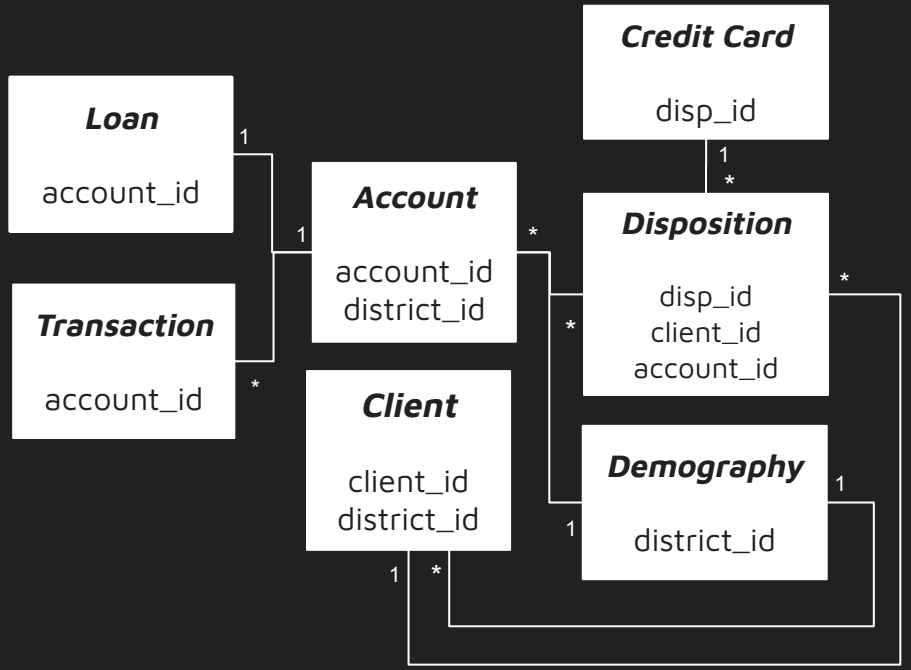
DATA MINING PROJECT

To loan or not to loan - that is the question

Group 44
Diana Freitas, up201806230
Mariana Ramos, up201806869
Paulo Ribeiro, up201806505

Professor Carlos Soares

DOMAIN DESCRIPTION



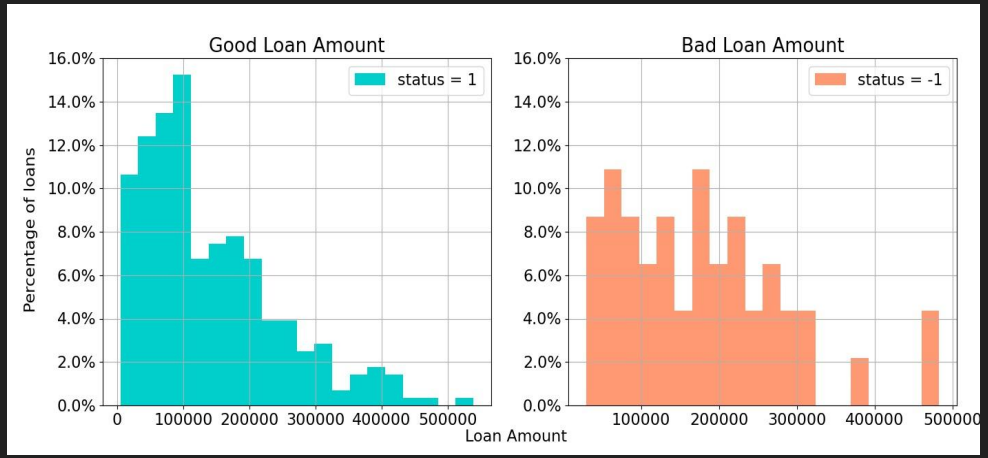
Records of Czech bank from 1993 to 1998.

- 4500 accounts
- 5369 clients
- 682 loans
- 77 districts
- 1056320 transactions
- 892 credit cards

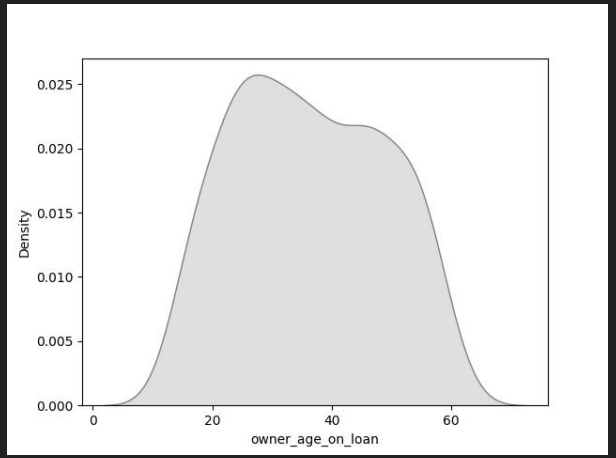
EXPLORATORY DATA ANALYSIS

- Status class is unbalanced;

- Loans with a higher ‘amount’ tend to be fraudulent;



- Age distribution follows expected pattern;

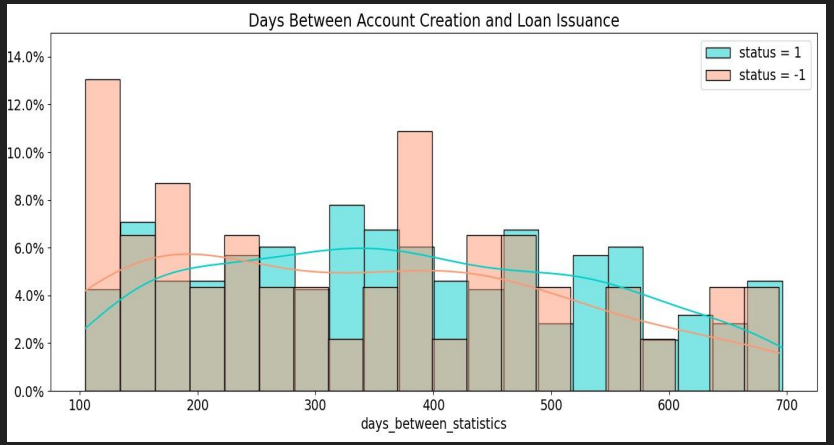


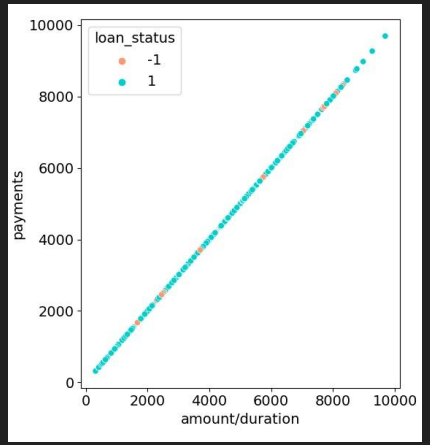
EXPLORATORY DATA ANALYSIS

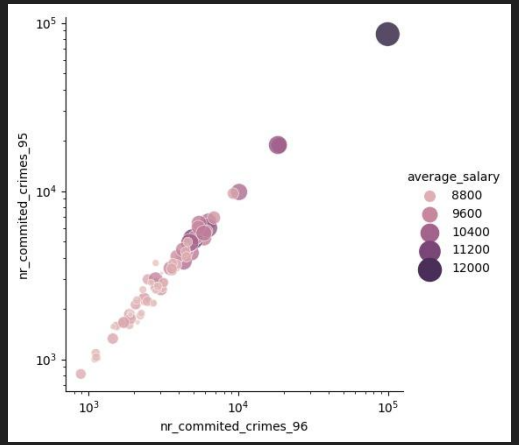
Fraud tends to occur on recently created accounts;

Correlation between loan amount payments and duration amount/duration = payment;

Districts with more criminality are the ones with higher salary;







EXPLORATORY DATA ANALYSIS

USER	No user made more than one loan
CREDIT CARD	The credit card type more used is the Classic one - 70%
LOAN STATUS	No correlation between gender / loan duration and loan status
DISTRICT	One of the districts has 663 clients
COMPETITION SET	In the competition set, all loans are from recent years
TRANSACTION	No transactions were made after a loan

PREDICTIVE DATA MINING TASK

PROBLEM DEFINITION

GOAL

To help bank managers minimize the losses, by avoiding trusting loans to non-compliant clients. We want to **predict if a given loan will end successfully** based on data about the clients and previous loans.

