

# Loaning 101

To loan or not to loan? That is the question

A Data Mining Project for Computational Learning



**Beatriz Aguiar**

up201906230

**João Marinho**

up201905952

**Margarida Vieira**

up201907907

# Agenda

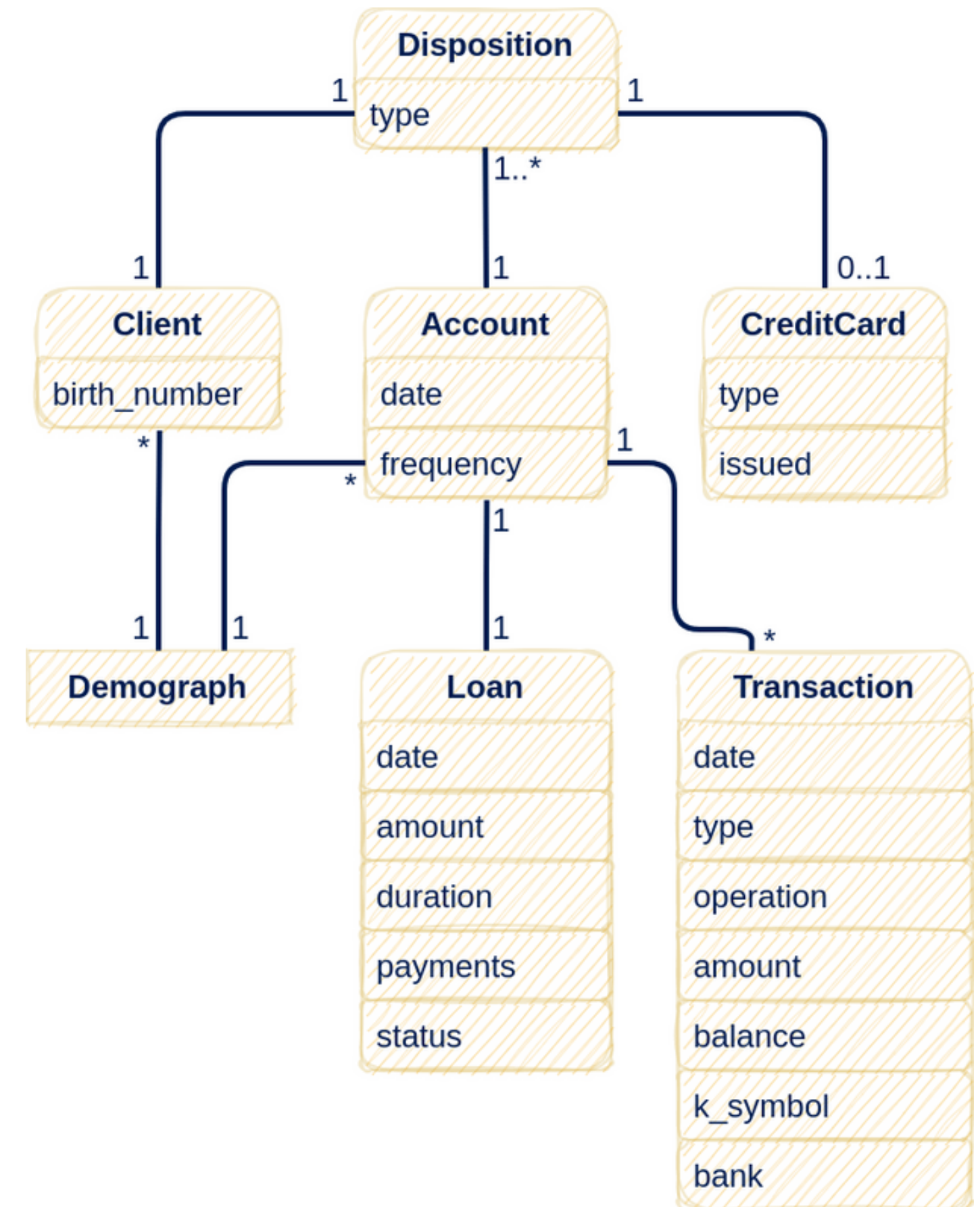
- 1** Domain Description
- 2** Exploratory Data Analysis
- 3** Predictive Data Mining Problem
  - 3.1** Problem Definition
  - 3.2** Data Preparation
  - 3.3** Experimental Setup
  - 3.4** Results
- 5** Descriptive Data Mining Problem
- 7** Conclusions

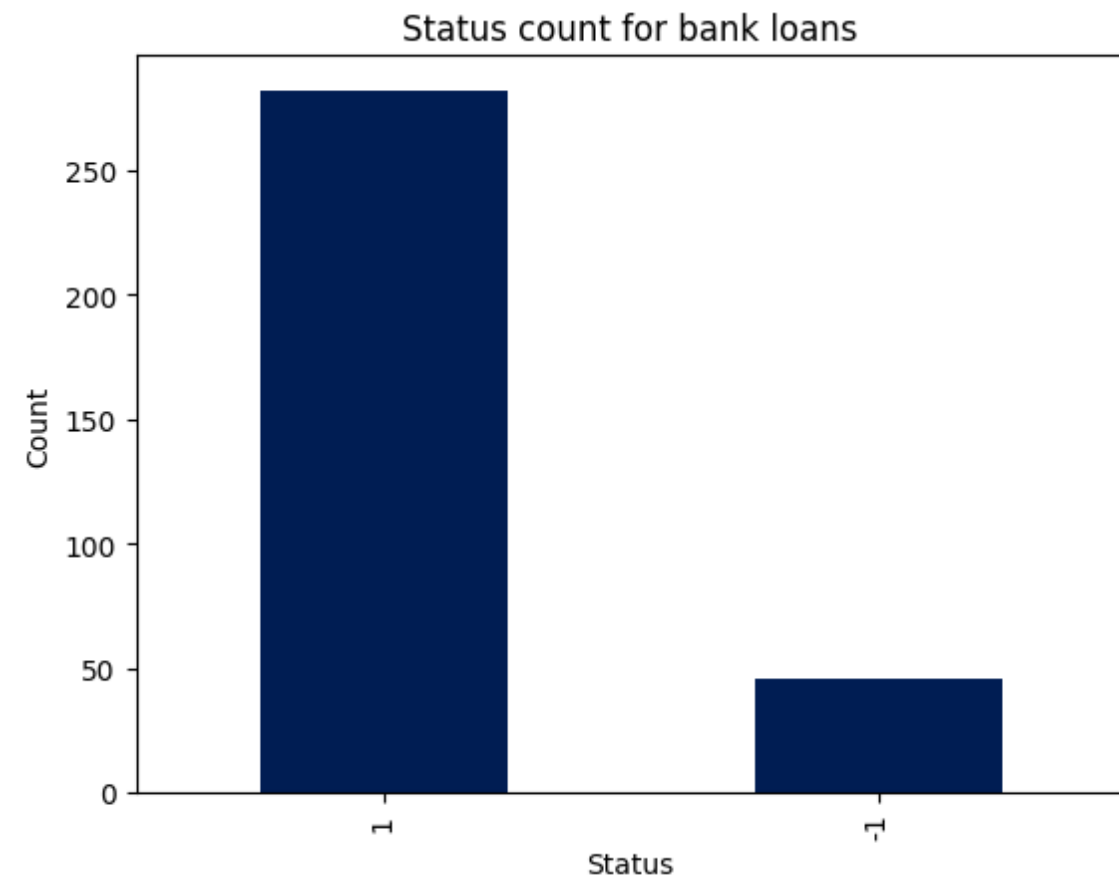


# Domain Description

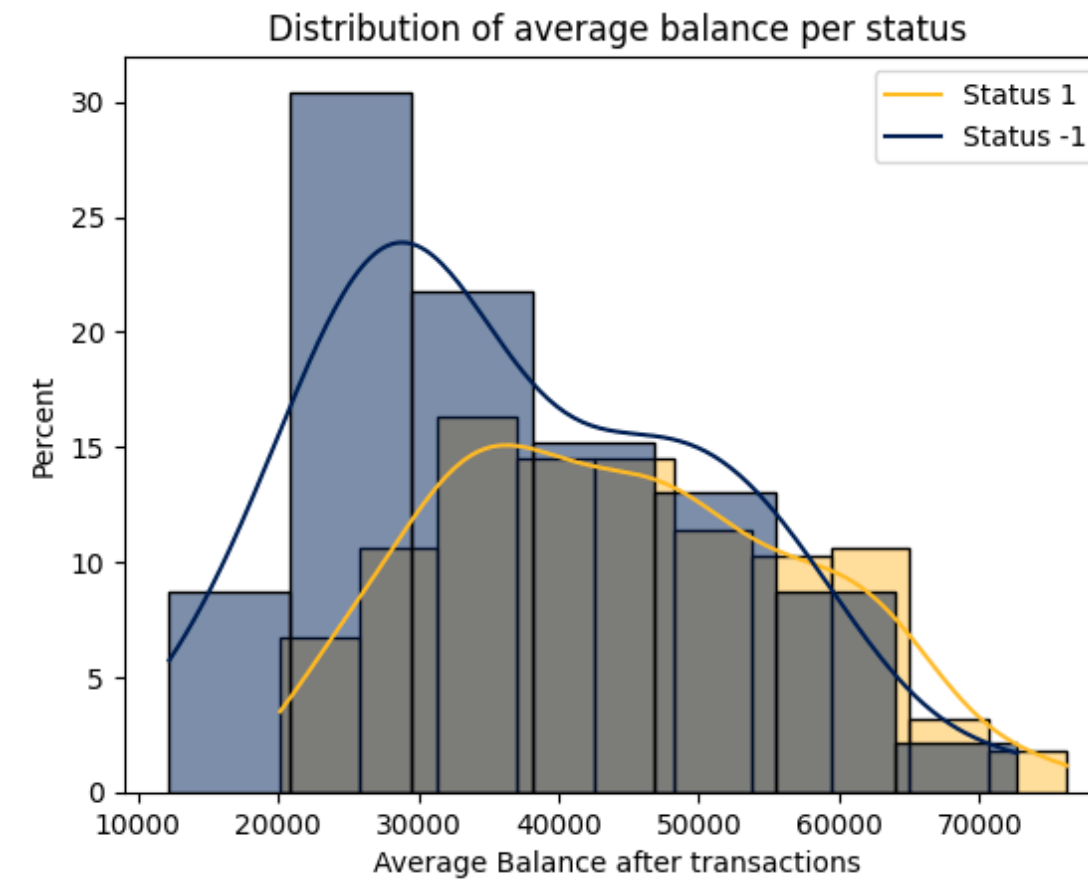
The various datasets comprise information on a bank's clients, their accounts (including transactions made within several months), the granted loans, the credit cards issued, as well as some demographic information.

