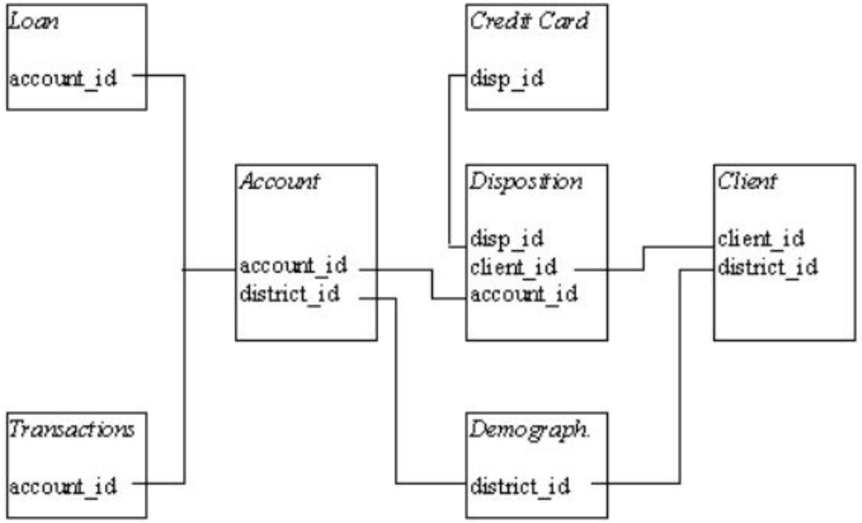
To loan or not to loan - that is the question

AC - 2022

Group 14
Fábio Huang - up201806829
Henrique Nunes - up201906852
Luís Guimarães - up202204188

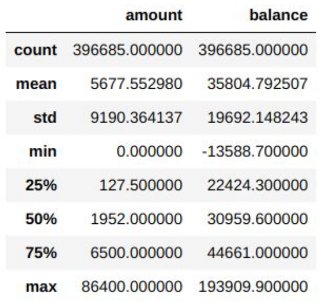
Domain description

- 4.500 **accounts** (3 features)
- 202 **cards** (3 features)
  - 177 for **development**
  - 25 for **competition**
- 5.369 **clients** (2 features)
- 5.369 **dispositions** (3 features)
- 77 **districts** (15 features)
- 426.885 **transactions** (9 features)
  - 396.685 for **development**
  - 30.200 for **competition**
- 682 **loans** (5 features + 1 label)
  - 328 for **development** (labeled)
  - 354 for **competition** (not labeled)

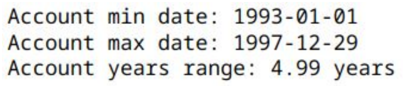


Exploratory Data Analysis

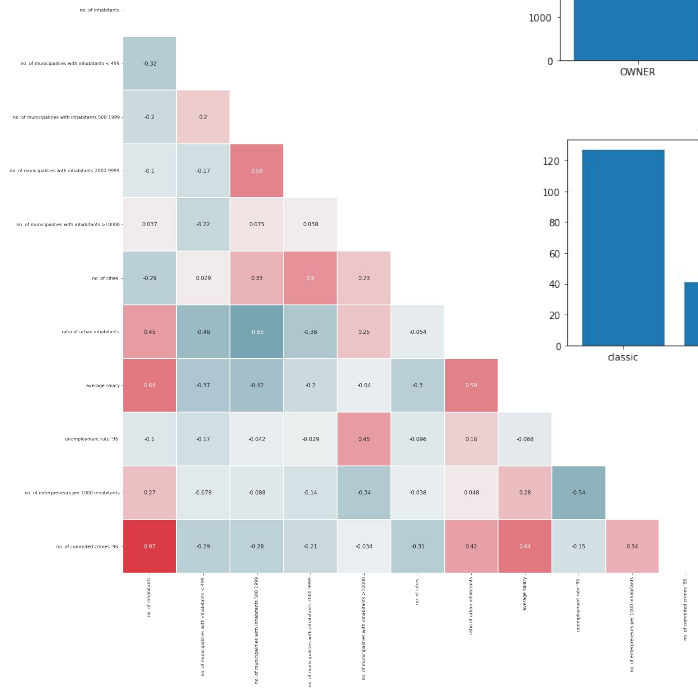
- For each feature, depending on its type, basic metrics were calculated to better understand the data:
  - frequency
  - mean
  - median
  - minimum
  - maximum
  - standard deviation
  - interquartile range
  - min max range
- For a multivariable analysis, it were calculated correlation matrices with the pearson method

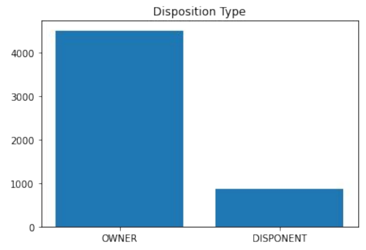


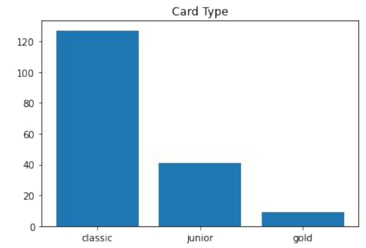
	amount	balance
count	396685.000000	396685.000000
mean	5677.552980	35804.792507
std	9190.364137	19692.148243
min	0.000000	-13588.700000
25%	127.500000	22424.300000
50%	1952.000000	30959.600000
75%	6500.000000	44661.000000
max	86400.000000	193909.900000



```
Account min date: 1993-01-01
Account max date: 1997-12-29
Account years range: 4.99 years
```

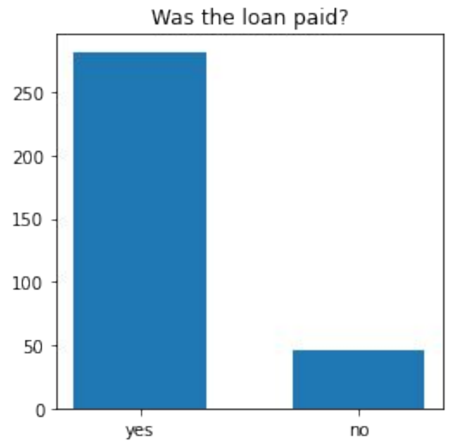


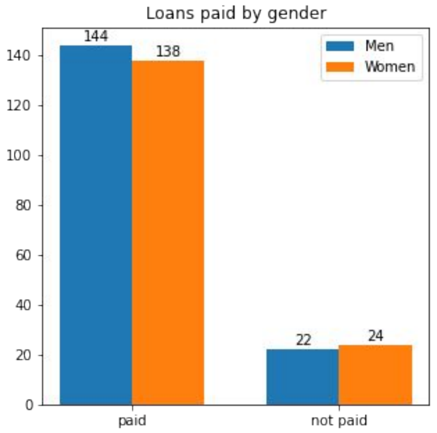


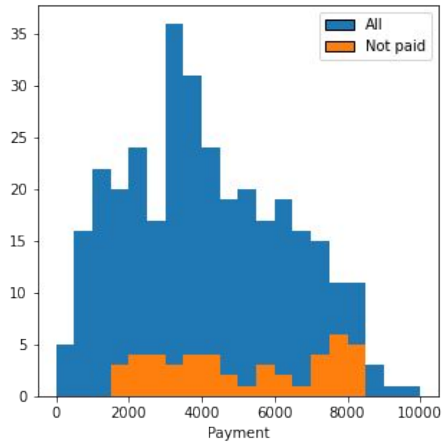


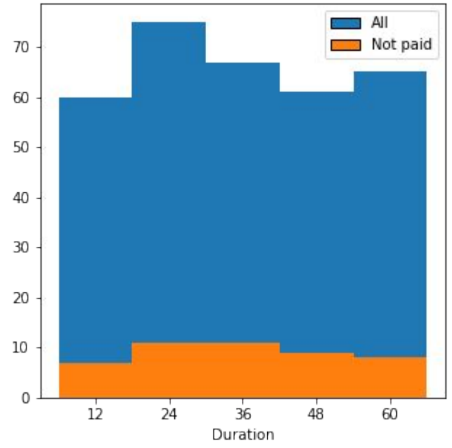
Exploratory data analysis

- The loan status class is unbalanced: only 14% of the loans were paid.
- The genders give no information on whether the loan was paid or not
- The ratio between the paid and the non paid loans is higher on higher monthly payments
- The loan duration gives no information on whether it was paid or not
- Only 11 loans (3%) have an associated credit card and all of them were paid
- Maximum of a single loan per account