NATURE

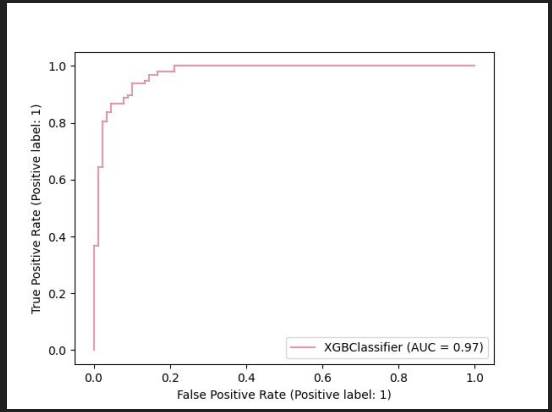
- Binary Classification
- Positive Class = default loan (-1)
- Probability of being a default loan

PERFORMANCE METRICS

- FP = successful loan predicted as default
- TP = default loan predicted as default
- **FN = default loan predicted as successful**
- TN = successful loan predicted as successful

PERFORMANCE METRICS – AUC

- How much the model is capable of distinguishing between default loans and successful loans;
- Higher AUC ⇒ better at predicting default loans as default and successful loans as successful.



DATA PREPARATION

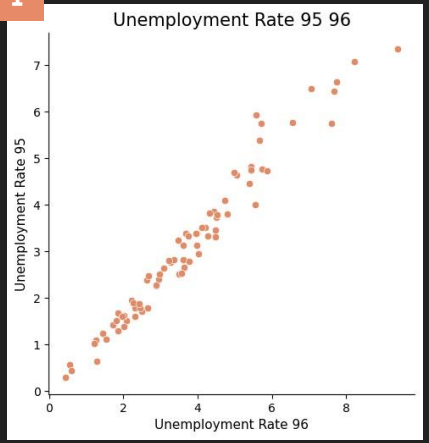
MISSING DATA

- **Unemployment Rate '95** and **No. of Committed Crimes '95** had missing values for 'Jesenik' district - replaced by '96 values; [1, 2]
- Bank, account, operation, k_symbol - justified in slide 25.

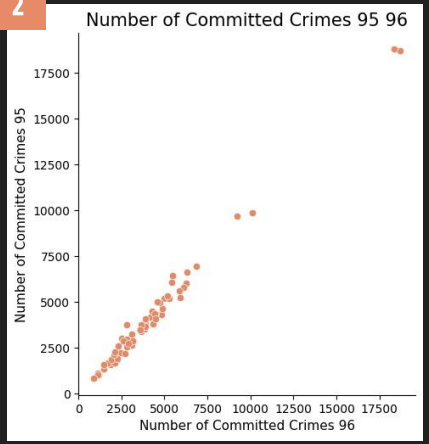
OUTLIERS

- Detected in: **No. of Inhabitants** [3], **No. of Committed Crimes `95** and **`96** and **Unemployment Rate '95** and **`96**;
- The low amount of domain details doesn't allow solid conclusions about their significance.

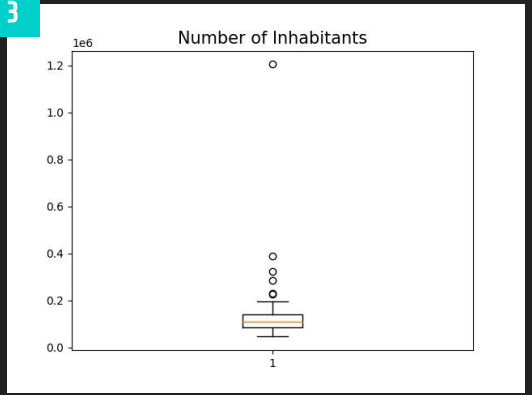
1



2



3

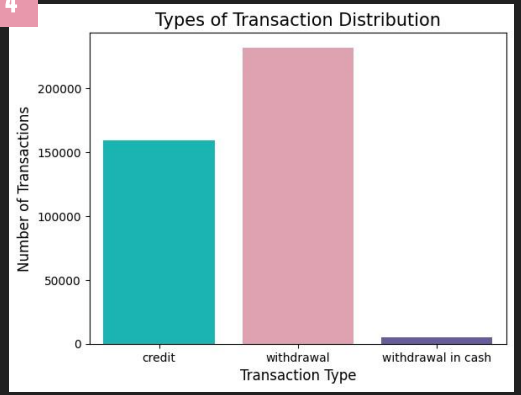


DATA PREPARATION

MISLABELED DATA

Transaction type 'withdrawal in cash' changed to 'withdrawal' - **operation** already described the mode (cash/credit) of the transaction. [4]

4



DATA TRANSFORMATION

- Negative **Amount** on withdrawals;
- Encode Categories - **LabelEncoder;**
- Scaling - **StandardScaler, MinMaxScaler.**

IRRELEVANCY

Removed ID's and **district_name.**

IMBALANCED DATA

Oversampling with **SMOTE.**

FEATURE ENGINEERING

(aggregation and domain knowledge)

Client Features

- Age on loan & gender

Demographic Features

- Criminality growth & unemployment growth
- Ratio of urban inhabitants (transformed)
- Average crimes & average unemployment
- Ratio of entrepreneurs

Financial Features

- Has disponent & has card
- Days between account creation and loan issuance
- Avg., Std. Deviation, Min. for balance
- Number of transactions
- Had negative balance, Last Balance
- Credit Ratio
- Min., Max. and Avg. amount
- For each operation & k_symbol mean amount and ratio

FEATURE SELECTION

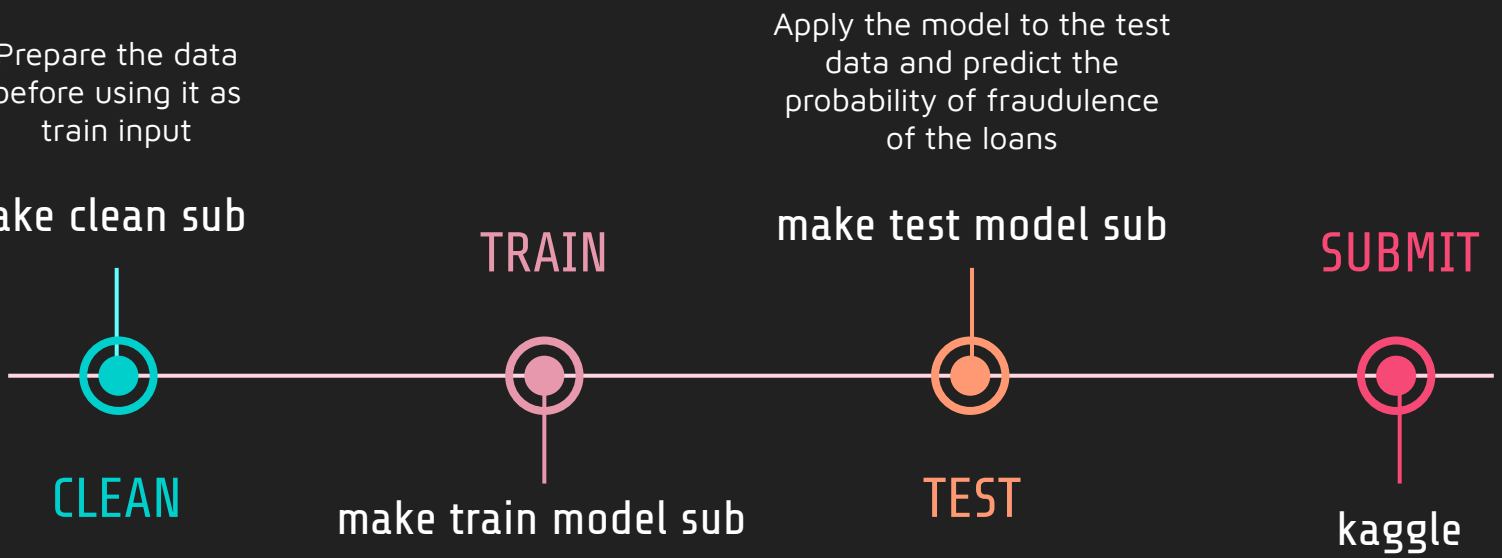
Methods:

- **Wrapper** Based (Backward & Forward Elimination) - *RFECV*, *RFE*, *SFS*.
- **Filter** Based (*SelectKBest* with f_classif) - correlation.
- Ensemble learning - *ExtraTreeClassifier.*

Strategies:

- Drop highly correlated features (**redundancy**)
  e.g. Removed **duration** (= **amount** / **payments**).
- Correlation between features and status (**relevancy**).
- Discard features with many missing values - bank, account.