- 4500 accounts
- 5369 clients
- 328 loans
- 396685 transactions
- 177 credit cards
- 77 districts



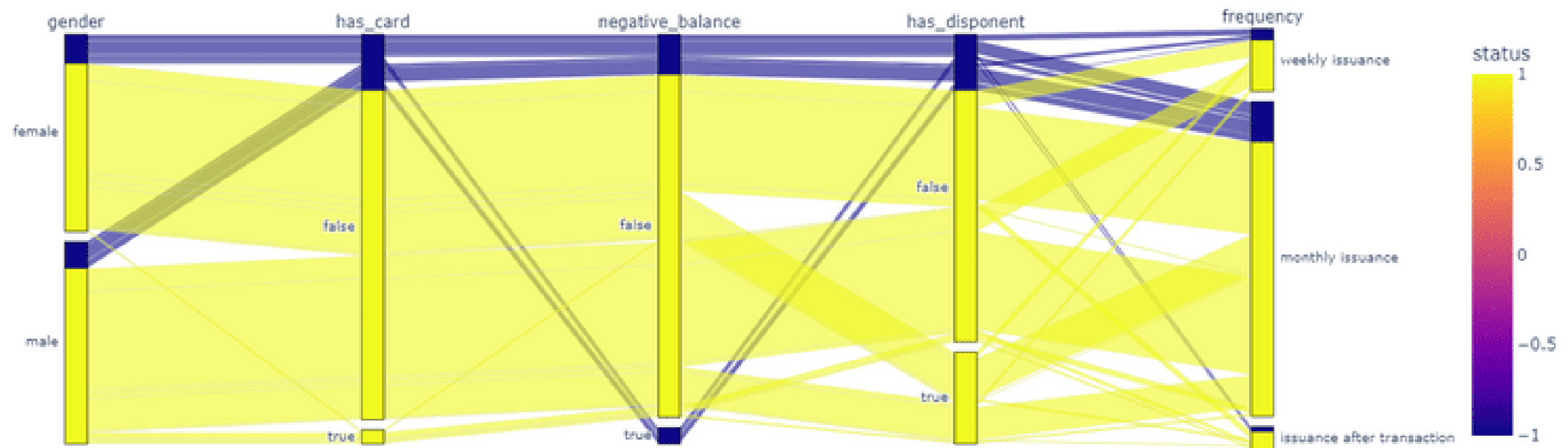


- status class is imbalanced



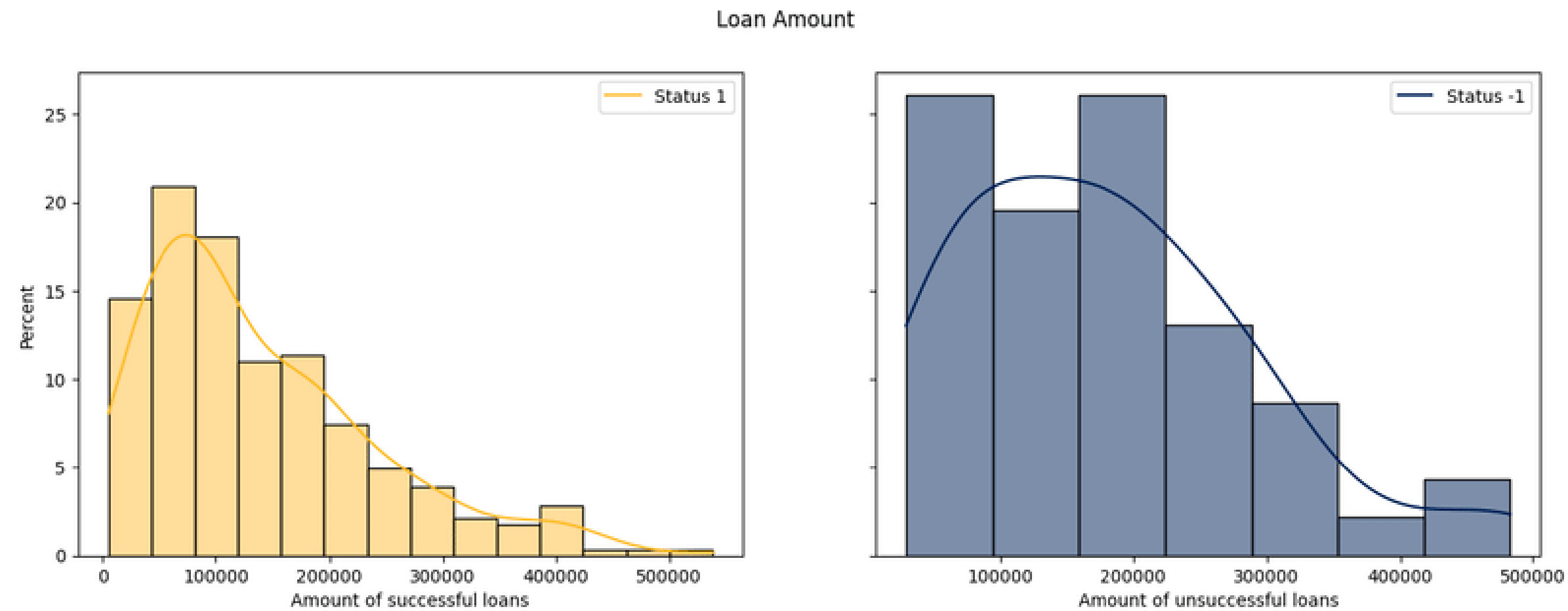
- loans associated with accounts with a lower average balance after transactions are more likely to be unsuccessful

# Exploratory Data Analysis



- accounts with cards are mainly owned by males
- loans associated with accounts that have had a negative balance tend to be fraudulent
- accounts that have had a negative balance do not have a disponent nor cards associated
- loans associated with accounts with disponents tend to be successful

