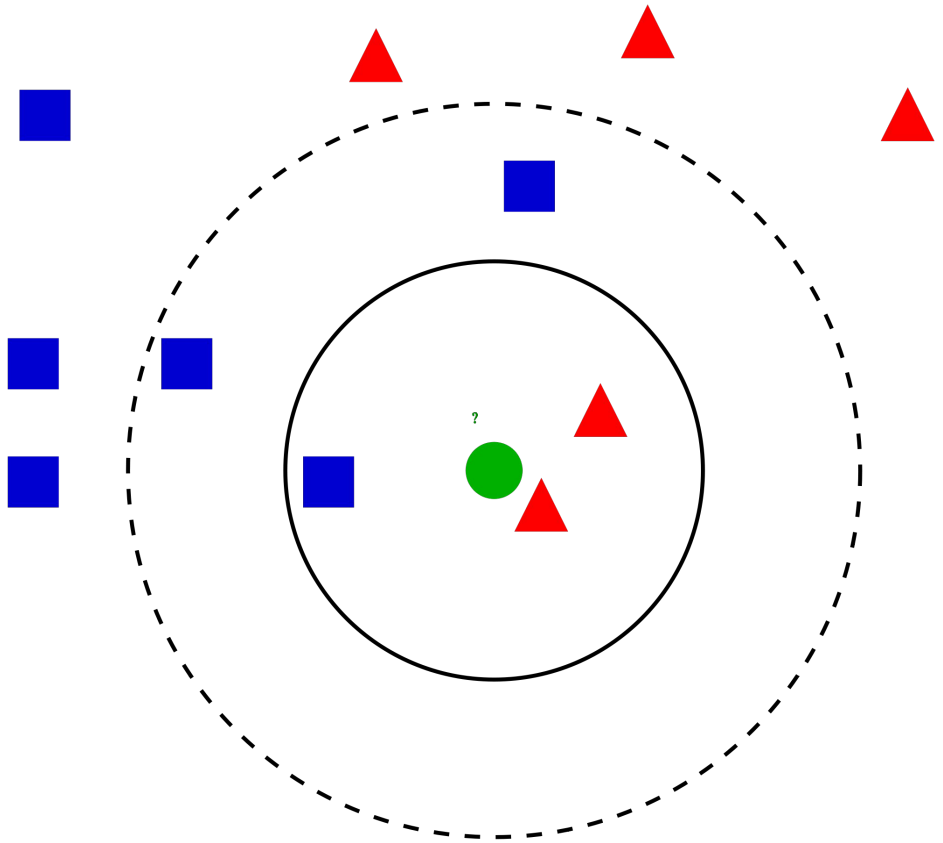


Results

KNN final tuned parameters:

- *'weights'*: ***'distance'***
- *'p'*: ***1***
- *'n_neighbors'*: ***5***
- *'metric'*: ***'minkowski'***
- *'leaf_size'*: ***50***