EXPERIMENTAL SETUP PIPELINE



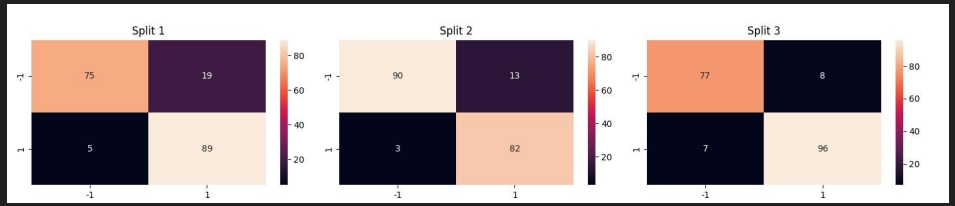
RESULTS

	AVERAGE TRAIN AUC	AVERAGE TEST AUC
LOGISTIC REGRESSION	0.917	0.9037
K-NEAREST NEIGHBORS	1.0	0.961739
XGBOOST	1.0	0.967639
RANDOM FOREST	1.0	0.970588
BAGGING	1.0	0.962986
NEURAL NETWORK	0.983	0.921393

ACCURACY AND STANDARD DEVIATION MEASURES

```
DECISION TREE: 0.85 accuracy with a standard deviation of 0.03
LOGISTIC REGRESSION: 0.81 accuracy with a standard deviation of 0.01
RANDOM FOREST: 0.91 accuracy with a standard deviation of 0.02
GRADIENT BOOSTING: 0.90 accuracy with a standard deviation of 0.02
SVM: 0.88 accuracy with a standard deviation of 0.02
KNN: 0.86 accuracy with a standard deviation of 0.03
NEURAL NETWORK: 0.85 accuracy with a standard deviation of 0.02
XGBOOST: 0.90 accuracy with a standard deviation of 0.02
BAGGING: 0.89 accuracy with a standard deviation of 0.02
```

CONFUSION MATRIX FOR DIFFERENT SPLITS IN CROSS VALIDATION



PUBLIC SCORE

0.97654

PRIVATE SCORE

0.93580

BEST ALGORITHM

XGBoost

16 FEATURES SELECTED

'has_disponent', 'last_balance', 'avg_balance', 'credit_ratio' ...

BEST N. OF SPLITS

3

DESCRIPTIVE DATA MINING TASK

CLUSTERING

1. **Personal** profile - only personal information;
2. **Demographic** profile - district details;
3. **Transaction** profile - transaction features;
4. **Economic** profile - transactions, loans & salary.

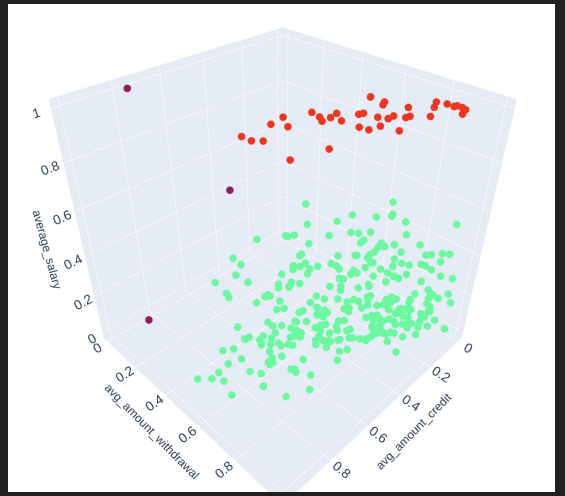
TESTED ALGORITHMS

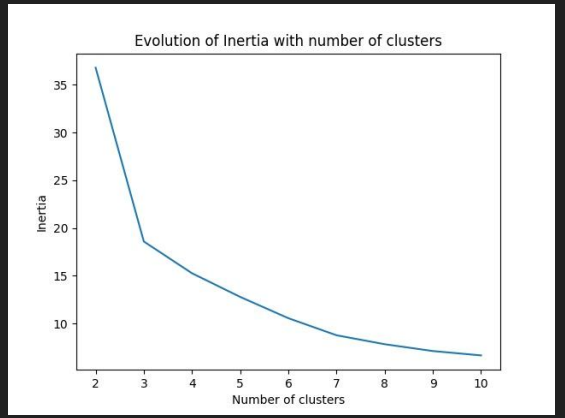
- DBSCAN- partitional
- K-Means - partitional
- K-Medoids (PAM) - partitional
- Agglomerative - hierarchical
- PCA for feature reduction

VALIDATION & PARAMETER TUNING

- Silhouette Coefficient
- Elbow-Method/Inertia

silhouette	epsilon	minPoints
0.495001	0.2	2
0.495001	0.2	3
0.495001	0.2	4
0.437134	0.3	2
0.437134	0.3	3
0.437134	0.3	4
0.527313	0.4	2
0.527313	0.4	3
0.527313	0.4	4
0.527313	0.5	2
0.527313	0.5	3
0.527313	0.5	4





ANNEXES

BUSINESS UNDERSTANDING

REQUIREMENTS OF THE END USER

- Help bank managers improve their understanding of the bank customers in order to identify high-value customers and minimize the bank losses. By knowing which customers to trust a loan, loaning to fraudulent and non-compliant clients could be avoided, based on the loan characteristics.

BUSINESS GOALS

- Identify most of the loans that will default;
- Answer the question "To loan or not to loan?";
- Determine if a given loan will end successfully, given data about the clients, previous loans and transactions.

DATA MINING GOALS

- Achieve the most accurate values for the loan success probability of each client, given that the output may vary between 0 (paid loan) and 1 (default loan).
- Reduce the number of False Negatives (default loan predicted as successful) and obtain the **maximum AUC.**
- This implies an increase of the True Positive Rate, therefore the desired ROC curve should define a higher TPR for small FPR values.

This project is categorized as a **Classification Binary Problem**, and we used the **Area Under ROC Curve (AUC) to evaluate its performance**.
The ROC (Receiver Operating Characteristic) curve is a probability curve that plots the True Positives Rate against False Positives Rate at various threshold values.
Since the status class is unbalanced (a lot more loans were paid than not paid), **one False Negative has much more impact than one False Positive**. For the bank, if they consider a bad loan as good (False Negative) they will lose money. **This is why our goal is to increase the AUC**, making sure this False Negatives impact is kept.

TOOLS

PROJECT MANAGEMENT AND COLLABORATION

When it comes to a collaborative project it is crucial to create a **plan of action** and **follow a methodology.** After understanding the business goals, translating them to data mining goals and deciding how we were going to evaluate our predictions, we focused on the collaboration tools necessary to distribute the work.

We decided to use:

- **GitSCM**: For collaboration between the group members through issues and version control;
- **Trello**: distribute major tasks;
- **Google Slides**: To build the report.

Besides these collaborations tools our plan also included the creation of the experimental setup Pipeline (cleaning - train - test - submit) through a **Makefile** and deciding the necessary tools to implement it!

ANALYTICS, DATA CLEANING AND TRAINING

Python: for the development of the entire Pipeline.

- Matplotlib & Seaborn: For plotting of data;
- Numpy: For numerical handling of data;
- Pandas: For data manipulation;
- Plotly: Tables;
- Scikit-learn: Training algorithms and functions.

DATABASE

SQL Database with 10 different tables, corresponding to the 10 different .csv files. This allowed us to create queries that were used during all the data understanding and cleaning processes. Ex:

```
df = db.df_query('SELECT * FROM loan_train JOIN
...)
```

DATA VISUALIZATION

Open Refine: for better visualization of the dataset;
MySQL Workbench: to test some of the more complex queries created during the project development.

DIMENSIONS OF DATA QUALITY

ACCURACY

How well does the information reflect reality?

- The data is relatively old, referring to the years 1993 to 1996, therefore the models used for predictions may be outdated. To improve accuracy, new data should be collected, such as updated criminality in districts, recent transactions, new clients, etc...

COMPLETENESS

Data fulfills expectations of comprehensiveness?

- There are some missing values, for example in the districts table, which we fulfilled to increase completeness.

CONSISTENCY

Information stored in one place matches relevant data stored elsewhere?

- The database is up to date according to initial data.