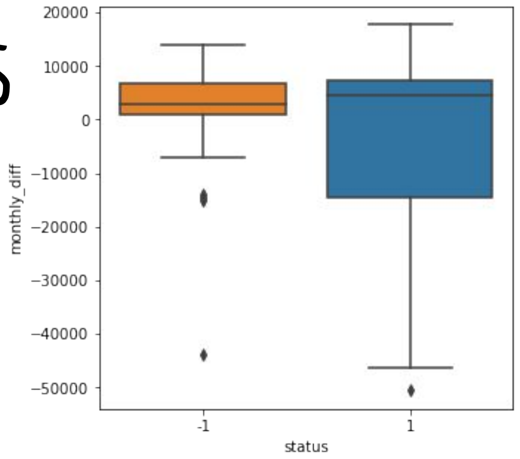


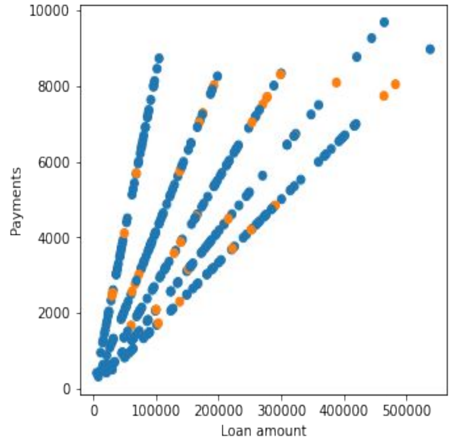


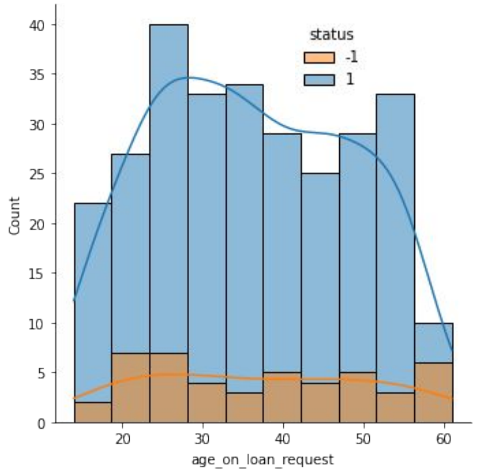


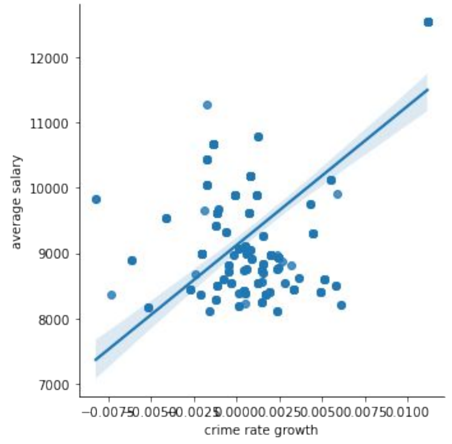
Exploratory data analysis

- The median value of the monthly diff* is higher on accounts of clients who paid their loan
- The bigger the loan amount, the bigger the payment
- Loan amount = loan payment * loan duration
- Ratio between paid and not paid is greater on age at loan above ~55 years.
- Growth of crime rate follows the average salary growth
- All loans associated to an account that has a disponent besides the owner (23%) were paid









* monthly diff - difference between the mean monthly credits and the mean monthly withdrawals for each account

Dimensions of Data Quality

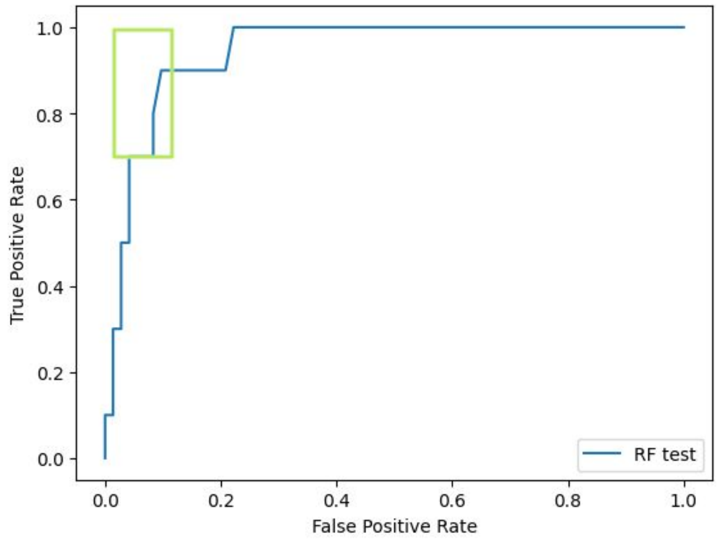
- Completeness
  - Although, there are many missing values, none of them can be considered mandatory -> the data is ***complete***
- Consistency
  - Overlapping info between *type* and *operation* of Transaction table -> the data is ***not consistent***
- Conformity
  - The client's birth date does not follow the format "yymmdd" since it adds 50 to the month to represent other gender -> the data is ***not conformant***
- Accuracy
  - Some transactions amounts and some consecutive negative balances seems suspicious -> the data is **not accurate**
- Integrity
  - There are orphaned record (e.g. many of the cards are not associated to a loan whose prediction is the purpose of the problem) -> the data **lacks integrity**
- Timeliness
  - The data is outdated since the dataset is from 1999 and we are in 2022 -> it **doesn't achieve timeliness**

Problem definition

- **Descriptive Problem:**
  - Identify groups of customer that represent relevant behaviors for the bank
- **Predictive Problem:**
  - Predict whether a given customer will be in debt. This is a **binary classification** problem, and the predicted value is either -1 (the customer will be in debt) or 1 (the customer will pay). This value corresponds to the **status** column in the table of loans.

Business and Data Mining Goals

- Reduce customers in debt significantly:
  - Issuing 0 loans (that would get paid) represents a loss for the bank (since the bank has expenses to pay), therefore it is not an option.
  - Our defined goal is to reduce customers in debt by 70%.
- Do not significantly reduce credit to “good customers”:
  - Give credit to at least 90% of the “good customers”



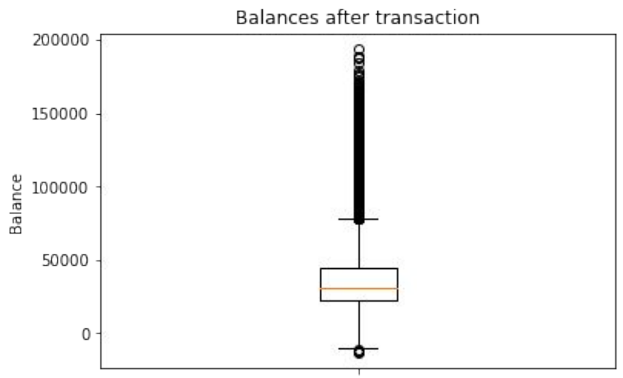
- The positive case is where the customer will be in debt (-1), so our model should have an ROC curve that, for a certain threshold, overlaps with the area defined by the conditions:
  - TPR > 0.70
  - FPR < 0.10