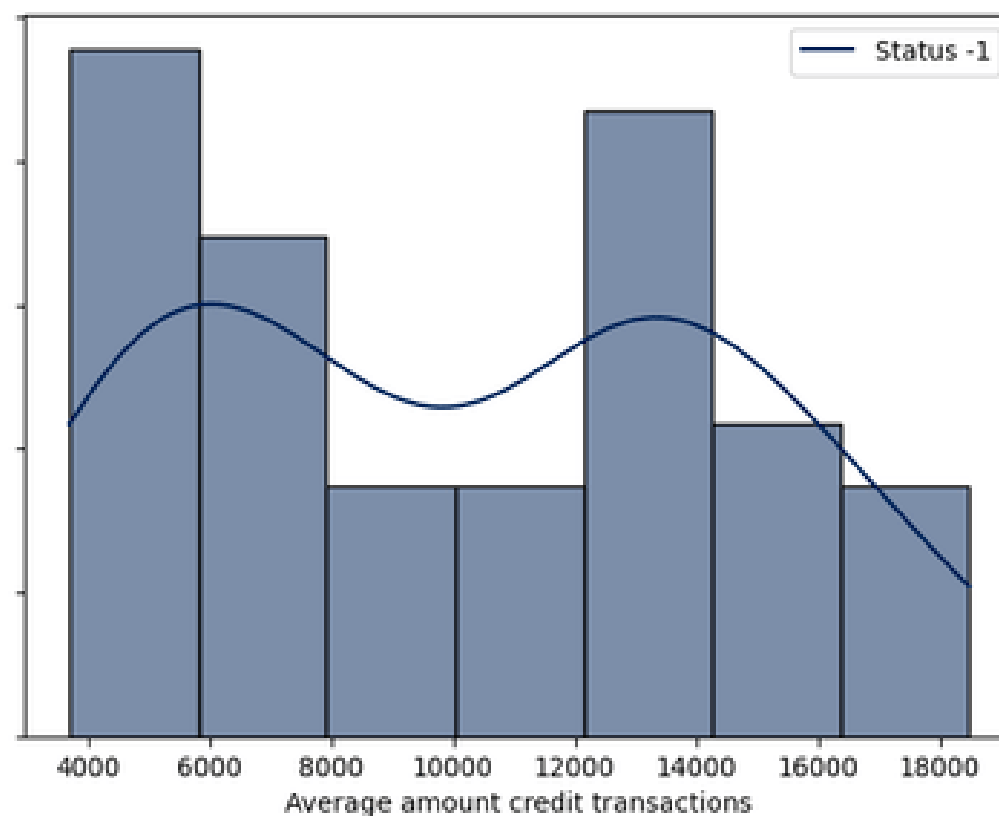
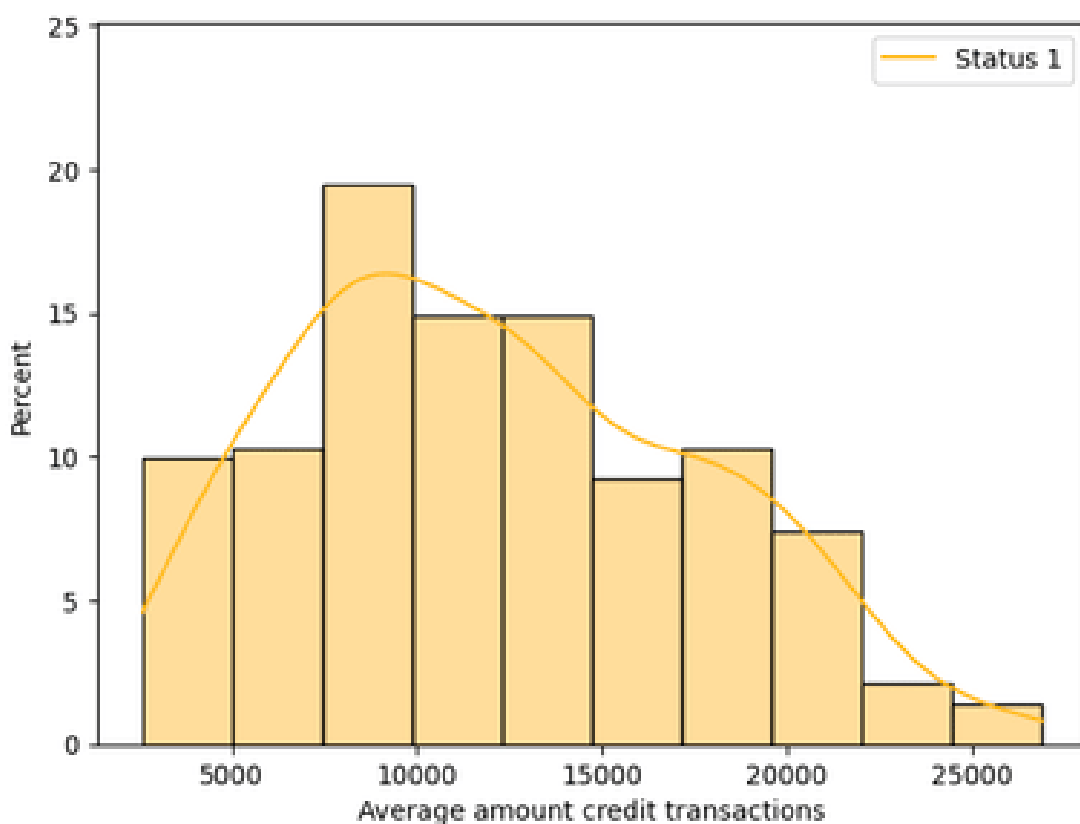
# Exploratory Data Analysis



- amount follows a F-distribution as expected in the loan market
- loans with an amount higher than 500000 tend to be successful

# Exploratory Data Analysis

Average amount of credit transactions

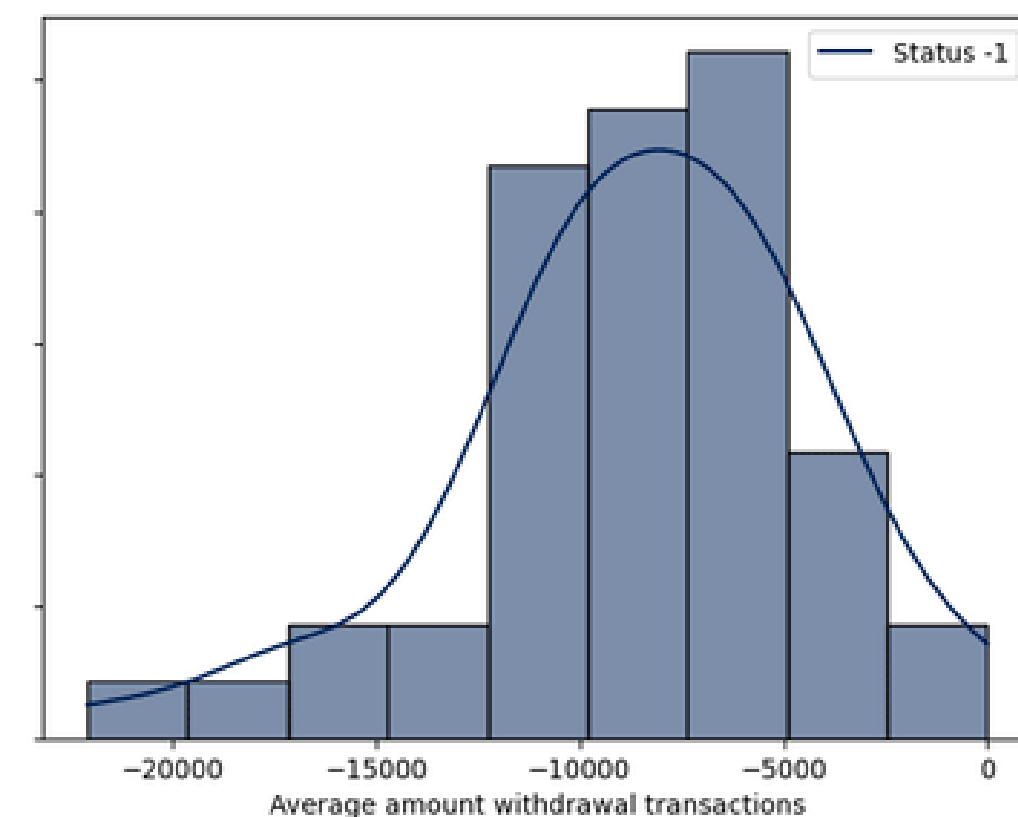
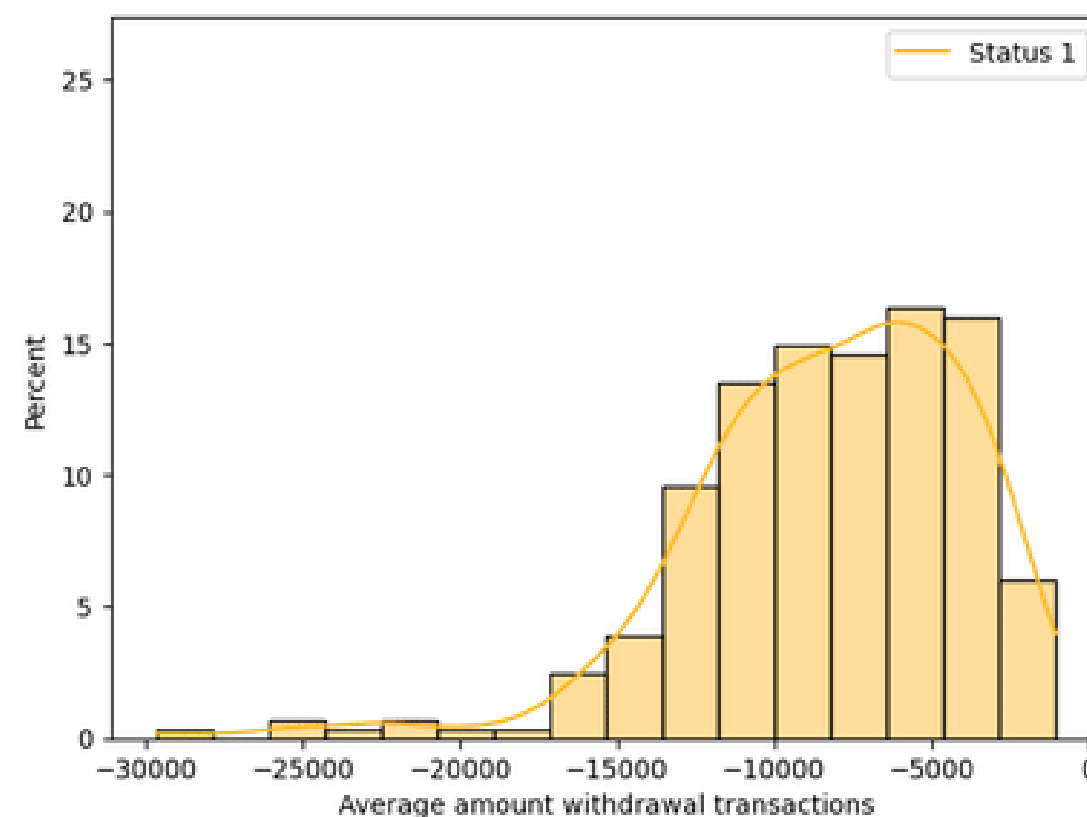


**loans associated with accounts where the credits' average amount is**

- higher tend to succeed, as there is no record of unsuccessful loans where this value is higher than 18000
- around 10000 tend to be successful