TIMELINESS

Is information available when we need it?

- MySQL database ensures that data is ready as soon as we need it to train the models, having no limitations of availability.

VALIDITY

Is information in a specific format?

- A strict format was defined for each type of data according to the business rules. For example, all dates are in the format YYYY-MM-DD and all invalid/missing values are set to NaN.

UNIQUENESS

Is there only one instance of the data appearing in the database?

- No duplicated data was found during the dataset exploration, so there isn't redundant data.

FEATURE IMPORTANCE - Clients & Accounts

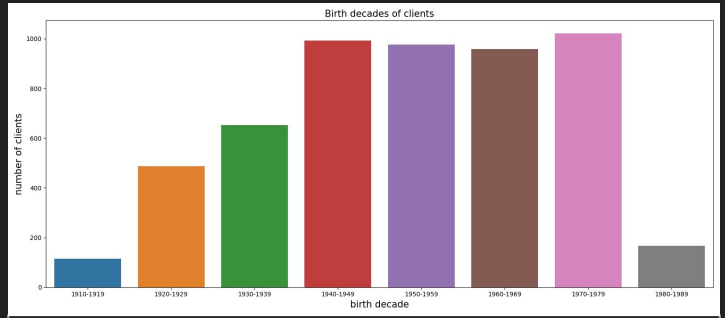
DATA CLEANING

- Split birth_number into:
  - **birth_date** (timestamp with the birth_date of the client in the format YYYY-MM-DD);
  - **gender** (binary column with the gender of the client, where 0 means Female and 1 means Male);
- Format the account creation date to a correct timestamp in the format YYYY-MM-DD;
- Encode the categorical feature "frequency" into numerical values.

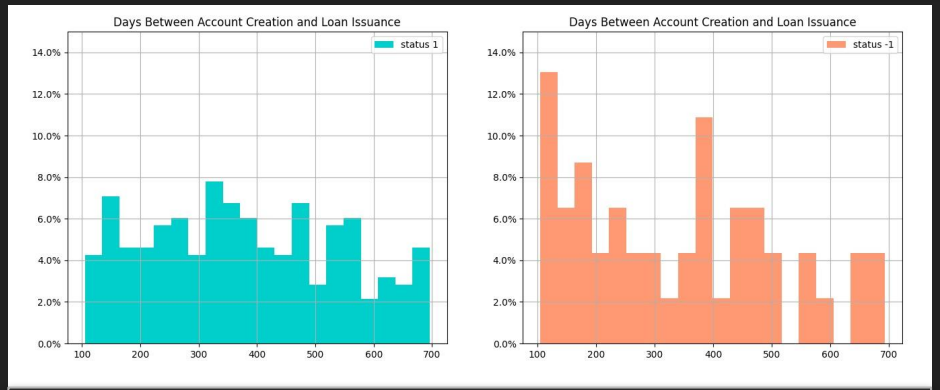
FEATURE ENGINEERING

- **same_district**: Boolean value set to 1 if the account was created on the owner district, 0 otherwise;
  - Fraudulent clients may be tempted to create the account in a different district for anonymity purposes and so that they become harder to track;
- **days_between**: Number of days between account creation and loan issuance;
  - To keep their regular accounts secure and not risk some possible consequences, users might create new accounts just for the issuance of fraudulent loans, therefore accounts with high activity in their initial phase may be suspicious;
- **age_when_loan**: Age of the owner of the account when a loan was issued;
  - The age of the clients at the moment of the loan may help determine if the loan will end successfully.

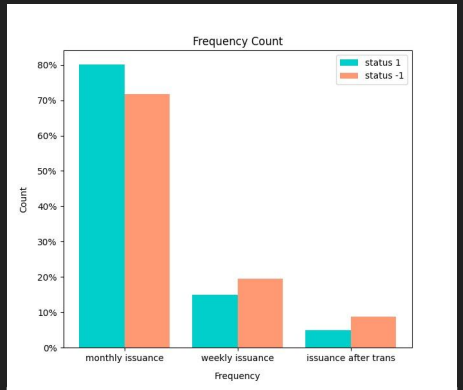
FEATURE IMPORTANCE – Clients & Accounts



Birth decades of the clients: there are not many clients from early decades nor recent ones



Days between account creation and loan issuance for each loan status
Fraud tends to occur on recently created accounts



Frequency for each status: the distribution is similar for both status

FEATURE IMPORTANCE - Cards & Disponents

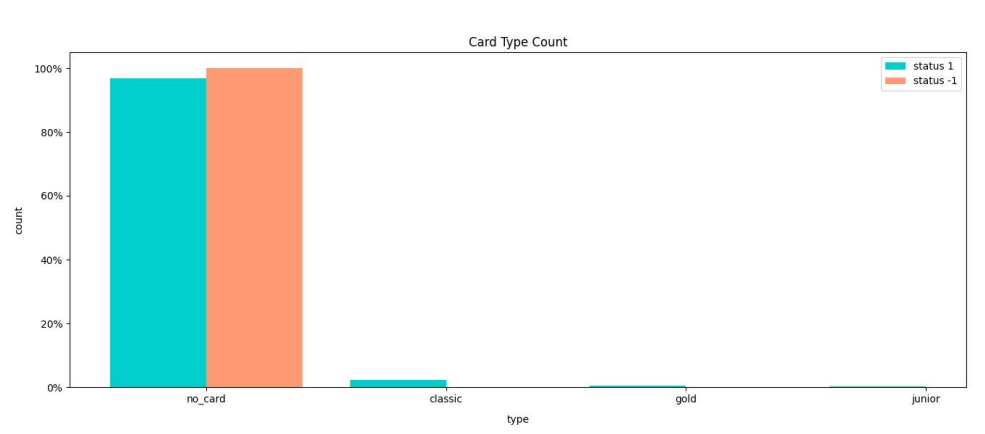
DATA CLEANING

- Created feature num_cards, containing the number of cards of a given account, to be used in the creation of the feature **has_card**, explained below;
- Created feature n_disponents, containing the number of disponents of a given account, to be used in the creation of the feature **has_disponent**, explained below;
- Encoded both of these new features to numerical values instead of boolean values.

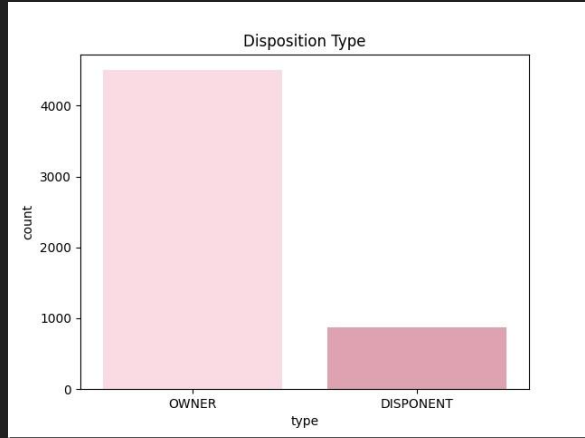
FEATURE ENGINEERING

- **has_card**: Boolean value set to 1 if the account associated with the loan has at least one card, 0 otherwise;
    - card_type has been proven insignificant since only loans that ended successfully have cards;
- **has_disponent**: Boolean value set to 1 if the account associated with the loan has at least one disponent associated, 0 otherwise
    - Accounts that are used by more than one client, i.e., have at least one disponent, might help decide whether a loan will be successful or not since only a few accounts fall into this type.