Data Preparation

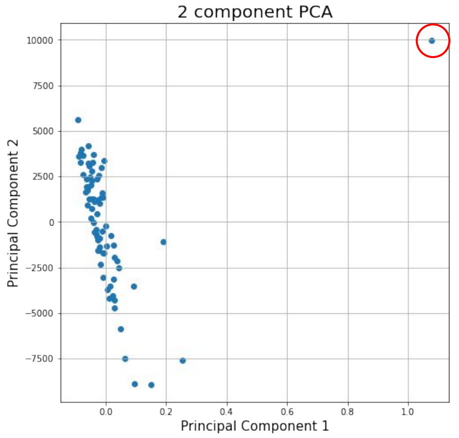
Data Integration

- Some date fields types were converted (e.g. integers dates were converted to date format)
- Some fields were renamed to facilitate the data preparation and data analysis
- All the tables available were joined into a single one.
    - The Transactions table was aggregated to have a single entry per *account_id*
    - In the Disposition table, the entries of a client that is not **owner** were removed to make the *account_id* unique
    - Join tables
        - The Loan table was left joined with Account table on *account_id*
        - The joined table was left joined with the modified Disposition table on *acccount_id*
        - The joined table was left joined with the Client table on *client_id*
        - The joined table was left joined with the Card table on *disposition_id*
        - The joined table was left joined with the aggregated Transaction table on *acocunt_id*
        - The joined table was left joined with the District table on *client_district_id*
    - Due to the join operator and the uniqueness of the joined feature, the number of entries of the Final table and the Loan table were equal as desired

Data Cleaning

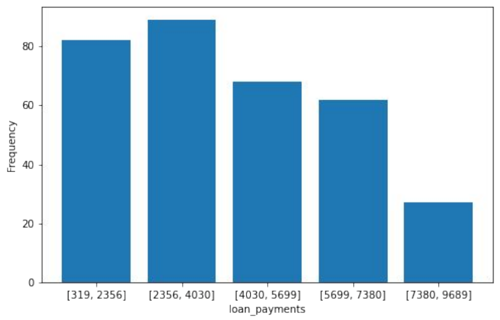


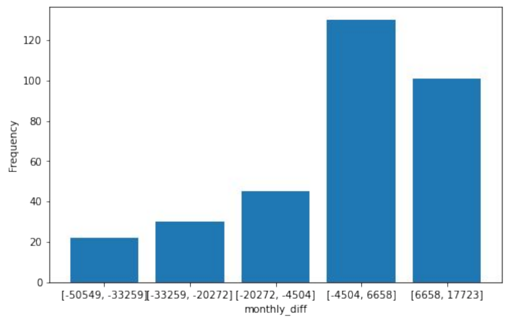
- It were not identified duplicated entries in any table
- Systematic removal of redundant fields on Final table
  - Fields with a correlation greater than 85% were removed
- Replace of missing values with complex and simple methods
  - Missing values on District table were replaced using a **linear regression** to obtain the data which best matches the entry. This was compared with the replacement by the field's **median** and **mean**
  - Others missing values were replaced using simpler methods
    - Field *operation* missing values in Transaction table were replaced by the **mode** of the respective transaction *type*
- Fields with more than 70% of missing values were removed
- Many outliers were found in District and Transaction table.
  - In District there is one district which have greater standards comparing to other regions, maybe because it's the capital
  - In Transaction the *amount* and *balance* field have many outliers, however there is no knowledge about its origin to attempt to address them.
- Since the reason of the outliers is unknown and the removal of them influences the prediction negatively, they were kept in the Final table

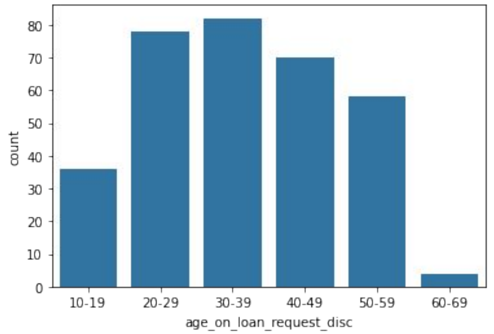


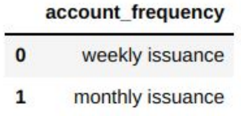
Data Transformation

- Discretization
  - Equal-width discretization
    - client’s age at the loan time
  - K-means discretization
    - loan *payments*
    - account *monthly_diff*
- Encoding
  - One-hot encoding
    - Account *frequency*
    - Client *gender*
  - Ordinal encoding
    - Others string type fields
- Rescaling
  - Standardization (Z-transformation)
  - Min-Max Scaling

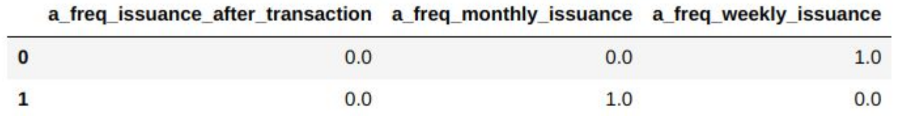








	account_frequency
0	weekly issuance
1	monthly issuance



	a_freq_issuance_after_transaction	a_freq_monthly_issuance	a_freq_weekly_issuance
0	0.0	0.0	1.0
1	0.0	1.0	0.0

Feature Engineering

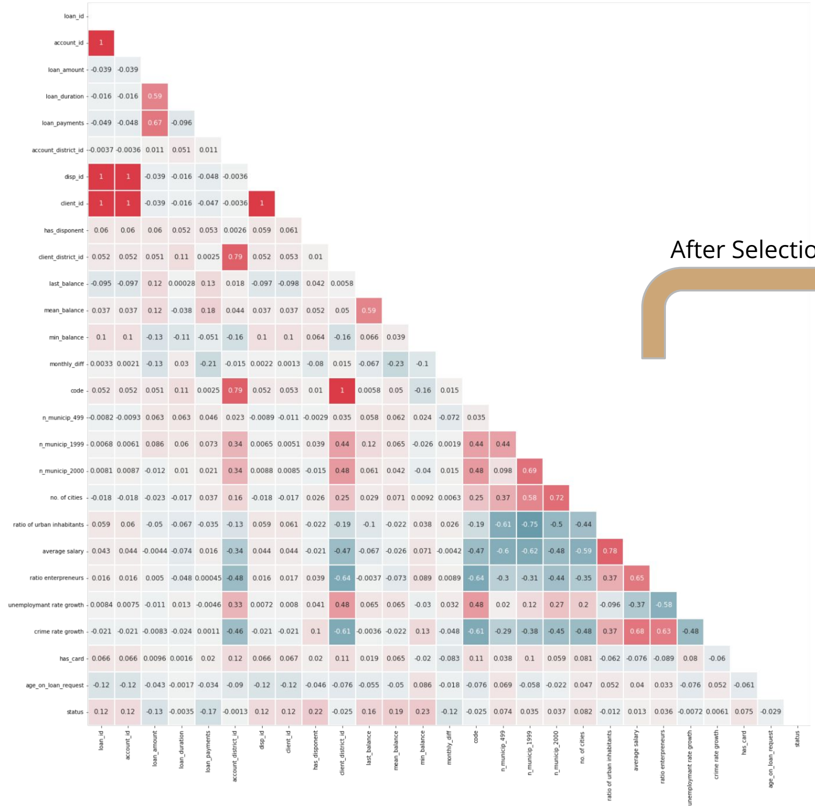
Some features were engineered for each individual table and for the final one. It was taking into account the business logic and the common sense.

- Client table
    - client *birth_date* and *gender* from client *birth_number*
- Disponent table
    - *has_disponent* that indicates if an account is associated with a client other than the owner. Those clients were removed from the table at the same time.
- District table
    - *entrepreneurs ratio*
    - *urban inhabitants ratio*
    - *unemployment rate growth*
    - *crime rate growth*
    - *no. of municipalities with inhabitants >2000*

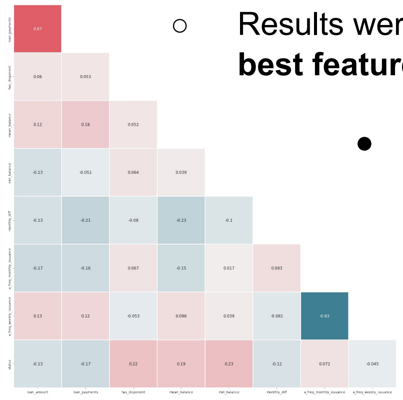
Feature Engineering

- Transaction table (aggregation grouping by *account_id*)
  - mean balance
  - minimum balance
  - last account balance before a loan be requested
  - negative balance (1 if the account ever had a negative balance, 0 otherwise)
  - difference between the monthly mean amount credited and the monthly mean amount withdrawn to obtain the mean *monthly_diff*. **This feature takes into account the business knowledge since it can be compared with the loan payments, which are also monthly**
- Final table
  - engineer *has_card*, by checking whether there was a card associated to the loan
  - difference between the date of the loan and the client birth date to obtain the *age at loan*