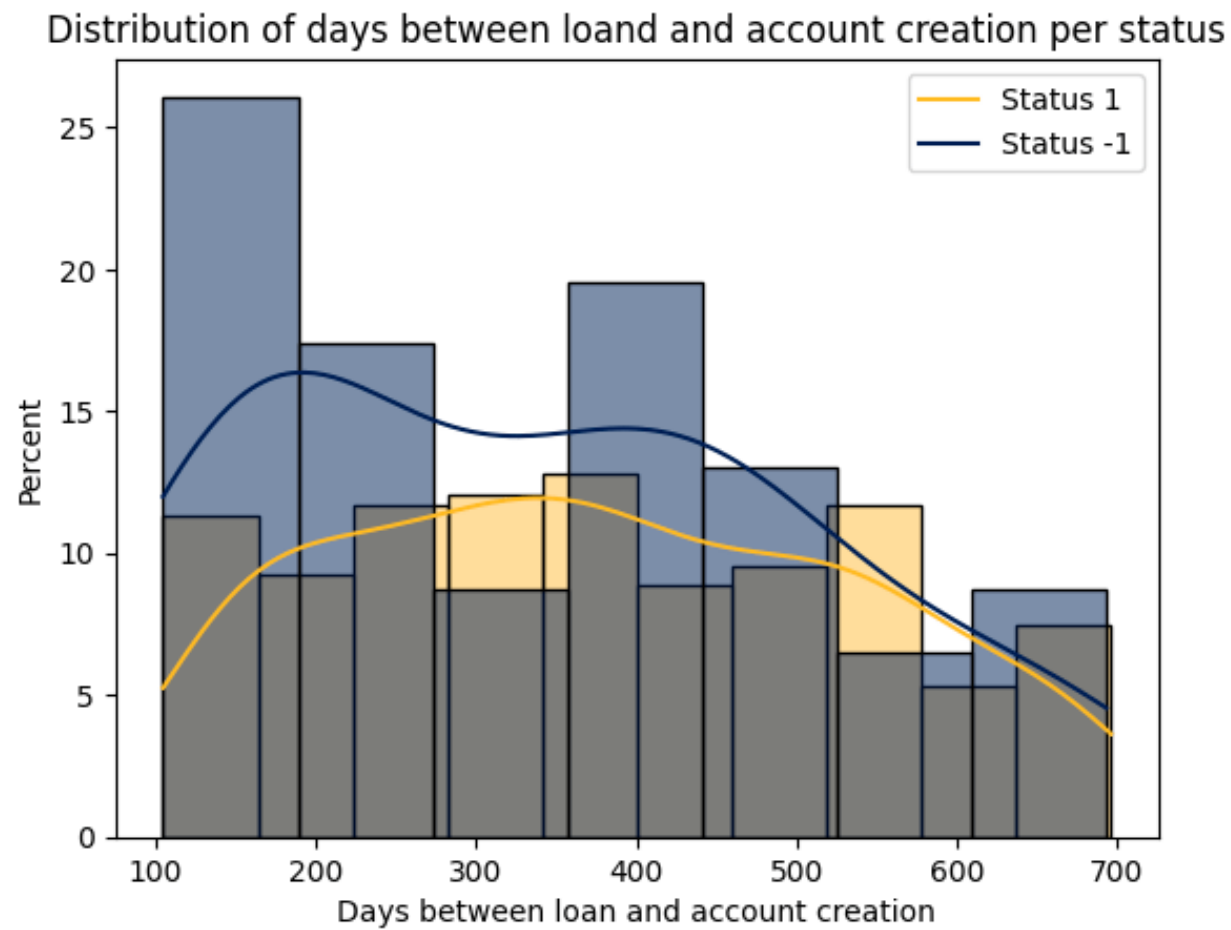
**loans associated with accounts where the withdrawal's average amount is**

- higher tend to succeed, as there is no record of unsuccessful loans where this value is higher than ~25000 (in absolute value)
- near 0 tend be fraudulent

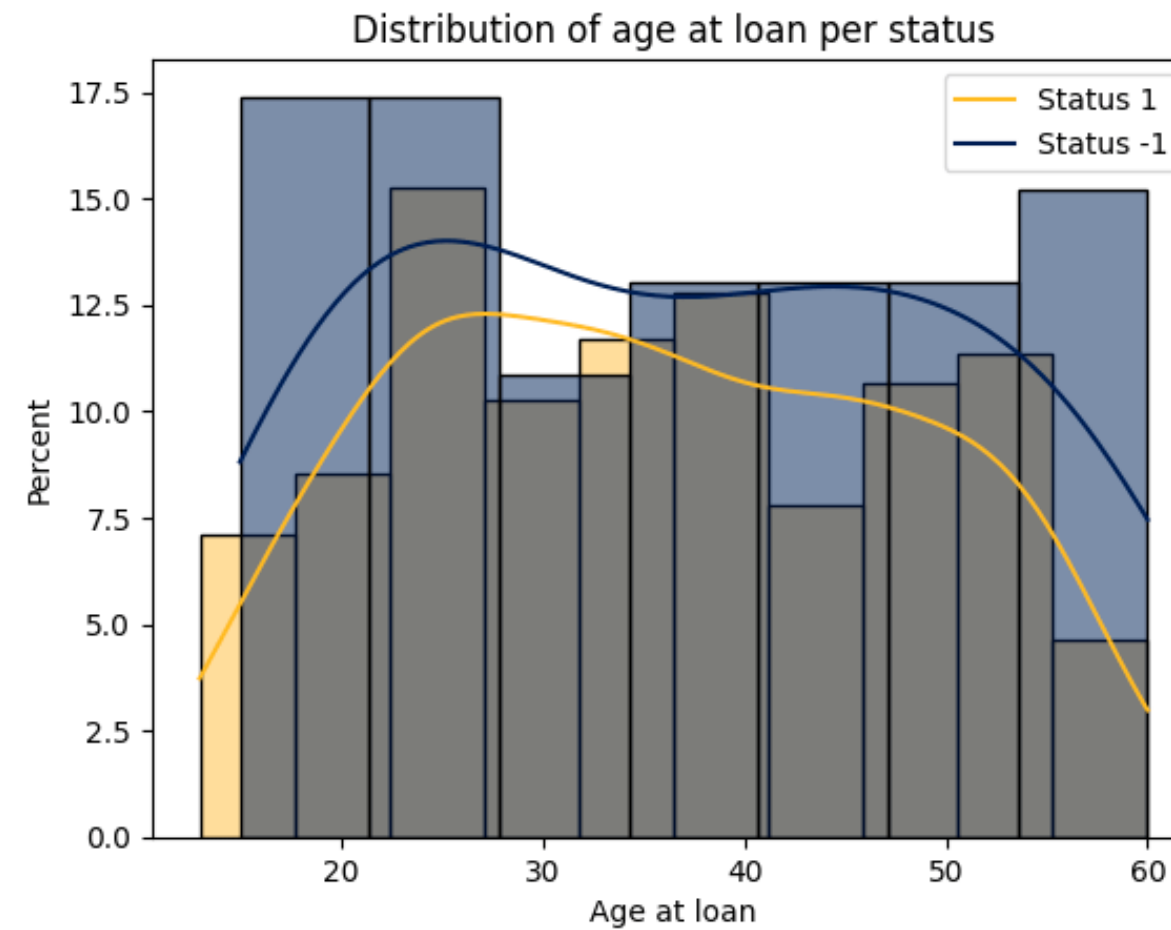


# Exploratory Data Analysis





- unsuccessful loans have a higher percentage for lower values however when taking into account the absolute number, nothing can be concluded



- age at loan distribution follows the expected pattern, not having a significant impact on the loan status
- peculiar loan issuance to underages

# Exploratory Data Analysis



Cardless accounts have no records of successful loans

Loans whose payments are fewer than 2000 and over 8000 payment units are prone to succeed

Accounts that have already had a negative balance have no records of successful loans

Districts with a higher rate of committed crimes have a higher average salary

Accounts that have an average balance below 20000 have no records of successful loans


The Moravia region contains almost 50% of fraudulent cases

# Exploratory Data Analysis

# Predictive Data Mining Problem




  
**Binary  
Classification  
Problem**

  
**Predict the  
probability of being  
a positive class**

  
**Positive  
Class**

Default Loan = -1

  
**Performance  
Metric**

AUC - measures the ability  
of a classifier to distinguish  
between two classes

Aiming to provide  
help in the loan  
granting decision  
process, resulting in  
a revenue increase  
by avoiding  
conceding loans to  
savage clients. We  
ought to **predict  
whether or not a  
granted loan will  
end successfully**

# Problem Definition

# Feature Engineering



## Financial features

- Number of transactions
- Transaction average amount & average amount per type
- Minimum & maximum transaction amount
- Credit ratio & withdrawal ratio
- Average, minimum, maximum & standard deviation transactions balance
- Has had a negative balance
- If the last transaction had a negative balance
- Average amount and count for each operation & k\_symbol
- Days between account creation & loan granting