FEATURE IMPORTANCE – Cards & Disponents



Card type distribution for each status: we can see that only loans that ended successfully have cards, even though most of the accounts don't have a card



Disposition Types: There are accounts used by more than one client (have a disponent). To reflect, this we created a has_disponent feature which will prove to be useful

FEATURE IMPORTANCE - Districts

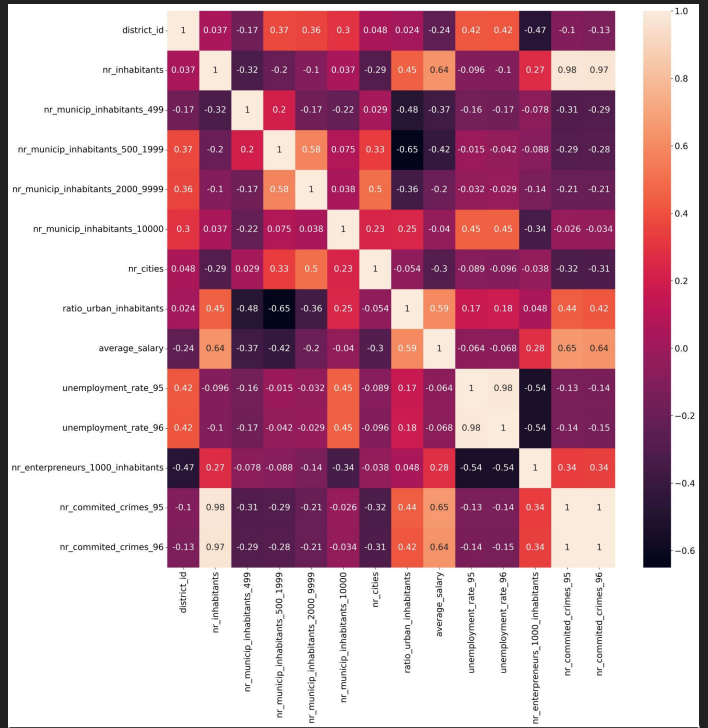
DATA CLEANING

- Replaced missing values of "nr_commited_crimes_95" and "unemployment_rate_95" by the values of the year of 96, due to the similarity found in the slide 8;
- Removed redundant columns already used in the feature engineering process;
- Encode the categorical feature "region" into numerical values.

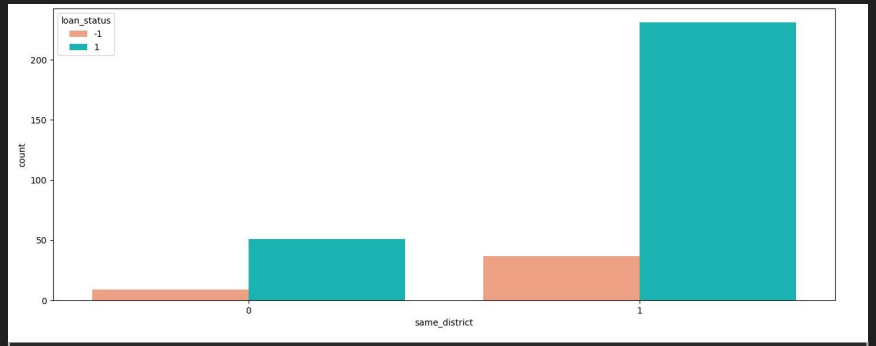
FEATURE ENGINEERING

- **ratio_entrepeneurs**: The ratio of entrepreneurs (number of entrepreneurs per 1000 inhabitants);
- **ratio_urban_inhabitants**: The ratio of urban inhabitants of the district;
- **avg_crimes**: The average of crimes committed in the given district (both 95 and 96 years) per inhabitant;
    - The loans from an account created on a district with high criminality rate may be suspicious;
- **avg_unemployment**: The average of unemployment rate of both 95 and 96 years;
    - Districts with high unemployment rate might result in clients who may not be able to pay the loan due to economic problems, therefore leading to its failure;
- **criminality_growth**: The evolution of criminality per inhabitant in the district from the year of 95 to 96;
- **unemployment_growth**: The evolution of unemployment in the district from the year of 95 to 96.

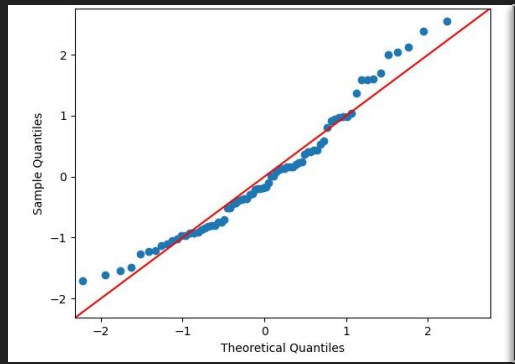
FEATURE IMPORTANCE – Districts



Correlation matrix that shows the relationship between features related to the districts



Distribution of same_district: most of the accounts were created in the same district of the client



Unemployment rate ´95: this column doesn't follow a normal distribution, so the missing values should not be filled with the mean

FEATURE IMPORTANCE - Transactions

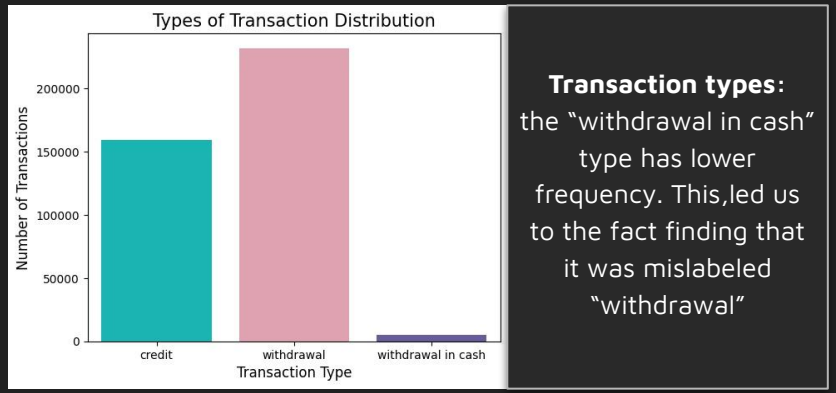
DATA CLEANING

- Deleted "bank" column since more than 50% of the transactions have no bank associated;
- Replaced the empty string values by NaNs, to facilitate the location of missing values;
- Replaced the missing values from "operation" column by the value "interest credited", because the "k_symbol" column has this value for all the transactions with a missing operation;
- Replaced the transaction type "withdrawal in cash" by "withdrawal", because this last has an insignificant frequency compared to the other two types;
- Made the amount of the transactions of type "withdrawal" negative, which makes sense in our context.

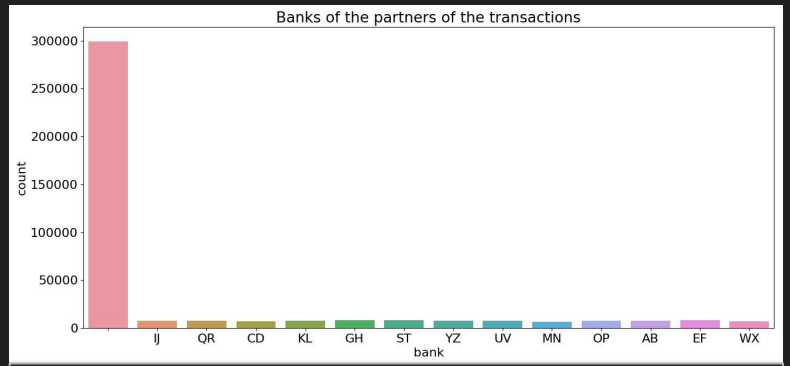
FEATURE ENGINEERING

- **avg_amount_credit**, **avg_amount_withdrawal**, **avg_amount_total**: for each account, calculated the average amount of the transactions of type "credit", "withdrawal" and both of these, respectively;
- **min_amount**, **max_amount**: minimum and maximum amount of the transactions of an account;
- **credit_ratio**: the ratio of transactions with type credit out of all the transactions of an account;
- **min_balance**, **avg_balance**, **std_balance**, **negative_balance**, **last_balance**, **last_balance_negative**: minimum, average and standard deviation of the transactions of an account, their last balance, and two booleans telling if that account ever had a negative balance or if its last balance was negative;
- **num_trans**: total number of transactions of a given account.