Feature Selection



After Selection



- **Filter Method:**
  - Removed all the features that were IDs
  - Removed features based on the **correlation threshold** (higher than 0.85)

- **Wrapper Method:**
  - Forward Selection
  - Backward Elimination
  - Bi-directional Elimination
  - Results were **compared** and the one with the **best features was selected** (by roc_auc)

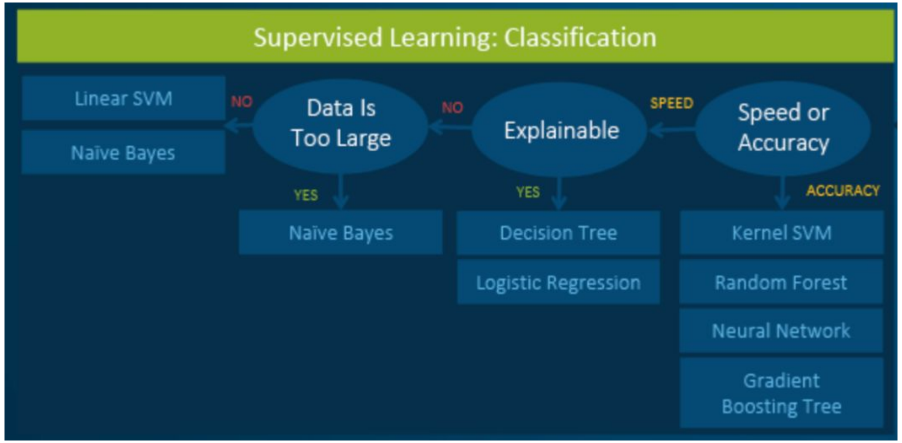
- **Best features combination**
  - loan_amount, loan_payments, has_disponent, mean_balance, min_balance, monthly_diff, a_freq_monthly_issuance, a_freq_weekly_issuance

Predictive Problem

Models

- **First considerations taken when choosing the model:**
  - Use algorithms that are **not restrictive -> more flexible algorithms**
  - Use **simple** algorithms that **take less time to train** (e.g. Decision Tree and Naive Bayes)
- **Then:**
  - Use **more complex** algorithms that usually has better results (e.g. Random Forest)
  - Use algorithms with different approaches, to achieve different bias

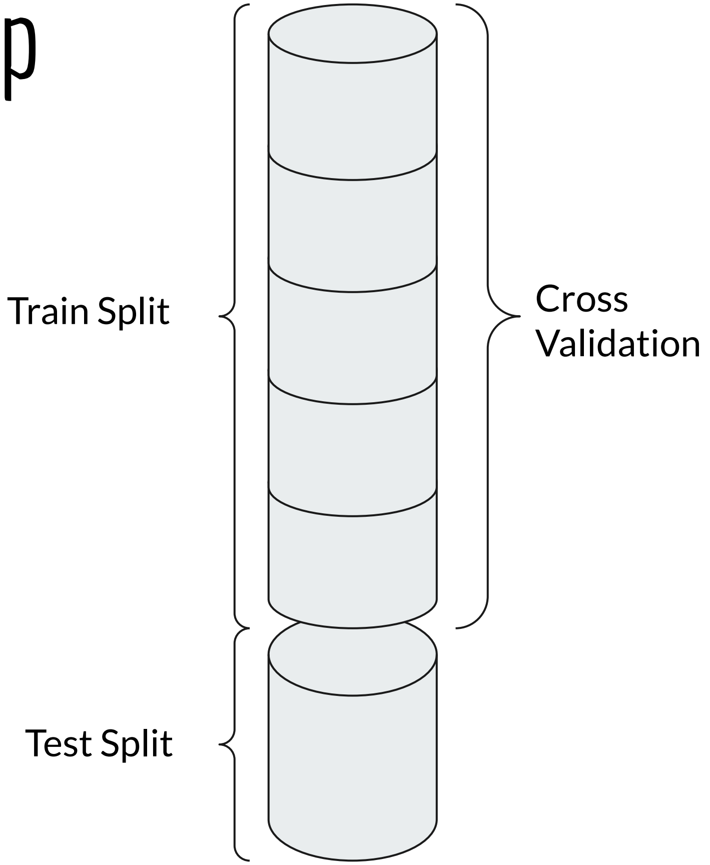
- **Models experimented:**
  - Decision Tree
  - K-nearest Neighbours
  - Naive Bayes
  - Support Vector Machine
  - Logistic Regression
  - Random Forest
  - Gradient Boost



Experimental Setup - Main Setup

Main setup:

- Import preprocessed data
- Split data using *Hold-Out* (75/25)
- Cross-Validation for parameter tuning on train split
  - Resample train data of each CV iteration imbalance ‘status’ class with Smote
  - Score with AUC
- Save best parameters and best score in file
- Fit the model with the train split using the best parameters
- Generate predictions of test split
- Evaluate
  - **AUC**, ROC, Precision, Recall, F-measure, Accuracy
- Repeat process to different models
  - Decision Tree, K Nearest Neighbor, Support Vector Machine, Logistic Regression, Random Forest, Gradient Boost



Experimental Setup - Secondary Setups

Hold-out

- Import preprocessed data
- Split data using *Hold-Out* (75/25)
- Use Smote on train split (optional)
- Rescale train and test split using Standardization or MinMax Scaling (optional)
- Fit different models with train split
  - DT, KNN, MLP, NB, SVM, LR, RF, GB
  - Hardcoded parameters
- Generate predictions of test split for the different models
- Evaluate all the models
  - **AUC**, ROC, Precision, Recall, F-measure, Accuracy, **Fit Execution Time**

Cross-Validation

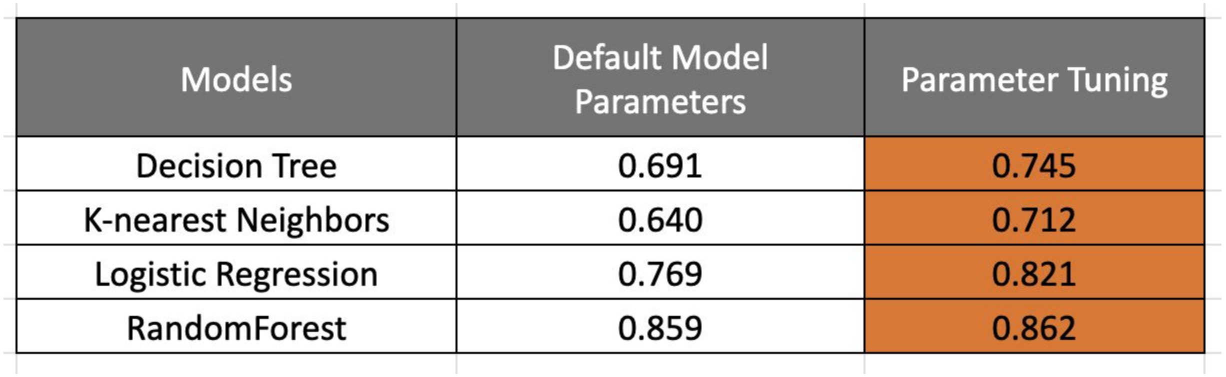
- Import preprocessed data
- Stratified 5 Fold repeated 3 times
- Use Smote on train split of each CV iteration (optional)
- Use different models
  - DT, KNN, MLP, NB, SVM, LR, RF, GB
  - Hardcoded parameters
- Evaluate all the models
  - **AUC** mean
  - Compare the 15 (5*3) evaluations distribution among all the models using boxplots

Kaggle

- Import preprocessed dev data
- Import preprocessed competition data
- Use Smote on dev data (optional)
- Rescale data using Standardization or MinMax Scaling (optional)
- Fit different models with dev data
  - DT, KNN, MLP, NB, SVM, LR, RF, GB
  - Hardcoded parameters
- Generate predictions of competition data for the different models
- Generate submission csv file