## Client Account features

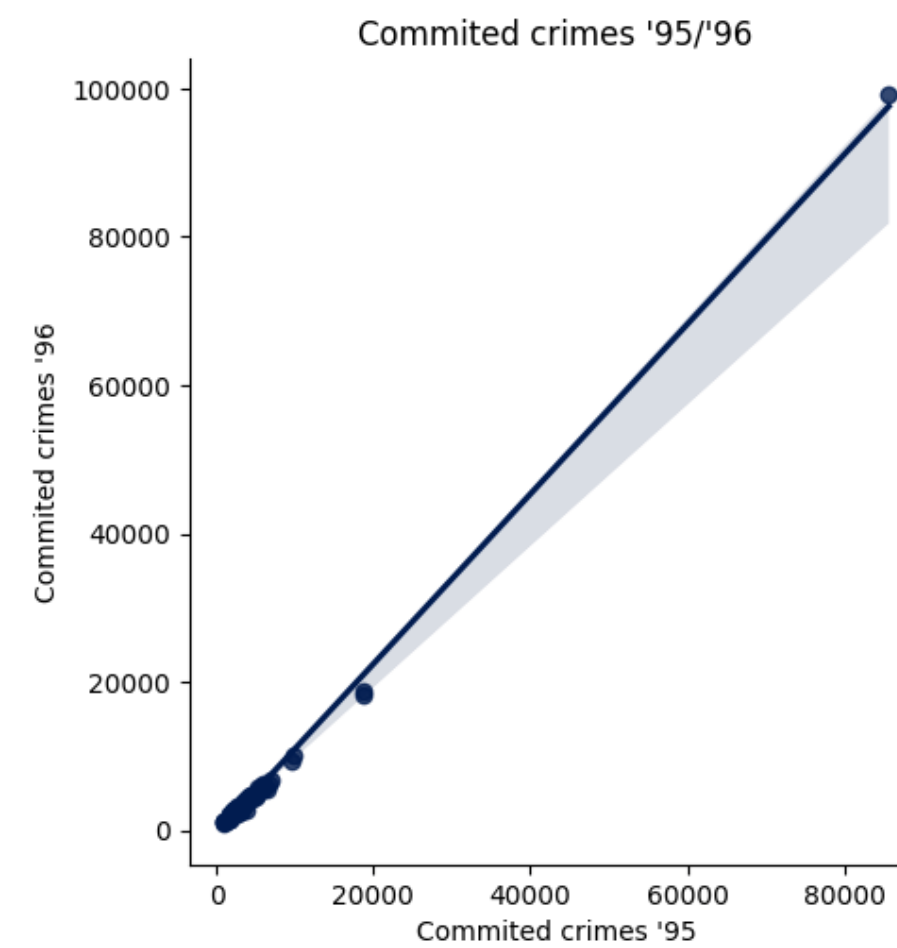
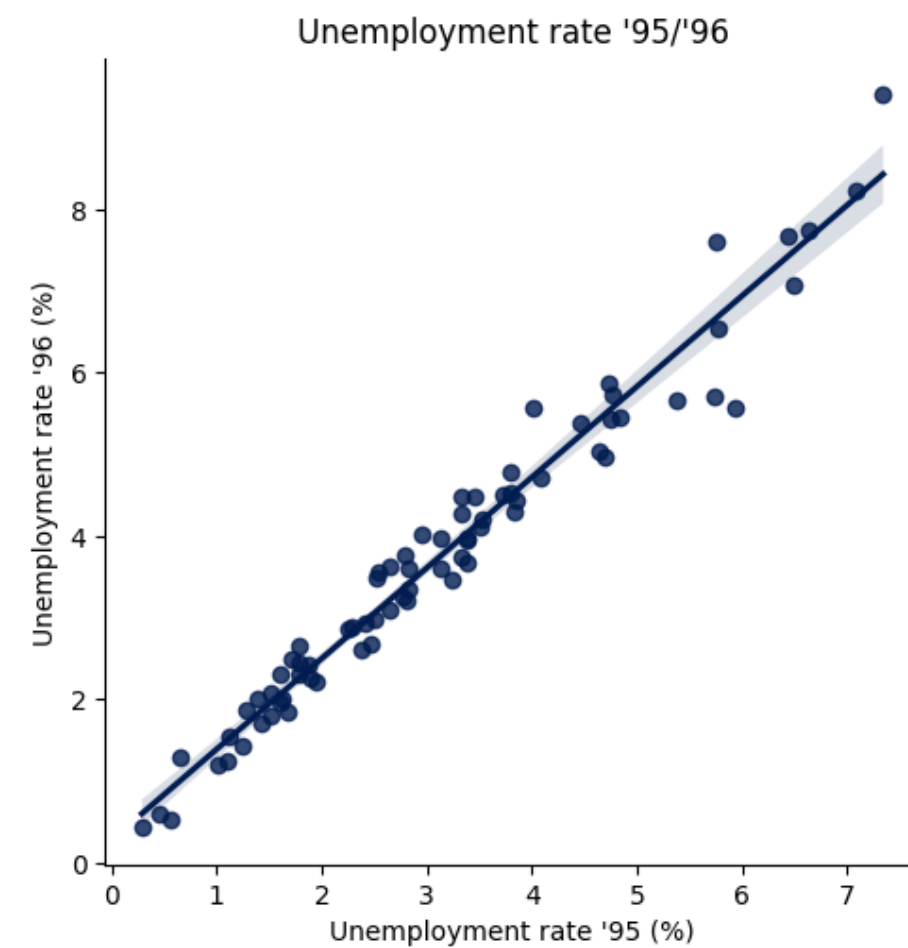
- Gender
- Age loan
- Number of disponents / has disponents
- Number of cards / has cards
- Same client and account's district



## Demographic features

- Criminality growth
- Unemployment growth
- Average committed crimes
- Average unemployment rate

# Data Preparation



# Missing Data

*Unemployment rate '95 and No. of committed crimes '95 missing values were **replaced** by correspondent '96 values.*

*\*Columns with more than 75% null values were removed.*

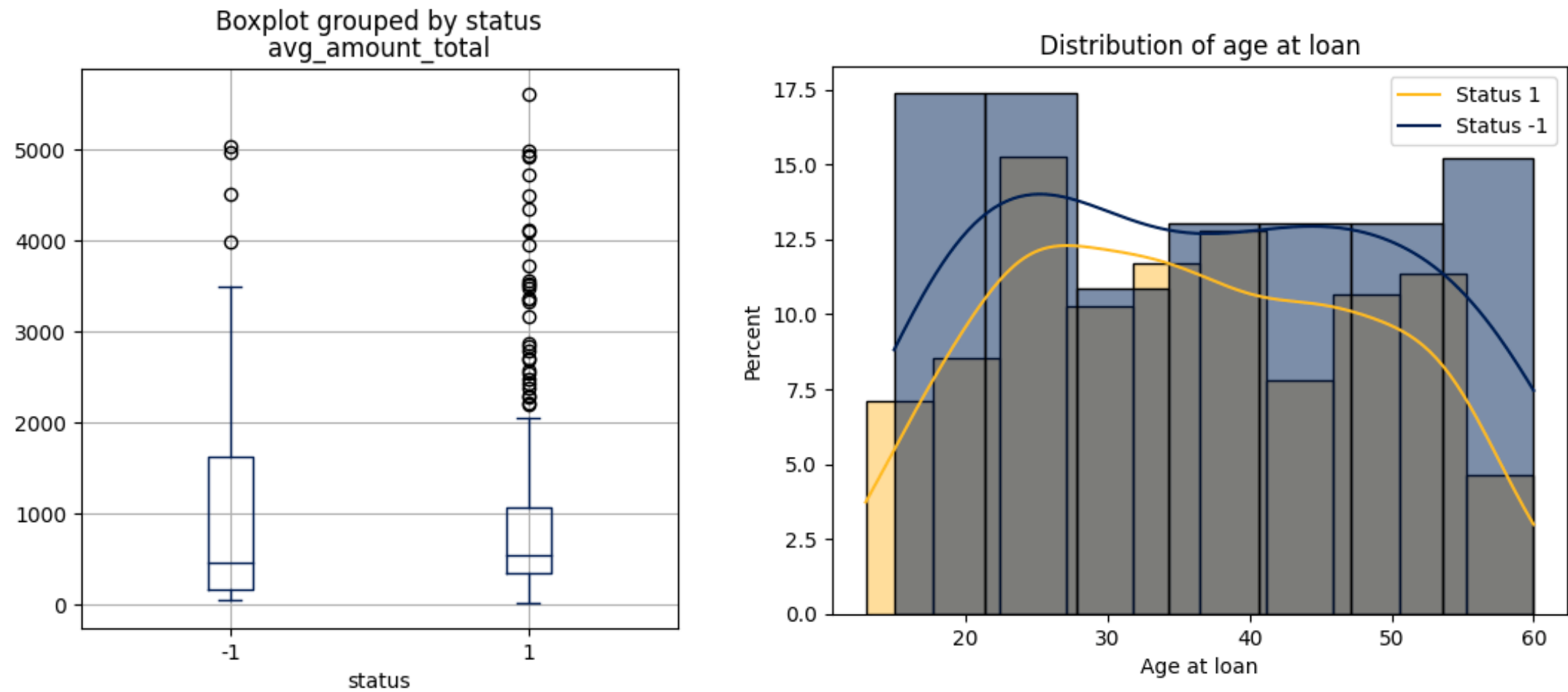
# Data Preparation

# Outliers

- a dataset without the amount outliers
- a dataset without ages at loan under 18

These two datasets were created to evaluate the impact of these outliers' presence.

Neither the datasets presented significant improvents in the results.