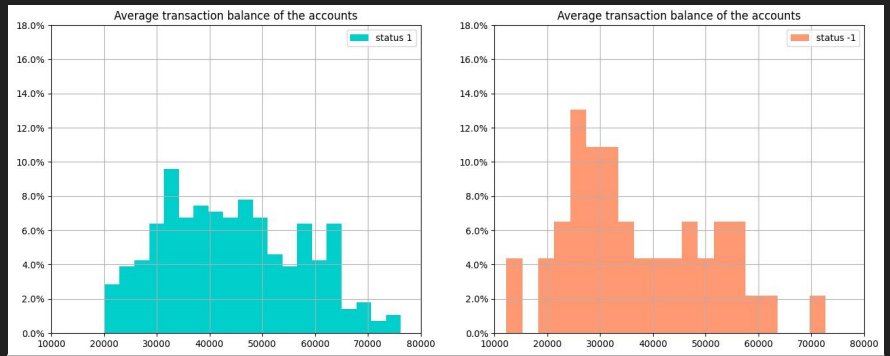
FEATURE IMPORTANCE – Transactions



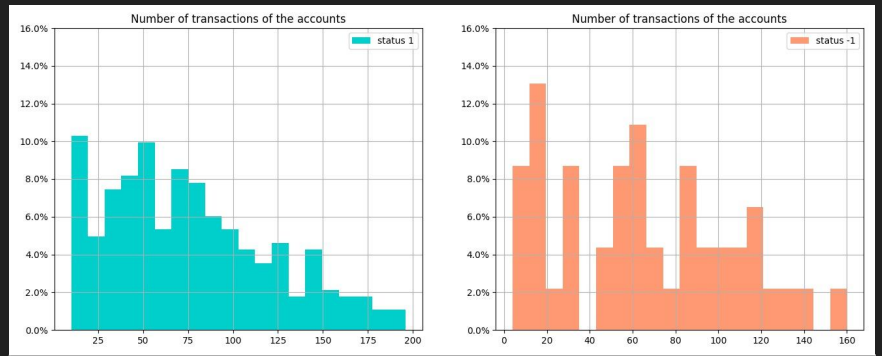
Transaction types: the "withdrawal in cash" type has lower frequency. This,led us to the fact finding that it was mislabeled "withdrawal"



Banks: Most of the transactions don't have a partner bank



Average balance: average transaction balance distribution for accounts with successful and default loans

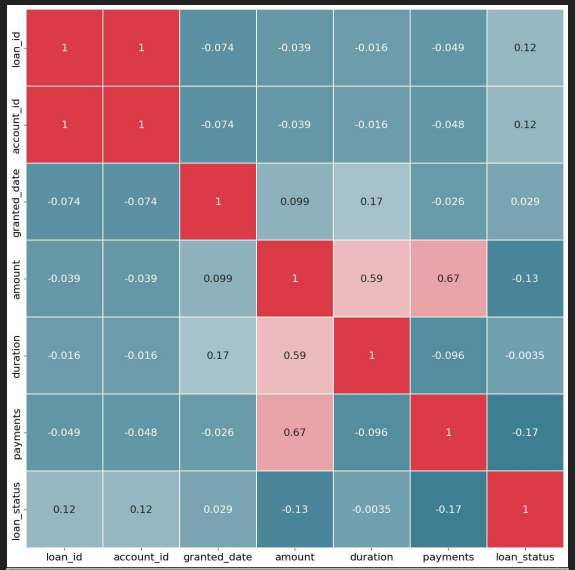


Number of transactions: total number of transactions distribution by accounts with successful and default loans

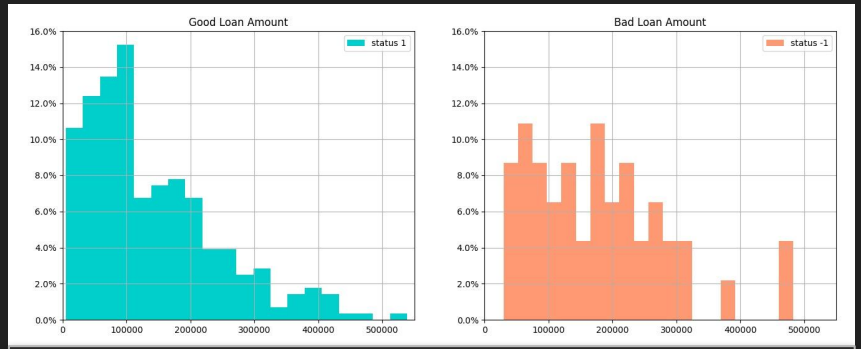
FEATURE IMPORTANCE - Loans

DATA CLEANING

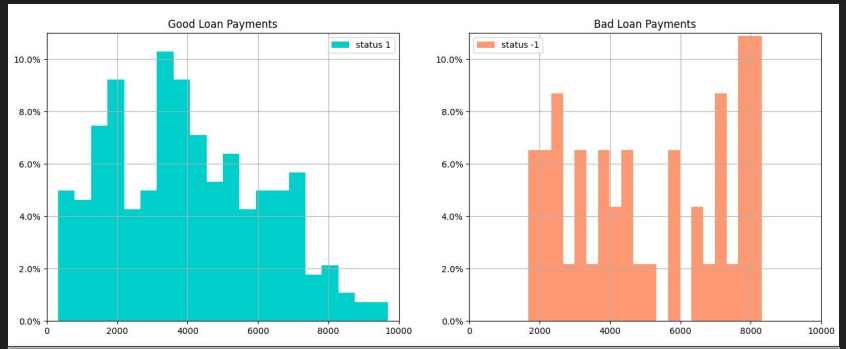
- Format the loan issuance date to a correct timestamp in the format YYYY-MM-DD



Correlation matrix: relationship between features related to the loans



Loan Amount: distribution of the amounts of money of the loans by the accounts with successful and default loans



Monthly Payments: distribution of payments by the accounts with successful and default loans

PREDICTIVE ALGORITHMS

DECISION TREE

- Decision Trees are easy to interpret (**White-Box**), don't require scaling of data and perform automatic feature selection.
- They are **prone to overfitting** - too many branches may reflect anomalies due to noise or outliers. To solve this, mechanisms like pruning or defining the maximum depth may be used. An example of a Decision Tree Model that may be overfitting may be seen in the last image, where the train AUC is much higher than the test AUC.
- Also, they don't handle correlated features well and a small change in the data can cause a large change in the structure of the decision tree.

```
{'criterion': ['gini', 'entropy'],
 'splitter': ['best', 'random'],
 'max_depth': [3,5,7,10, 12],
 'min_samples_split': [1,2,3],
 'min_samples_leaf': [1,2,3],
 'min_weight_fraction_leaf': [0.0],
 'max_features': [None, 'auto', 'sqrt', 'log2', 10, 12, 15],
 'max_leaf_nodes': [None],
 'min_impurity_decrease': [0.0],
 'class_weight': [None],
 'ccp_alpha': [0.0]}
```

Grid Search Parameters for Decision Tree

```
Best Parameters:  {'ccp_alpha': 0.0, 'class_weight': None, 'criterion':
'entropy', 'max_depth': 7, 'max_features': 10, 'max_leaf_nodes': None, '
min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split':
 3, 'min_weight_fraction_leaf': 0.0, 'splitter': 'best'}
Best Score:  0.9359635825006617
```

Example of GridSearch output for Decision Tree

```
Confusion matrix of split 1:
[[81 18]
 [17 72]]
Confusion matrix of split 2:
[[78  6]
 [16 88]]
Confusion matrix of split 3:
[[85 14]
 [19 70]]
AUC Train scores of each fold - [0.994266542087828, 0.9845931222335716,
 0.9900053795407571]
AUC Test scores of each fold - [0.8409942117807285, 0.9428228021978022,
 0.8912155260469867]
Average Train AUC =  0.99
Average Test AUC =  0.891678
```

Example of local results for Decision Tree

PREDICTIVE ALGORITHMS

RANDOM FOREST

- Random Forest is an **ensemble** learning method that averages multiple decision trees, trained on different parts of the same training set, with the goal of reducing the variance.
- Each tree is built over a subset of random observations and features. As trees are de-correlated, Random Forest is **less prone to overfitting** than a Decision Tree.
- It is **not sensible to outliers**, as the final outcome is taken by consulting many decision trees.
- The results are **harder to visualize**, in contrast with Decision Trees.

```
{'n_estimators': [int(x) for x in range(2, 14, 2)],
 'max_features': ['auto', 'sqrt'],
 'max_depth': max_depth,
 'criterion': ['gini', 'entropy'],
 'min_samples_split':  [2, 4, 6, 8],
 'min_samples_leaf':  [1, 2, 4, 6],
 'class_weight': ["balanced", "balanced_subsample", None]}
```

Grid Search Parameters for Random Forest

```
Best Parameters:  {'class_weight': 'balanced', 'criterion': 'entropy
', 'max_depth': 14, 'max_features': 'auto', 'min_samples_leaf': 1, '
min_samples_split': 2, 'n_estimators': 10}
Best Score:  0.980821946393164
```

Example of GridSearch output for Random Forest

```
Confusion matrix of split 1:
[[86 10]
 [ 6 86]]
Confusion matrix of split 2:
[[89 11]
 [ 6 82]]
Confusion matrix of split 3:
[[81  5]
 [ 3 99]]
AUC Train scores of each fold - [1.0, 1.0, 1.0]
AUC Test scores of each fold - [0.9810914855072463, 0.9803409090909091, 0.9903100775193798]
Average Train AUC =  1.0
Average Test AUC =  0.983914
```