Hyperparameter Tuning

- **Grid Search was used**
    - GridSearchCV from Python sklearn
    - Simulates every combination of the value ranges
    - Cross Validation with k = 5 and 3 repetitions
- **Results:**
    - Overall the **performance of the models was improved**, however only a **slight improvement was seen from Random Forest**
    - Explanation: Might be because the default parameters of the Random Forest was already quite optimal for the dataset we had
    - Below we can observe the results (metric: AUC) seen before and after applying with the parameters found using grid search:



Models	Default Model Parameters	Parameter Tuning
Decision Tree	0.691	🟧 0.745
K-nearest Neighbors	0.640	🟧 0.712
Logistic Regression	0.769	🟧 0.821
RandomForest	0.859	🟧 0.862

🟧 **Best Result**

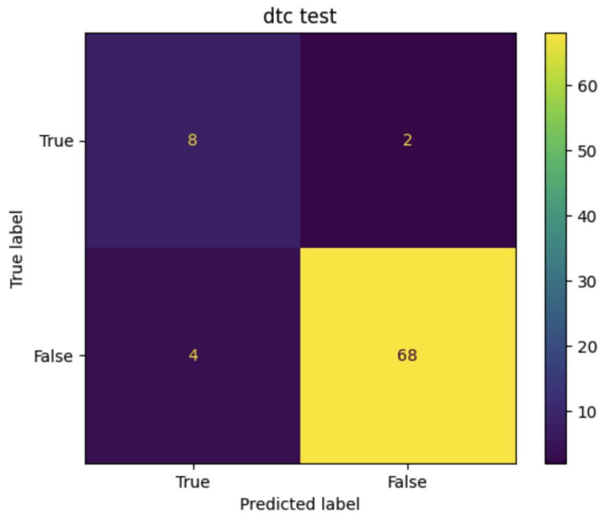
Decision Tree

- **Characteristics:**
  - Creates a model that predicts the value of a target variable of a target variable by learning simple decision rules inferred from the data features

    Benefits:
    - Simple to interpret, since it is a white box model
    - Handles numerical and categorical data
    - Requires little data preparation compared to others

    Drawbacks:
    - May create over-complex trees, overfitting

- **Results obtained:**
  - AUC: 0.872
  - Recall: 0.8
  - Precision: 0.67
  - F_Measure: 72.73



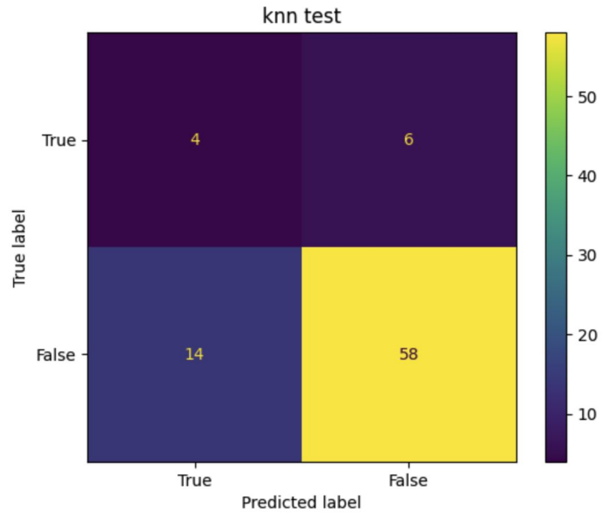
K-nearest Neighbours

- **Characteristics:**
  - Finds the closest similar points, taking K nearest neighbors based on Euclidean distance, following basic steps:
    - Calculate distance
    - Find the closest neighbors
    - Vote for the labels

    Benefits:
      - Simple to interpret
      - Does not need any training on data

    Drawbacks:
      - Need to find the optimal k, which can be hard
      - Sensitive to outliers

- **Results obtained:**
  - AUC: 0.778
  - Recall: 0.4
  - Precision: 0.22
  - F_Measure: 28.57



Naive Bayes (The Worst)

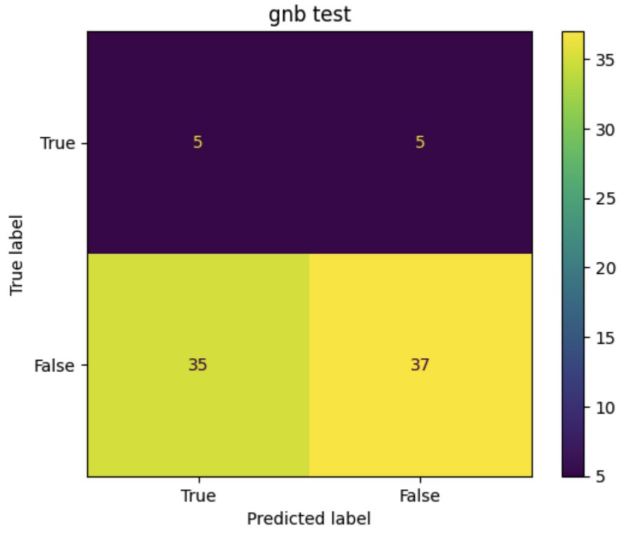
- **Characteristics:**
  - Group of supervised machine learning algorithms based on Bayes Theorem, which finds the probability of an event occurring given the probability of another event that has already occurred

$$P(A|B) = \frac{P(A \bigcap B)}{P(B)} = \frac{P(A) \cdot P(B|A)}{P(B)}$$

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

Benefits:
- Not sensitive to irrelevant features
- It is fast

Drawbacks:
- Assumes all the features are independent
- Lousy estimator



- **Results obtained:**
  - AUC: 0.544
  - Recall: 0.5
  - Precision: 0.12
  - F_Measure: 20.0

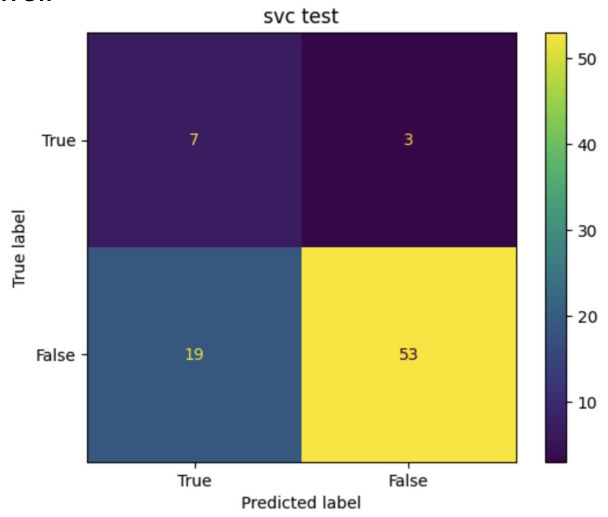
Support Vector Machine

- **Characteristics:**
  - Tries to find a hyperplane in a N-dimensional space (N = n. of features), which distinctly classifies the data points

    Benefits:
    - More effective in high dimensional spaces
    - When classes in the data are well separated, works really well

    Drawbacks:
    - Need to find the optimal kernel function, can be difficult

- **Results obtained:**
  - AUC: 0.789
  - Recall: 0.5
  - Precision: 0.22
  - F_Measure: 30.77



Logistic Regression

- **Characteristics:**
  - Find the probability of event success and failure. It learns a linear relationship from the given dataset and then introduces a non-linearity in the form of the Sigmoid function.