Algorithm	Dataset	Original	Amount Outliers	Age Outliers
Logistic Regression		0.87	0.82	0.87
K-Nearest Neighbours		0.81	0.79	0.75
Random Forest		0.77	0.77	0.76

[Table 1] Best AUC of each dataset using three different algorithms

# Data Preparation

# Data Transformation

- Format dates
- Encode categories
- Standardize column names
- Fix mislabelled data
- Drop irrelevant features
- Drop highly correlated features
- Drop features with many missing values

# Feature Selection

- **Wrapper Based** SequentialFeatureSelector with forward elimination
- **Filter Based** SelectKBest with f\_classif

# Imbalanced Data

Oversample using **SMOTE**.

# Data Preparation



Clean and  
prepare dataset

## Clean



## Evaluate

Evaluate models'  
performance using  
different algorithms  
and techniques

- Parameter Tunning
- Normalization & Scaling
- Oversample
- Feature Selection
- Cross Validation

Apply the model with the best  
AUC score to the test data and  
predict the probability of a  
loan being unsuccessful

## Test



## Submit

Submit a .csv file with the  
predictions result to  
Kaggle

# Experimental Setup

# Results

Oversample	Algorithm	Average AUC
	Decision Tree	0.8
	Logistic Regression	0.87
	Naive Bayes	0.79
	K Nearest Neighbours	0.81
	Random Forest	0.77
	Gradient Boosting	0.79
	SVC	0.81
	XGBoost	0.75
	MLP	0.83

[Table 2] Best AUC results for every algorithm used

## What about Feature Selection?



[Table 3] Detailed specification of the model with the best result

### Best Result

Algorithm		Logistic Regression
Parameters	C	0.01
	class_weight	balanced
	max_iter	1000
	penalty	l2
	solver	newton-cg

# Descriptive Data Mining Problem



# Requirements with the end user

The descriptive problem can be presented as the search for different profiles for the customers of a bank.

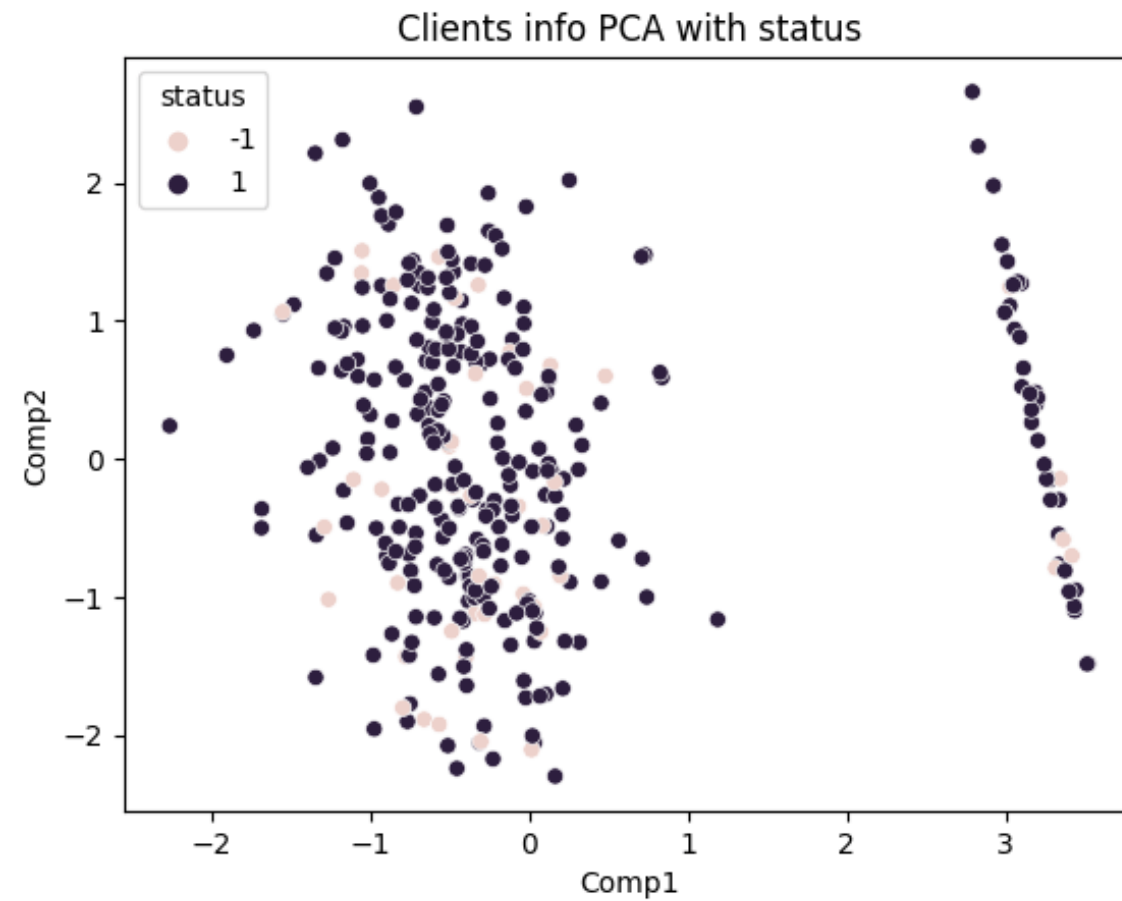
This information brings great value to bank managers as after tracing different groups of customers it is possible to create more mutually beneficial solutions, allowing for higher revenues and more satisfied customers.



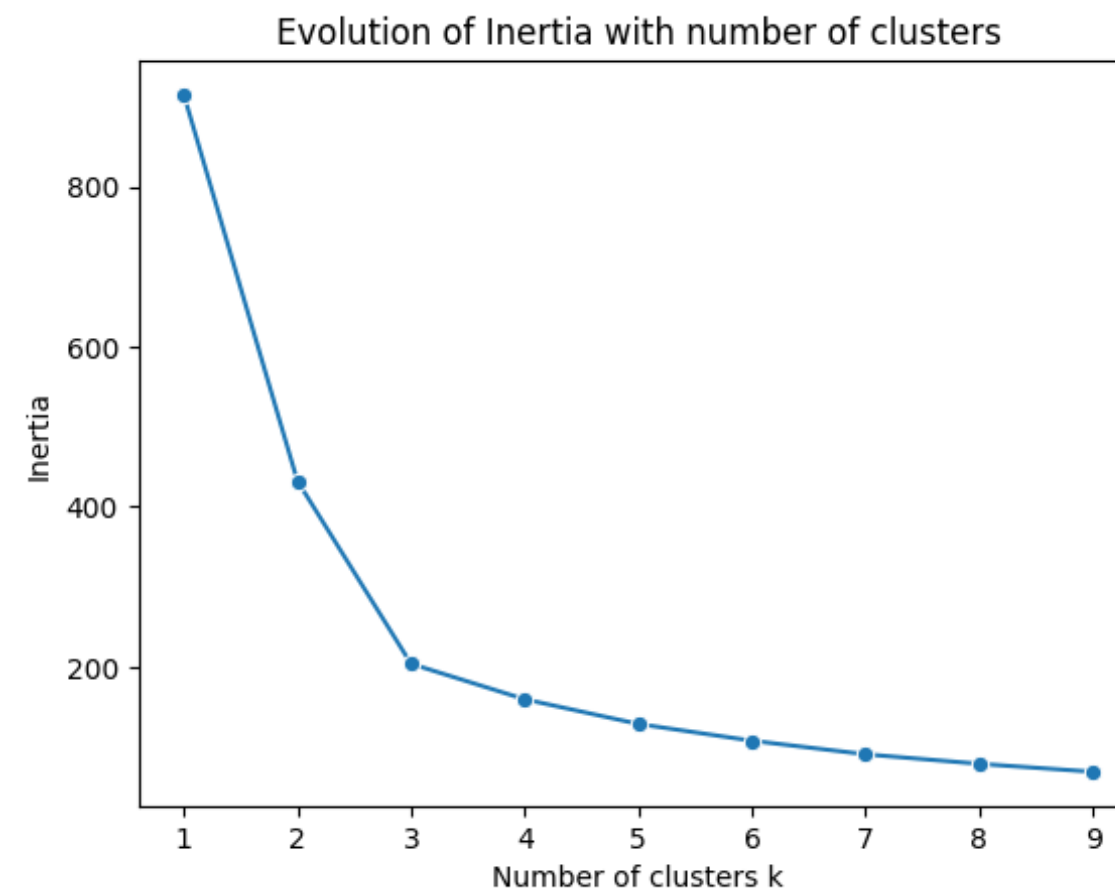
## Data Mining goals

Generate a number of clusters capable of distinguishing different relevant client profiles