Example of local results for Random Forest

PREDICTIVE ALGORITHMS

XGBCLASSIFIER

- XGBClassifier is provided by **XGBoost**, an optimized Gradient Boosting library designed to be efficient. It is an **ensemble** learning method that uses **Decision Trees as base learners**.
- As a **Gradient Boosting** algorithm, it works by building weak prediction models sequentially, where each model tries to minimize the error of the previous one.
- Boosting algorithms are usually slow to learn, but also highly accurate. However, XGBoost is faster than most of the ensembles. It also handles missing values and does not require scaling.
- However, it is prone to overfit given that it continually tries to minimize errors. Therefore, it requires properly **tuned parameters** - **GridSearch** was used for this purpose.
- Finally, as it is a **Black-Box** model, it is difficult to interpret.

```
{'min_child_weight':range(1,6,2),
 'gamma': [0,0.2,0.5,1,1.5,3],
 'max_depth':[5,8,10,15],
 'reg_alpha':[1e-5, 1e-2, 0.1, 1]}
```

Grid Search Parameters for XGBoost

```
Best Parameters:  {'gamma': 0.2, 'max_depth': 5, 'min_child_weight': 1, 'reg_alpha': 1e-05}
Best Score:  0.9728068382781468
```

Example of GridSearch output for XGBoost

```
Confusion matrix of split 1:
[[79 13]
 [ 2 94]]
Confusion matrix of split 2:
[[87  6]
 [ 5 90]]
Confusion matrix of split 3:
[[89  8]
 [15 76]]
AUC Train scores of each fold - [1.0, 1.0, 0.9999716994481394]
AUC Test scores of each fold - [0.9655797101449275, 0.9857385398981324,
 0.9423926588875042]
Average Train AUC =  1.0
Average Test AUC =  0.96457
```

Example of local results for XGBoost

PREDICTIVE ALGORITHMS

LOGISTIC REGRESSION

- Logistic regression is used when the dependent variable is categorical and works better on binary classification problems, which is the case.
- It learns a linear relationship from the given dataset and then introduces a non-linearity in the form of the Sigmoid function.
- It is simple to implement so it is commonly used as a benchmark model to measure performance. Therefore, we used it in our first submission with the loan table.
- However, it has poor performance on non-linear data, and linearly separable data is rarely found in real scenarios. It is also not an adequate option in the presence of irrelevant or highly correlated features.

```
{'penalty': ['l2', 'none'],
 'C': [0.01, 0.05, 0.1, 0.2, 0.5, 1.0],
 'solver': ['newton-cg', 'lbfgs', 'sag', 'saga'],
 'class_weight': ["balanced", None]}
```

Grid Search Parameters for Logistic Regression

```
Best Parameters:  {'C': 0.01, 'class_weight': None,
'penalty': 'none', 'solver': 'newton-cg'}
Best Score:  0.9306834379588479
```

Example of GridSearch output for Logistic Regression

```
Confusion matrix of split 1:
[[86 14]
 [15 73]]
Confusion matrix of split 2:
[[77 14]
 [13 84]]
Confusion matrix of split 3:
[[61 30]
 [15 82]]
AUC Train scores of each fold - [0.9223405460518863, 0.924239422668742, 0.9298712324890336]
AUC Test scores of each fold - [0.9285227272727272, 0.9276084740002266, 0.8921490880253767]
Average Train AUC =  0.925
Average Test AUC =  0.916093
```

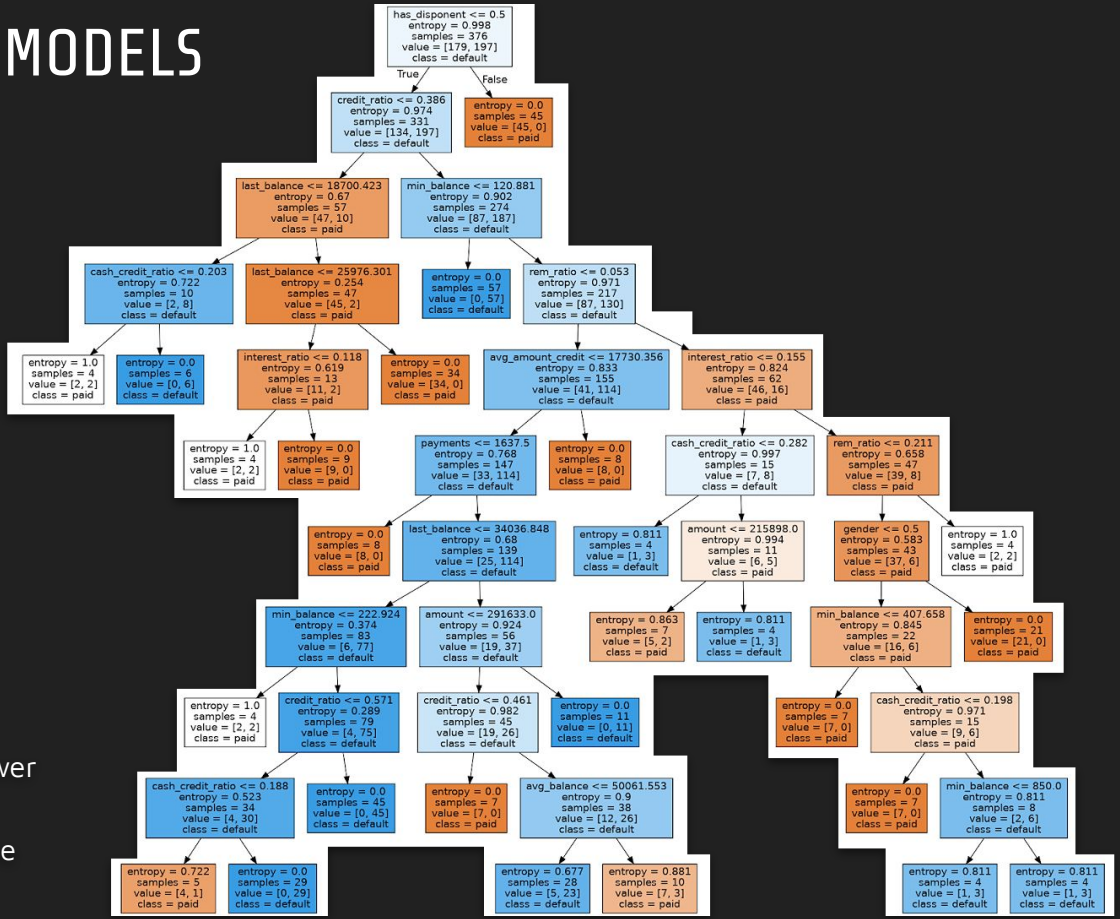
Example of local results for Logistic Regression

ANALYSIS OF WHITE-BOX MODELS

In White-Box models it's clear how they behave, how they produce predictions and what the influencing variables are.

DECISION TREE

- This Decision Tree Model is the one that provides the best local scores among the Decision Trees, with an AUC of 0.91. It uses Filter Feature Selection, SMOTE and 3 split cross-validation.
- Even though it is quite deep to interpret, we can see that the **main splitting factor is the has_disponent** feature, which may indicate that accounts with disponent pay the loan.
- Also, we can see that clients with more than 39% of credits and that reached a balance lower than 121 normally default.
- The decision features are mostly related to the **transactions and loan features**.



DESCRIPTIVE ALGORITHMS

DBSCAN

Handles clusters with different shapes/sizes;
Identifies outliers as noise;
Does not require a predetermined number of clusters;
Requires **radius (ε)** and **min. number of points**.

1. Classify each observation as: core, border or noise point;
2. Eliminate the noise points;
3. Points within a distance are allocated to the same group;
4. Border points to the group of the nearest core point.