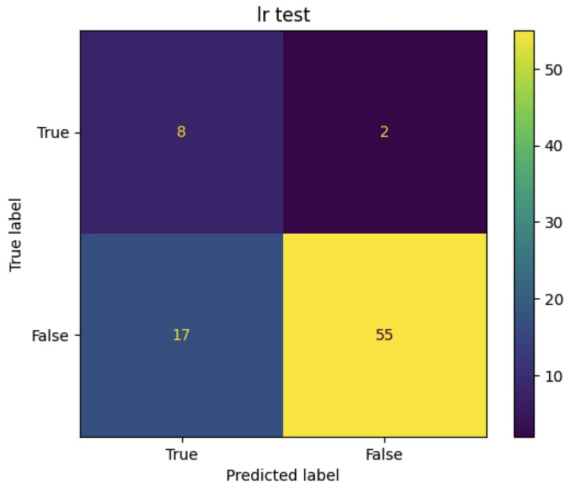
Benefits:

- Easy to interpret and efficient to train

Drawbacks:

- May to lead to overfitting if n. observations < n. features

- **Results obtained:**
  - AUC: 0.892
  - Recall: 0.8
  - Precision: 0.32
  - F_Measure: 45.71



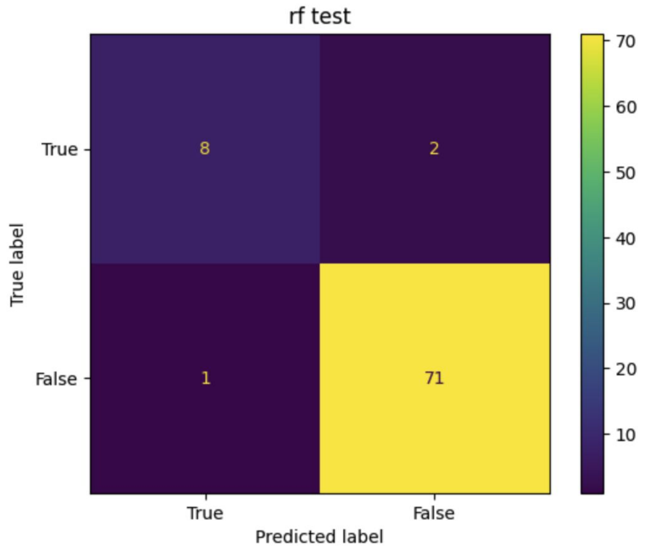
Random Forest

- **Characteristics:**
  - Based on the concept of ensemble learning, which combines multiple classifiers to improve the performance of the model, in case, It combines a various number of trees on a number of subsets of a given dataset and takes the average to improve the performance.

    Benefits:
    - Enhances the accuracy
    - Prevents overfitting issues
    - Can handle large datasets with high dimensionality

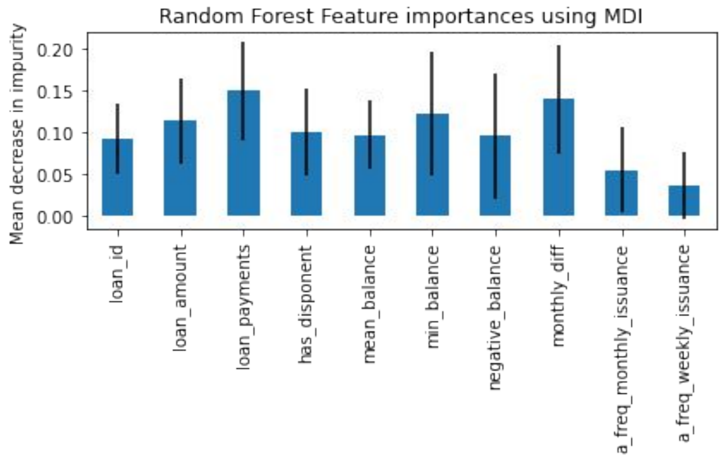
    Drawbacks:
    - It is a black-box model, therefore hard to interpret and explain

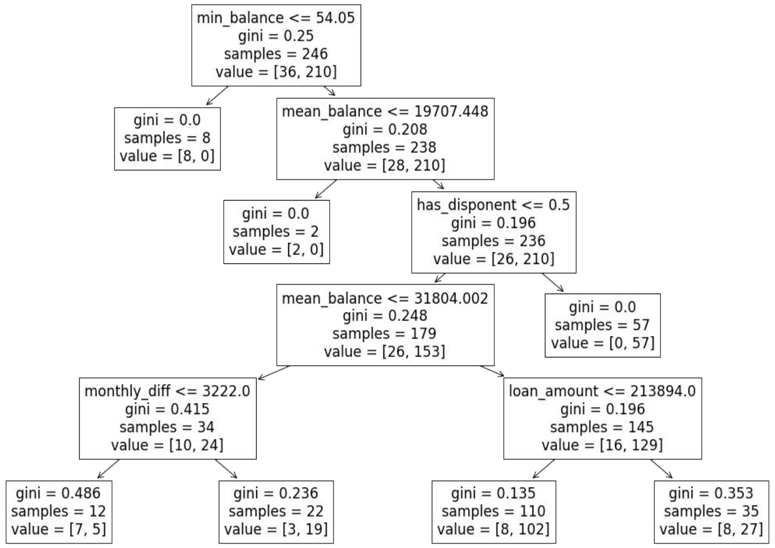
- **Results obtained:**
  - AUC: 0.972
  - Recall: 0.8
  - Precision: 0.89
  - F_Measure: 84.21



Results - Feature Importance

- In the decision tree generated with the DTC model, it is possible to observe that the features engineered were useful to that model. Some were even capable to split with a *gini* of 0
- The ones higher in the tree have more importance compared to the ones below.
- The Random Forest feature importance looks similar to the one of Decision Tree, what is expected due to its similarities

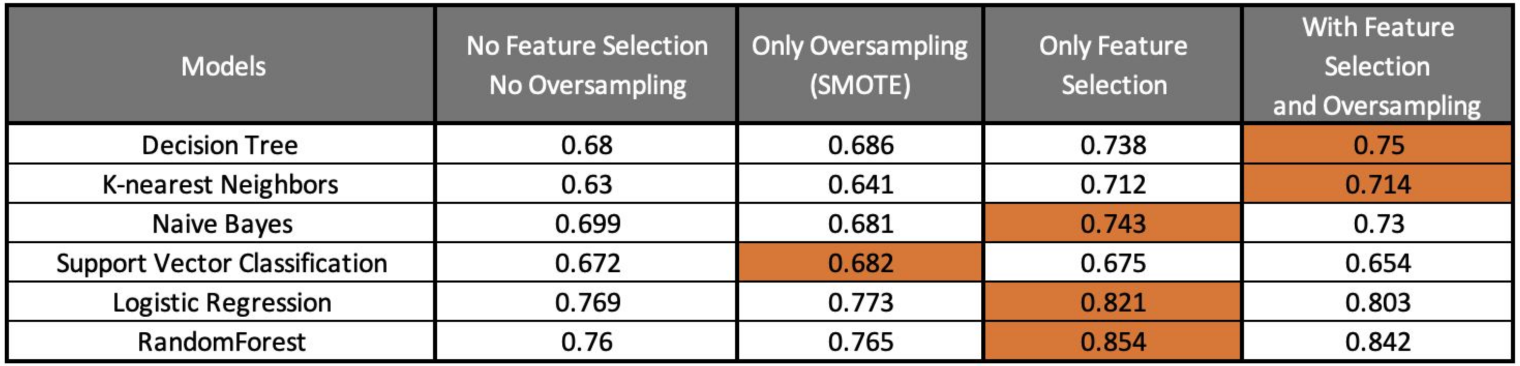



min_balance <= 54.05 gini = 0.25 samples = 246 value = [36, 210]
gini = 0.0 samples = 8 value = [8, 0]
mean_balance <= 19707.448 gini = 0.208 samples = 238 value = [28, 210]
gini = 0.0 samples = 2 value = [2, 0]
has_disponent <= 0.5 gini = 0.196 samples = 236 value = [26, 210]
mean_balance <= 31804.002 gini = 0.248 samples = 179 value = [26, 153]
gini = 0.0 samples = 57 value = [0, 57]
monthly_diff <= 3222.0 gini = 0.415 samples = 34 value = [10, 24]
loan_amount <= 213894.0 gini = 0.196 samples = 145 value = [16, 129]
gini = 0.486 samples = 12 value = [7, 5]
gini = 0.236 samples = 22 value = [3, 19]
gini = 0.135 samples = 110 value = [8, 102]
gini = 0.353 samples = 35 value = [8, 27]

Results

- **Does Feature Selection and Oversampling improve the results?**
  - Different models were tested using 4 different setups
  - Results before and after using feature selection and oversampling were compared
  - Cross Validation with k = 5 and 3 repetitions
  - **Oversampling has no significant impact** on the results
  - However **Feature selection does improves the results** in every model
  - Other benefit observed by applying feature selection, was the **reduction in execution time**
  - The best result for each model was seen in different setups, however the best results for most models were seen by only using feature selection