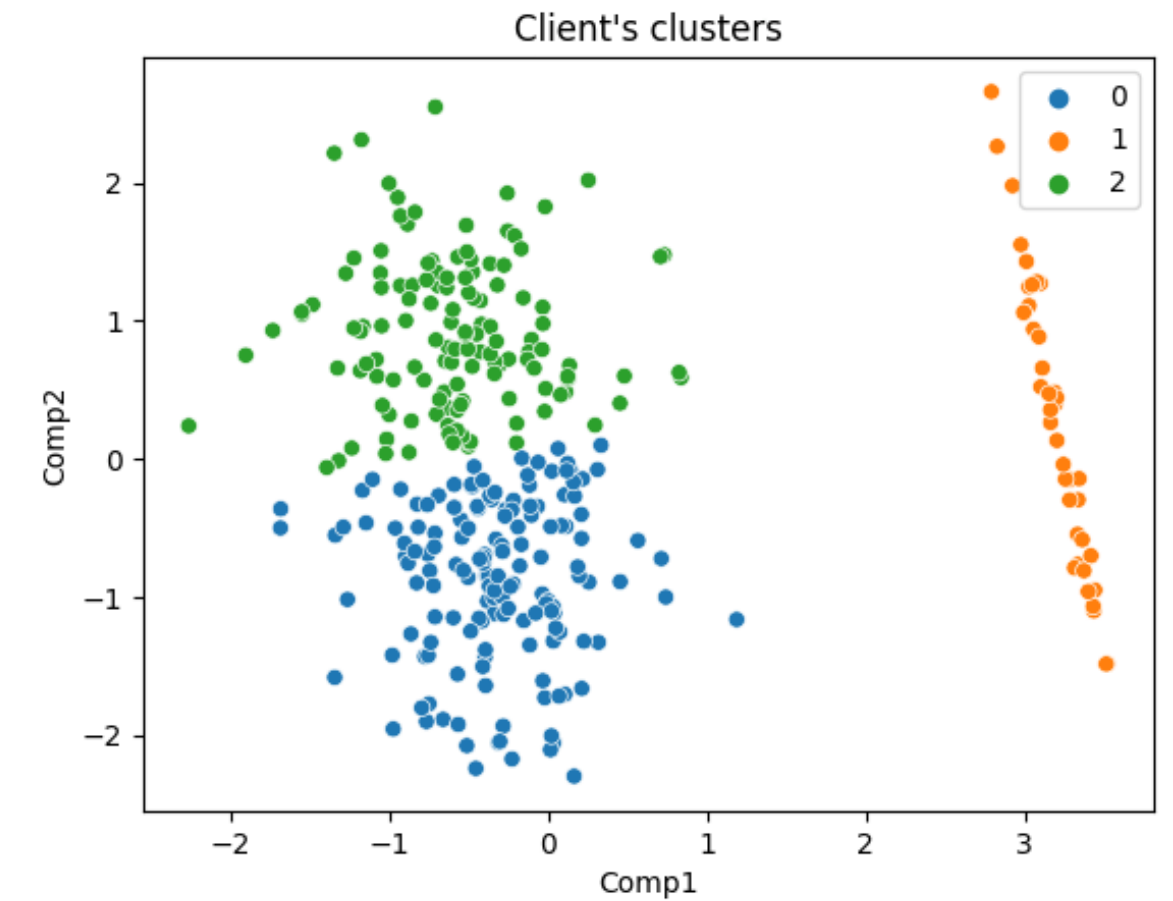
# Descriptive Problem



- PCA was used for reducing the dimensionality of our datasets. Used features: gender, age, avg salary, avg balance & criminality growth



- The Elbow method allowed to find the optimal number of clusters, k



- K-Means was employed for the clusters generation

# Clustering



ANNEX

Given that the bank stores accurate data about their clients and previous records (transactions, granted loans, etc.), our product aims to help bank managers decide which loan requests should be accepted or not. This will provide help in the loan granting decision process, via a data-based input that will lead to better results, i.e. the bank loans will mostly succeed, resulting in a revenue increase by means of loan interest rates.

By automating the predicting process of loan success payment, our product will avoid money loss while improving decisions, and also decrease labor, thus reducing work expenses.

# Requirements with the end user

## Business goals

- Reduce the bank's bad credit by 4%
  - Minimize the number of unsuccessful loans to avoid money loss
  - Correctly predict, at least, 8 out of 10 loans
- Automate the process of deciding whether or not a loan should be granted
  - Decrease labour
  - Accelerate the decision process
  - Unbias decisions
- The goal is achieved on time, i.e. the final model must be completed at the end of the due date

## Data Mining goals

- Building a model to predict the probability of a loan being successful
- 1 if unsuccessful, 0 otherwise
- Obtain an AUC of, at least, 0.8
- Induce a higher weight for false-negative outcomes, due to the more significant impact they have compared to false positives; although, aiming to decrease both results

# Business Understanding





# Account

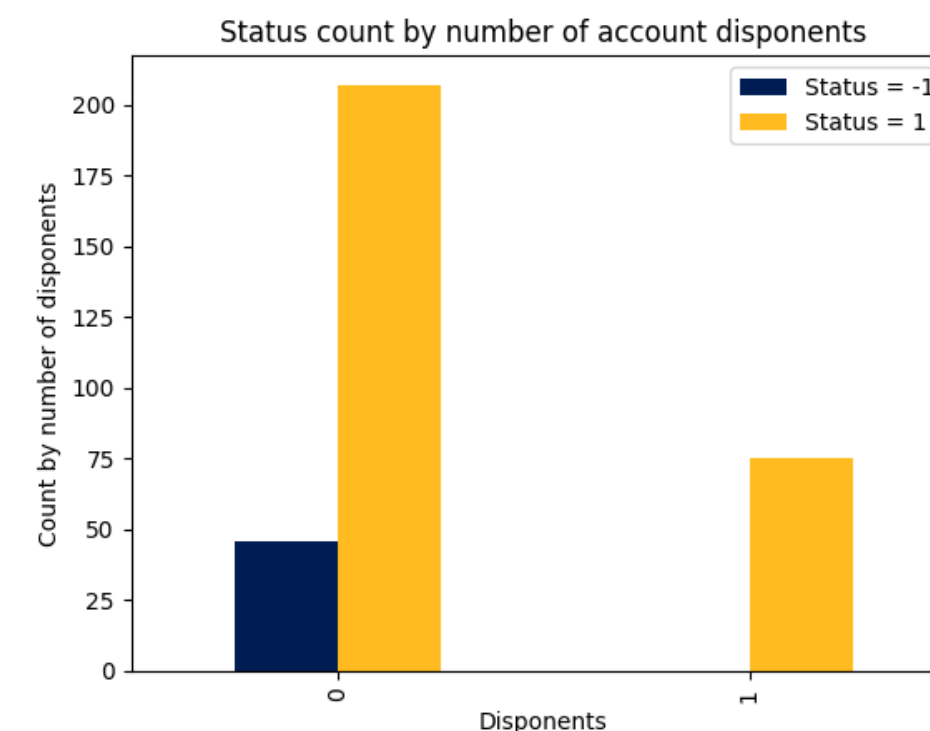
- Format the account **creation date**
- Encode **frequency** into numerical values
- Rename **district\_id** to **account\_district\_id**

# Disponent

- Drop non-owners, since only an account owner can request a loan
- Calculate the number of disponents per account to evaluate the impact of an account not being managed by only one client
- Create **has\_disponent** feature, after concluding that loans associated with accounts with a disponent other than the owner tend to be fraudulent

# Client

- From birth\_number extract **gender** and **birth\_date**
- Rename **district\_id** to **client\_district\_id**
- Drop **birth\_number** as it was no longer needed