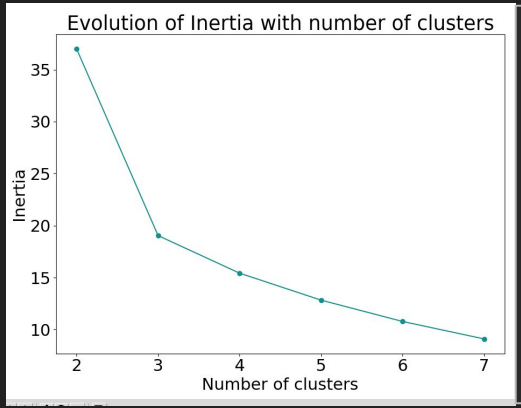
Silhouette Coefficient was used for parameter tuning and to evaluate the performance of the algorithm.
Silhouette values close to 1 ⇒ clusters well distinguished.

no_clusters	silhouette_score	epsilon	min_points
3	0.495001	0.2	2
3	0.495001	0.2	3
3	0.495001	0.2	4
2	0.437134	0.3	2
2	0.437134	0.3	3
2	0.437134	0.3	4
2	0.527313	0.4	2

K-MEANS

Requires **number of clusters**;
Fast and scalable;
Squared Euclidean distance as criterion.

Randomly selects the centers of k groups and iteratively tries to optimize the positions of the centroids (allocates each observation to the group whose center is nearest and re-calculates the centers) until they stabilize.



Elbow Method was used to select the optimal number of clusters. The Inertia plot on the left was used to finds the elbow point, the ideal number of clusters, for the features *average_salary*, *avg_balance* and *avg_amount_credit*.

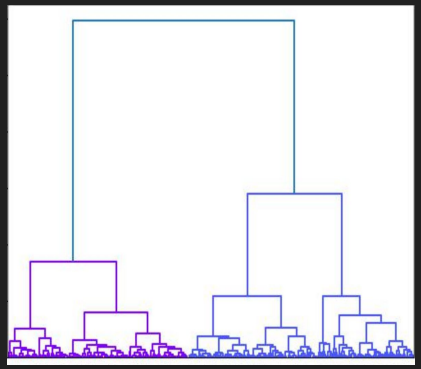
DESCRIPTIVE ALGORITHMS

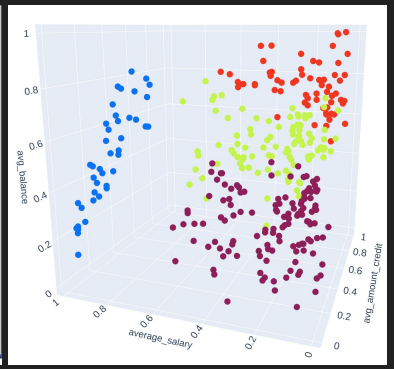
AGGLOMERATIVE

A bottom-up hierarchical method that starts with as many groups as there are cases and, on each upper level, merges the most similar pair of groups into a single group.

Different proximity measures were tested (ward, average, complete or single) and result in different clusters.

The **Dendogram** shows 2 clusters or 4 smaller clusters for the *average_salary*, *avg_balance* and *avg_amount_credit*.
Using 4 clusters and **ward linkage** results in a group with high salary and 3 groups with average salary and increasing amount of credits and average balance.





PAM - K-MEDOIDS

While K-Means minimizes the within cluster sum of squares, **K-Medoids** minimizes the sum of distances between each point and the medoid of its cluster. It is more robust to noise and outliers than K-Means, as it uses data points as centroids instead of averages.

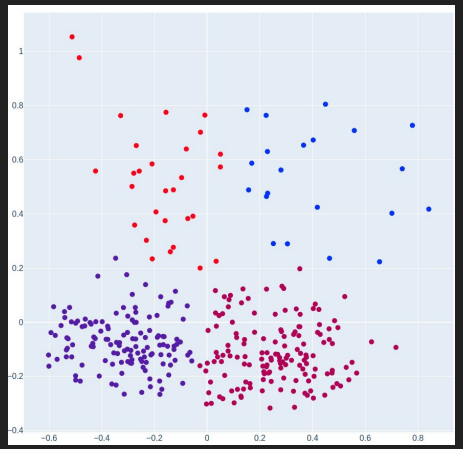
PAM greedily selects the medoids in the build phase. Then, in the swap phase, the medoids are swapped with other points in order to decrease the average dissimilarity coefficient.

K-Medoids vs K-Means: the example shows information about the clusters that result from applying each algorithm on 2 PCA components, generated from economic features (min_balance, credit_ratio, avg_balance, std_balance,avg_amount_withdrawal, avg_amount_total). We can see that, in this case, the **Inertia** value is lower and the **Silhouette Score** is higher when using K-Means which makes it a better choice.

```
KMedoids Clusters:
 Counter({1: 117, 3: 99, 0: 69, 2: 43})
KMedoids Centers:
 [[ 0.03467948 -0.11177621]
 [-0.33222752 -0.07561027]
 [ 0.05039535  0.57327688]
 [ 0.35451565 -0.11528226]]
KMedoids Inertia:
55.463234082224176
KMedoids Silhouette Score:
0.40691785787771895
```

```
KMeans Clusters:
 Counter({1: 146, 0: 132, 3: 29, 2: 21})
KMeans Centers:
 [[-0.31356204 -0.07322453]
 [ 0.25918307 -0.11439253]
 [ 0.40945594  0.53204591]
 [-0.17410734  0.52393045]]
KMeans Inertia:
12.903928856099585
KMeans Silhouette Score:
0.5355056692432675
```

DESCRIPTIVE ALGORITHMS – RESULTS



Features:

- **PCA** was used for dimensionality reduction.
- min_balance, avg_balance, avg_amount_withdrawal, std_balance, credit_ratio, avg_amount_total.

Performance:

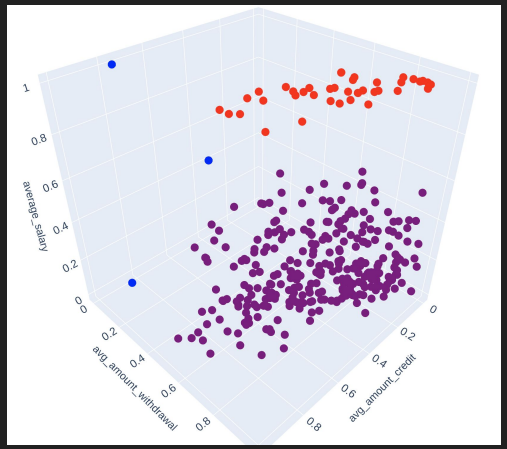
Silhouette Score ≈ 0.5355

Algorithm: K-Means

Clusters: 4

This clusters represent economic groups of clients that requested loans. The average values for each feature are shown below for the 4 clusters.

Mean Feature Values	C0	C1	C2	C3
avg_balance	54021.18	32940.28	31600.28	49926.26
min_balance	656.23	592.36	662.07	642.86
avg_amount_withdrawal	-10226.89	-4691.50	-7914.88	-18457.37
std_balance	24223.97	11437.49	12786.15	23874.53
avg_amount_total	735.69	448.72	2802.85	3584.87
credit_ratio	0.4195	0.4135	0.7442	0.6801



Features:

average_salary, avg_amount_credit, avg_amount_withdrawal.

Performance:

Silhouette Score ≈ 0.4950

Algorithm: Agglomerative, with average linkage

Clusters: 3

This clusters represent simpler groups of clients that requested loans. It uses both economic(average amount of credits and withdrawals) and demographic information (average salary in the district of the client).

We can distinguish 2 dense clusters, one representing clients with high salary and other with low/average salary, both performing withdrawals of high/medium amounts. The other cluster represents clients that have a small average amount of withdrawals. If we use DBSCAN, the points of this last cluster are identified as "noise".

CONCLUSIONS AND FUTURE WORK

CONCLUSIONS

- By progressively adding new tables and features, we were able to iteratively experiment with different data preparation strategies and greatly improve our understanding of the problem, as well as the results;
- The features created in the Feature Engineering phase allowed us to improve the Kaggle score a lot;
- Tree-based algorithms achieve the best results, due to the low quantity of information about the loans;
- The **oversampling** with SMOTE also contributed to our best scores;
- Given that there's a low amount of data about the loans, we didn't find it advantageous to use **sampling** methods, or else we could lose important information;
- Likewise, we didn't use many splits during the **cross-validation** method, due to the unbalanced classes, which could impair the positive class.

FUTURE WORK

- Creation of more features, there is always room for creativity and infinite possibilities;
- Given that the train data has observations from 1993 to 1996 and that the test data has observations from 1997 to 1998, we could use the older observations of the train data for train and the more recent for test in order to simulate a real environment;
- Use the conclusions deduced about the clients during the descriptive approach in the predictive problem.

GROUP EVALUATION

- Diana Freitas, up201806230 - 0.33
- Mariana Ramos, up201806869 - 0.33
- Paulo Ribeiro, up201806505 - 0.33