Best score: 0.6419119458128077

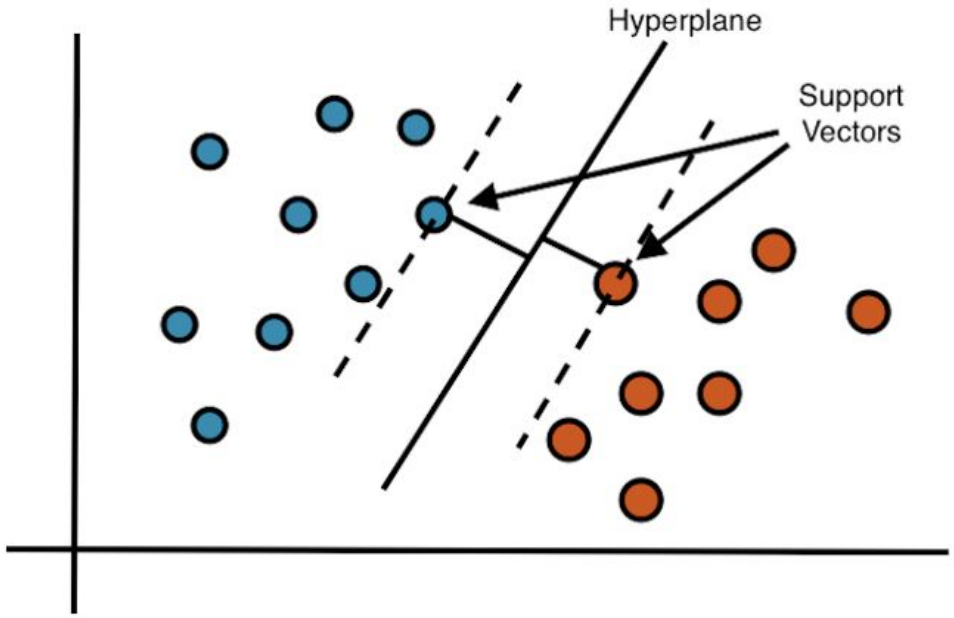


Results

SVM final tuned parameters:

- *'c': 2.1883771885577437*
- *'coef0': 4.563184273193753*
- *'gamma': 'scale'*
- *'kernel': 'rbf'*

Best score: 0.787050492610837 4

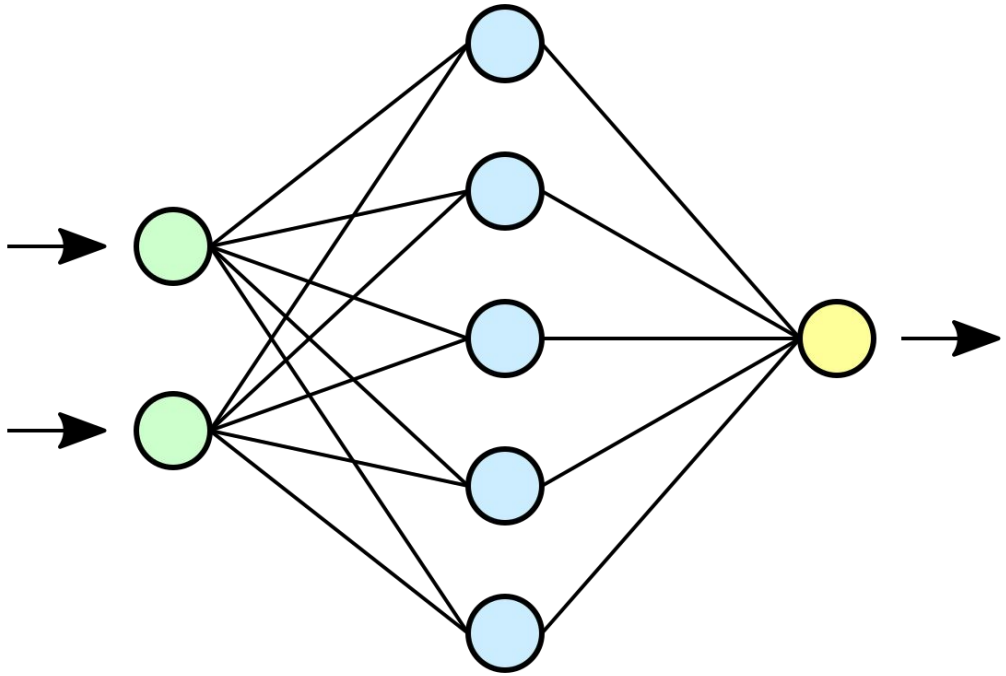


Results

NN final tuned parameters:

- *'activation': 'relu'*
- *'alpha': 0.0009908165306793224*
- *'hidden_layer_sizes': (100, 50, 20)*
- *'learning_rate_init': 0.006125260668293882*
- *'solver': 'lbfgs'*

Best score: 0.8295320197044337



Conclusions, Limitations and Future Work

Based on the obtained results, the Random Forest algorithm is the best to implement since it allowed us to get our best score. Considering our performance in Kaggle with a private score of 0.92098, we believe we had a good performance. This way, we feel that our work was at some point successful, even though if we were to do this project again we believe we would come up with better results since we understand way more the methodologies and process now than in the beginning.

In our development, the following of the methodology CRISP-DM was proved beneficial since it allowed a better organization and ensured the progress was done in a way so that we could understand the problem, the goals, the data and the modeling process. It was essential the addition of new features because they increased the performance a lot, specially the fine-tuning of the parameters chosen for hypertuning and value range.

Unfortunately, we consider that the dataset is too small and lead to a high chance of overfitting. Nevertheless, we understand that when working with enterprises and companies in the real world we could be faced with a even more incomplete dataset so it was enriching working with one that was far from perfect in order to prepare us in some way.