AUC for ROC for 4 different variants of Cross-Validation experimental setup



Models	No Feature Selection No Oversampling	Only Oversampling (SMOTE)	Only Feature Selection	With Feature Selection and Oversampling
Decision Tree	0.68	0.686	0.738	**0.75**
K-nearest Neighbors	0.63	0.641	0.712	**0.714**
Naive Bayes	0.699	0.681	**0.743**	0.73
Support Vector Classification	0.672	**0.682**	0.675	0.654
Logistic Regression	0.769	0.773	**0.821**	0.803
RandomForest	0.76	0.765	**0.854**	0.842

Note*: The models were tested using the default parameters (no hyper parameter tuning)

🟧 **Best Result**

Results

The Best - Random Forest

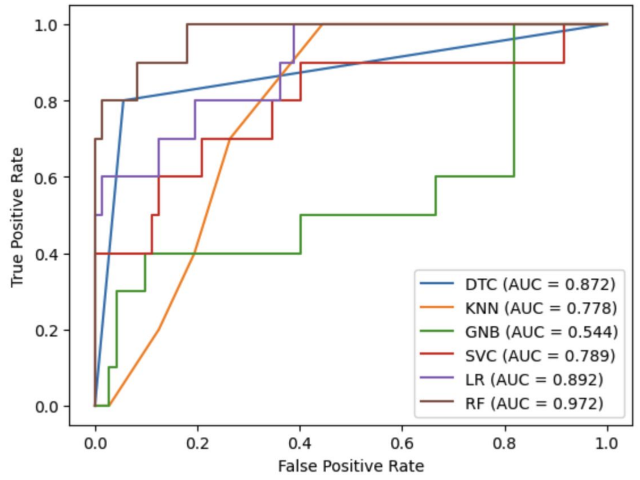
- Overall the best results were found by using Random Forest
- A score of 0.972 AUC was seen in the hold-out setup
- The submission in Kaggle gave 0.941 AUC on the public score and 0.889 on private score
- The difference of scoring might be due to overfitting

Other alternatives models:

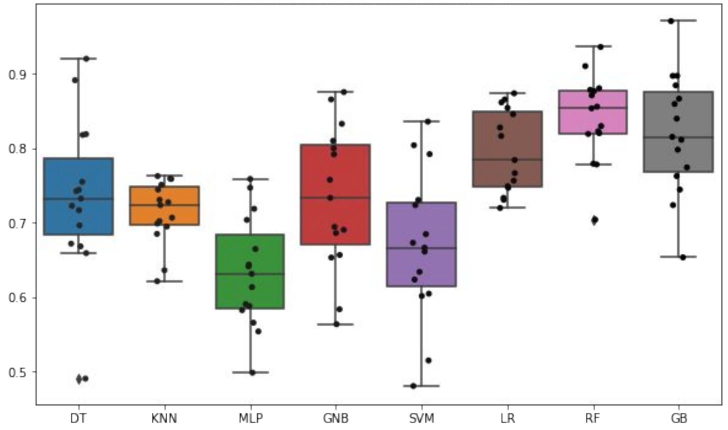
- Other models that had great results were Logistic Regression and Gradient Boosting
- Gradient Boosting results were great, however, we were not able to look and test further into this model

Evaluation analysis

- The figure at the bottom shows that the score obtained in cross validation can vary a lot, probably due to the low quantity of training data
- The figure at the top shows in general better results, however it seems it was overfitting, once the kaggle submissions showed that the evaluation more accurate was the one obtained with the cross validation



AUC for each model with CV and SMOTE

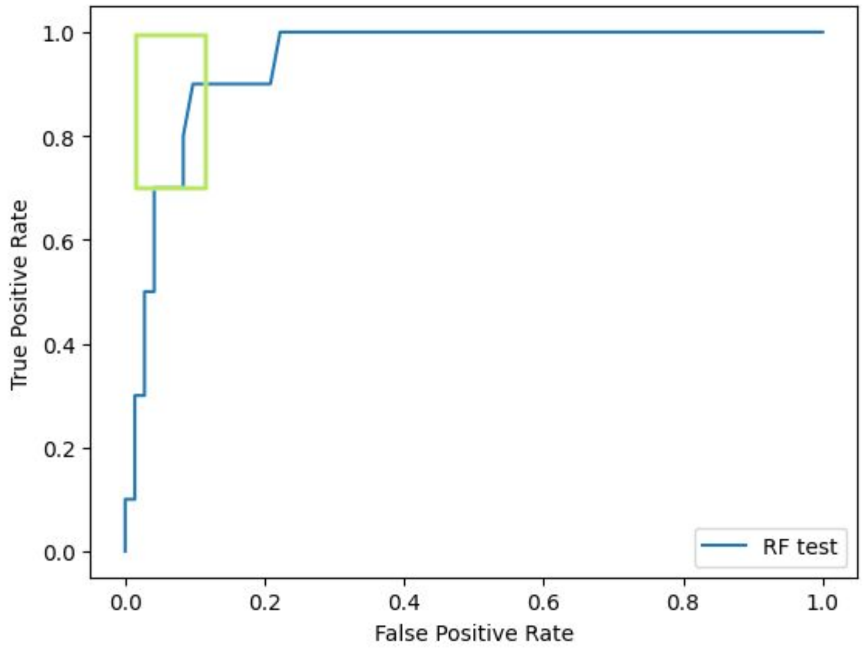


Predictive Problem

- The data mining goal was successfully achieved:
  - TPR > 0.70

    and
  - FPR < 0.10

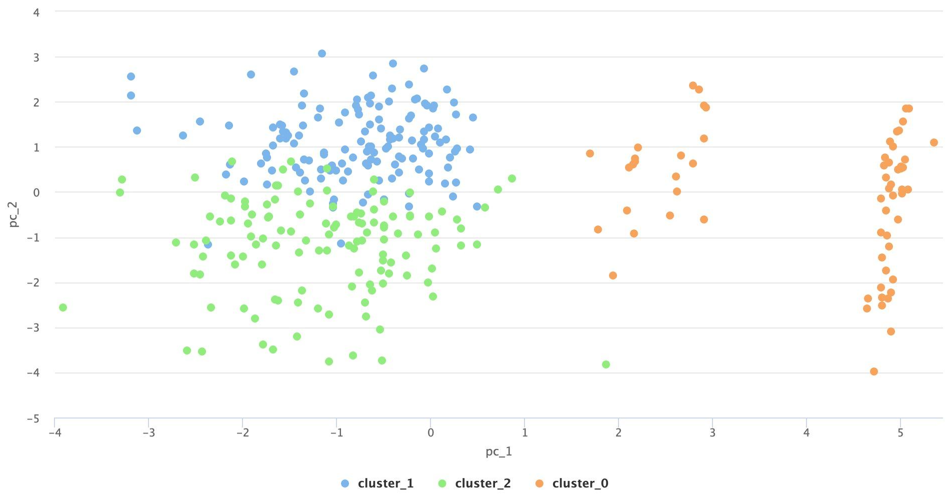
✓

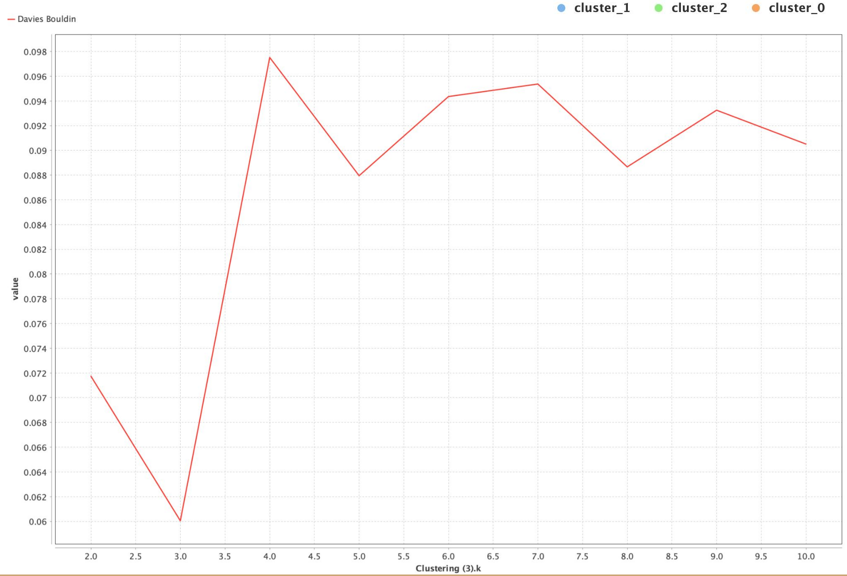


Descriptive Problem

Descriptive Problem

- **Evaluation Measures:**
  - Davies-Bouldin Index
- **Visualization:**
  - PCA
  - SVD
- **Algorithms:**
  - k-means
  - k-medoids
- **Tools:**
  - RapidMiner Studio





Descriptive Problem

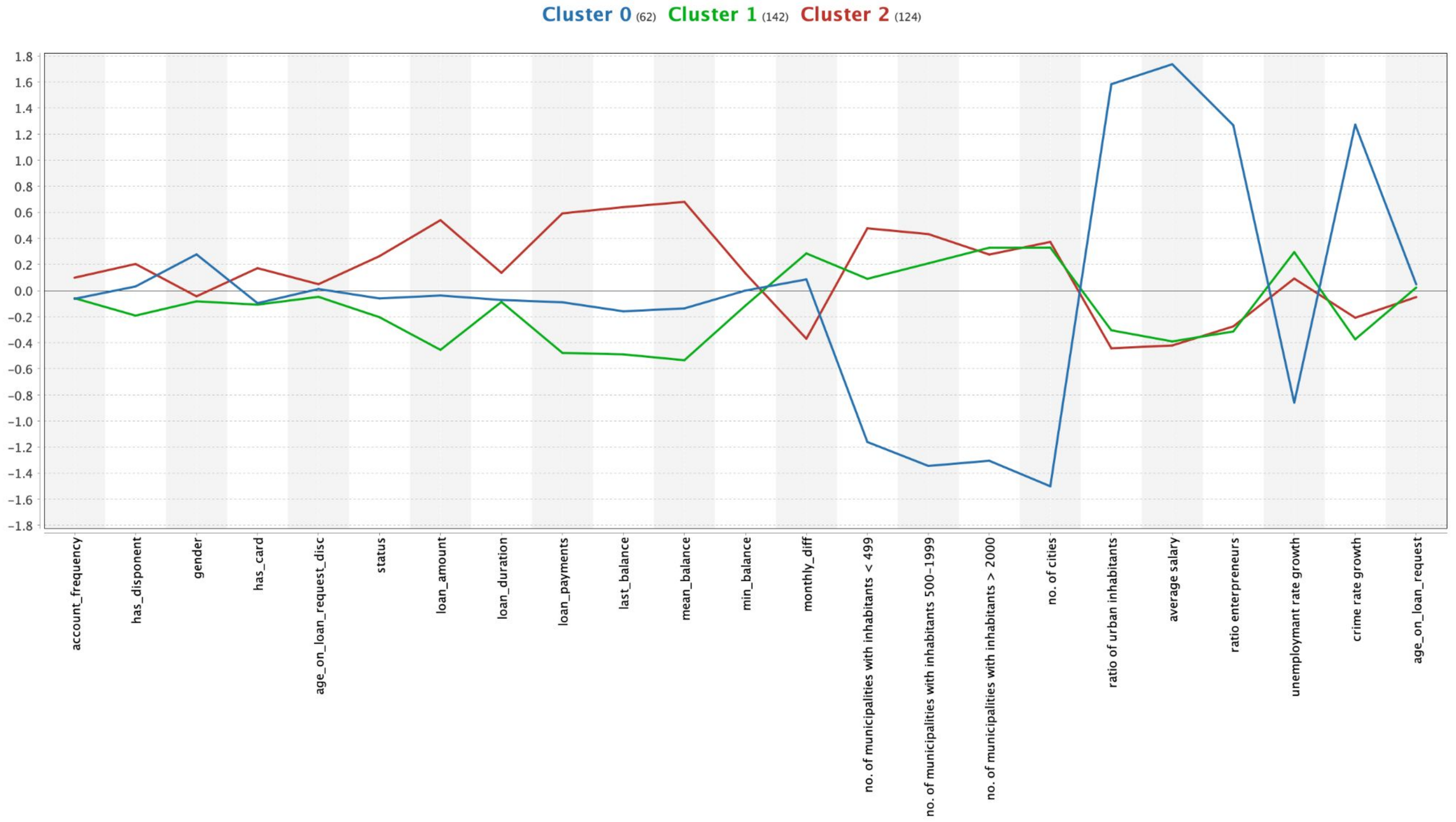
- **Evaluation measures:**
  - The elbow method was first use to evaluate the optimal number of clusters in each algorithm. However, after using the z-transformation to normalize the data, it became difficult to choose the optimal k.
  - Davies-Bouldin Index was chosen instead. This metric is lower the smaller the intra-cluster distances and the bigger the inter-cluster distances are.

- **Visualization**:

  - PCA and SVD both resulted in similar visualization plots. The one presented in the previous slide is the output of a PCA.

- **Data chosen:**
  - Several datasets were tested, but the one which was the result of merging every dataset supported all the conclusions we found using each dataset separately. The features used were the following:
    - account_frequency, has_disponent, gender, has_card, age_on_loan_request_disc, status, loan_amount, loan_duration, loan_payments, last_balance, mean_balance, min_balance, monthly_diff, no. of municipalities with inhabitants < 499, no. of municipalities with inhabitants 500-1999, no. of municipalities with inhabitants > 2000, no. of cities, ratio of urban inhabitants, average salary, ratio enterpreneurs, unemploymant rate growth, crime rate growth, age_on_loan_request.

Descriptive Problem



Descriptive Problem

- **Algorithms:**
  - Both k-means and k-medoids were tested. Intuitively, k-medoids is similar to k-means but uses actual data points as “centroids”, rather than abstract ones.
  - The optimal k for the k-medoids algorithm was always higher than that of the k-means, which made it more difficult to gather conclusions from, so we ended up using the result from k-means to gather our conclusions.

- **Conclusions:**
  - **Cluster 1**: Clients have higher *mean balance*, *last balance*, *loan_amount*, *status* and lower *monthly_diff*. *has_card* is also higher than average, as well as *has_disponent*.
  - **Cluster 2**: Clients share some similarities with **cluster 1**, but most attributes mentioned above are inverted, namely *mean balance*, *last balance*, *loan_amount* and *status* are lower and *monthly_diff* is higher. *has_disponent* is also lower than average.
  - **Cluster 3**: Clients are from more urban areas (with a higher *ratio of urban inhabitants*) where the *average salary*, the *ratio enterpreneurs* and the *crime rate growth* is way higher than average and the *no. of cities* and the *unempolymant rate growth* is way lower, have a tendency to be female (higher *gender*) and the value of *status* is very close to the average.

Project Management, Tools and Individual Factor

- To manage the project content, collaborate and communicate were used:
  - GitHub, Discord
- To develop the project were used:
  - Jupyter Notebook, Python, RapidMiner
- In python, were used the following packages:
  - Pandas, Numpy, Matplotlib, Seaborn, scikit-learn, imblearn, mlxtend
- Attempt to follow CRISP-DM in an agile way by building the 5 first phases (business understanding, data understanding, data preparation, modeling, and evaluation) and then iterate through them, trying to improve the outcome in each iteration.

Name (Number)	Contribution [0,1]
Fábio Huang (up201806829)	1
Henrique Nunes (up201906852)	1
Luís Guimarães (up202204188)	1

Conclusions

- Parameter tuning has a relevant impact on the results
- Random Forest was the best model among the ones tested. Even so, other models like Logistic Regression and Gradient Boost could be improved with a deeper exploration of its parameters and with some features changes that would better fit them.
- The small quantity of data makes the evaluation of the models harder, decreasing the confidence about the results obtained in the evaluation phase
- Although it depends on the experimental setup, the defined data mining goals were achieved but its confidence couldn’t measured

Future Work

- Improve feature engineering to have better data to train
- Split experimental setups by model so that a setup could be more adjusted to a specific model instead of having a general setup and observe which model better fits it
- Use other more complex models such as Neural Networks
- Generalize wrapper method of feature selection to select features depending on the model