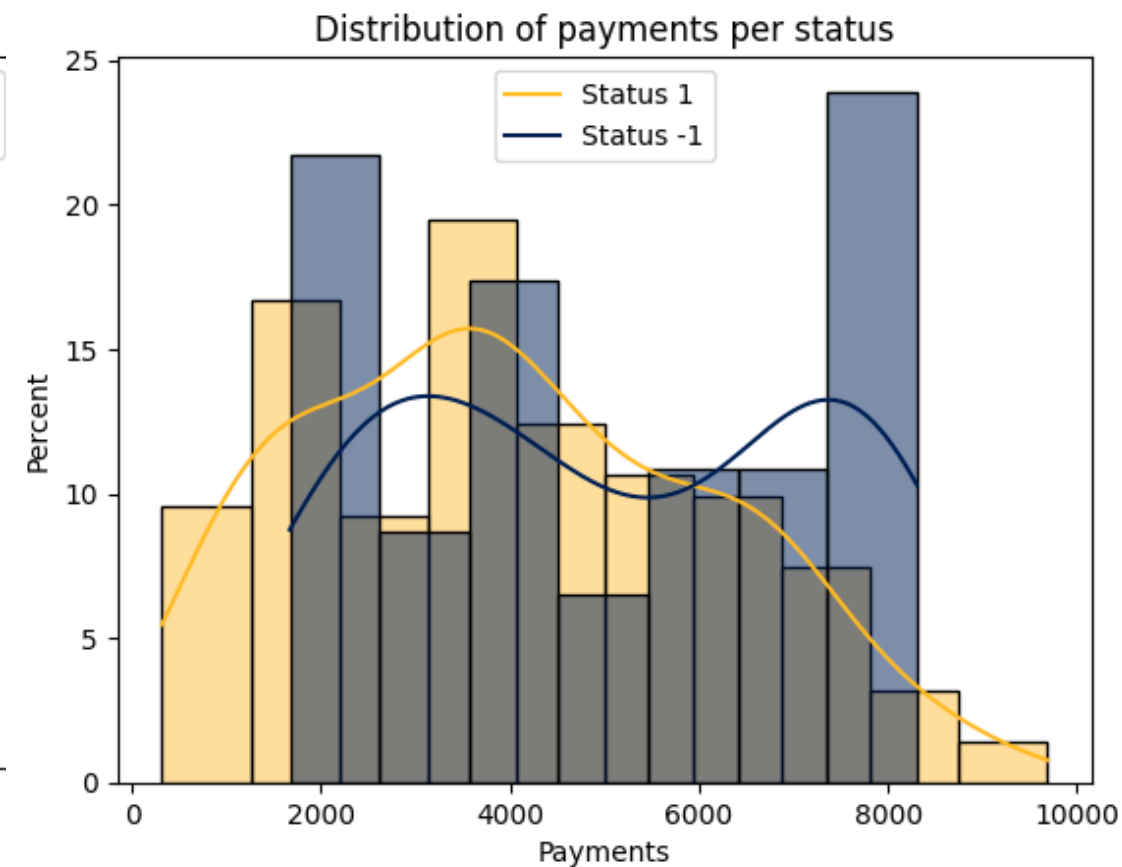
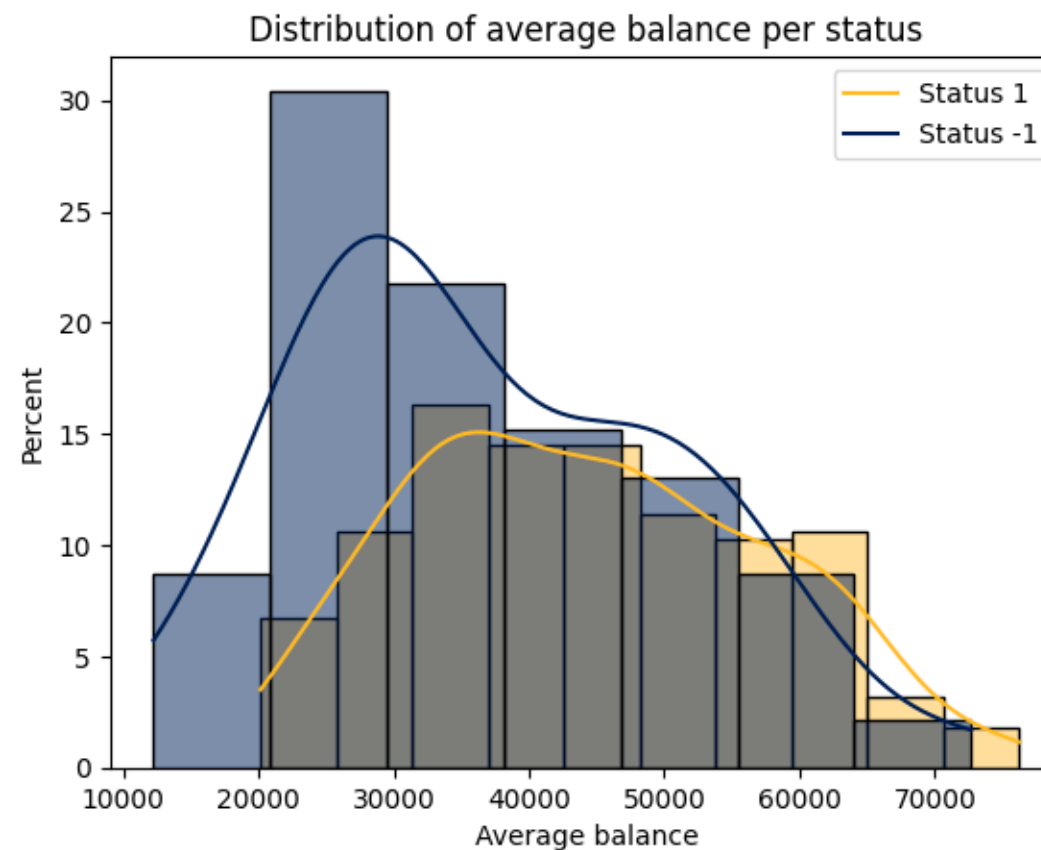


# Feature Importance



# Card

- Calculate number of cards per account
- Create **has\_card** feature, upon verifying that accounts with card had no record of being fraudulent



# Transactions

- Replace empty strings with NaN's
- Drop **bank** column for its null values
- Replace null values of **operation** with *interested credit*
- Replace *withdrawal in cash* with *withdrawal* in **type** column
- Calculate **average\_amount** by **type**
- Calculate **amount** by type
- Calculate **max, min, max balance & number of transactions**

# Loan

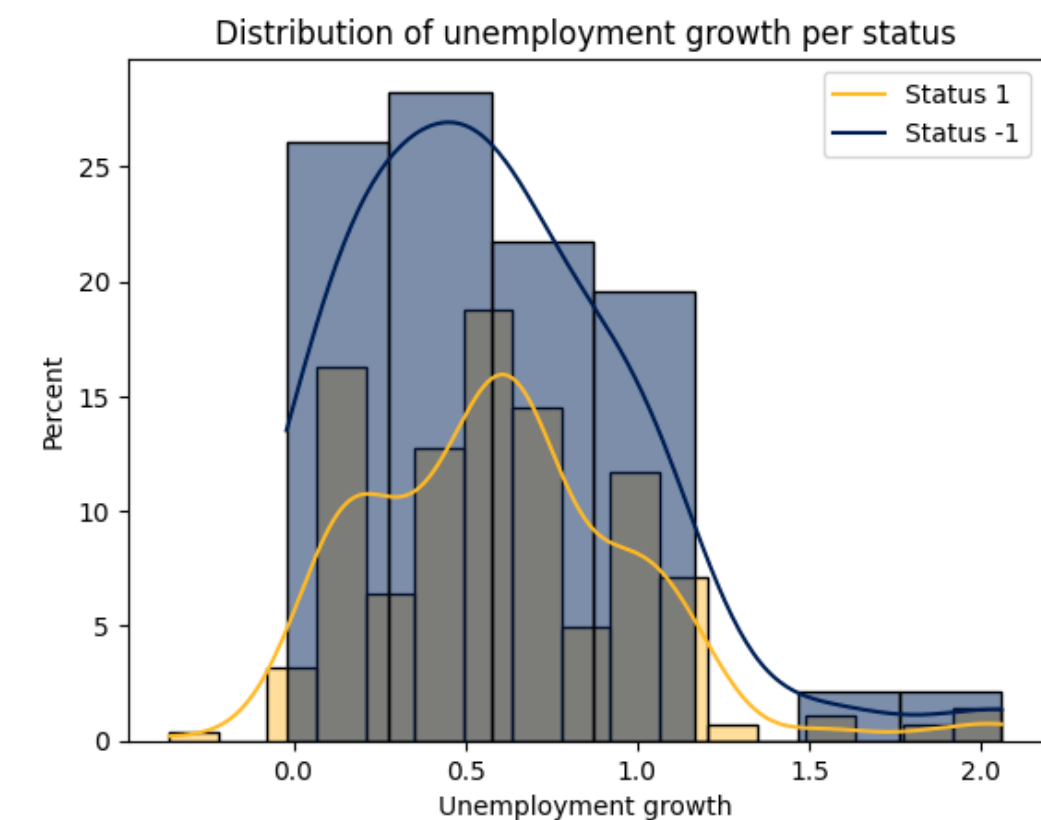
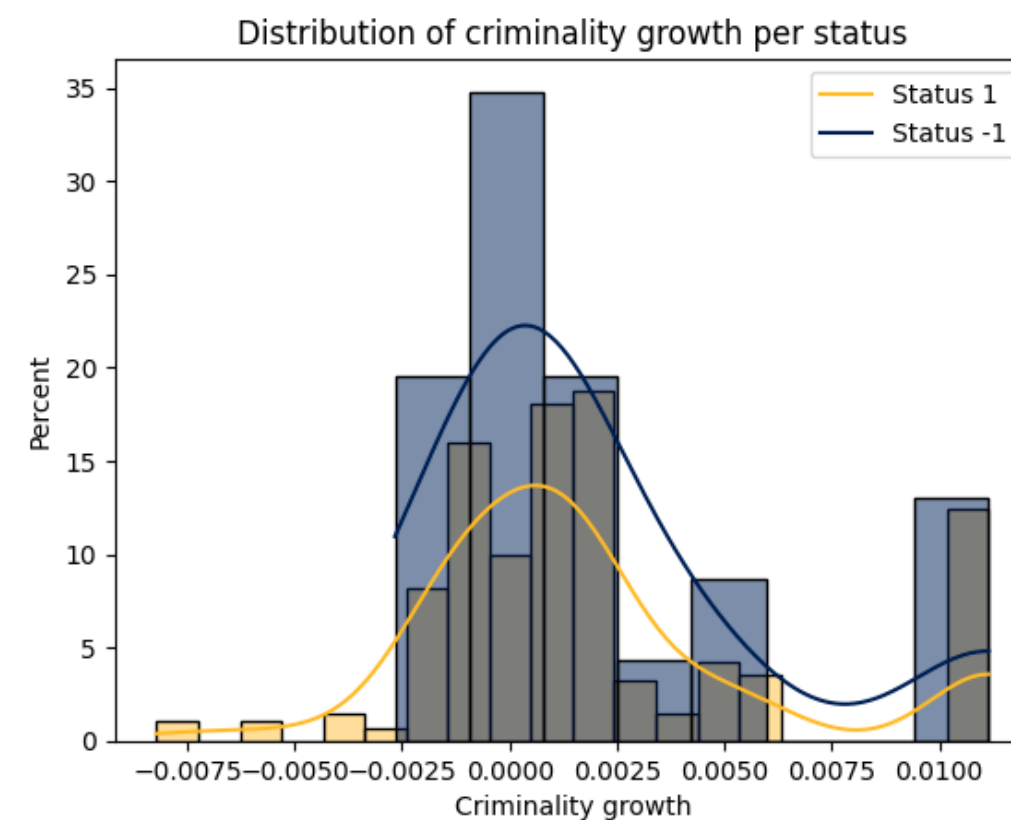
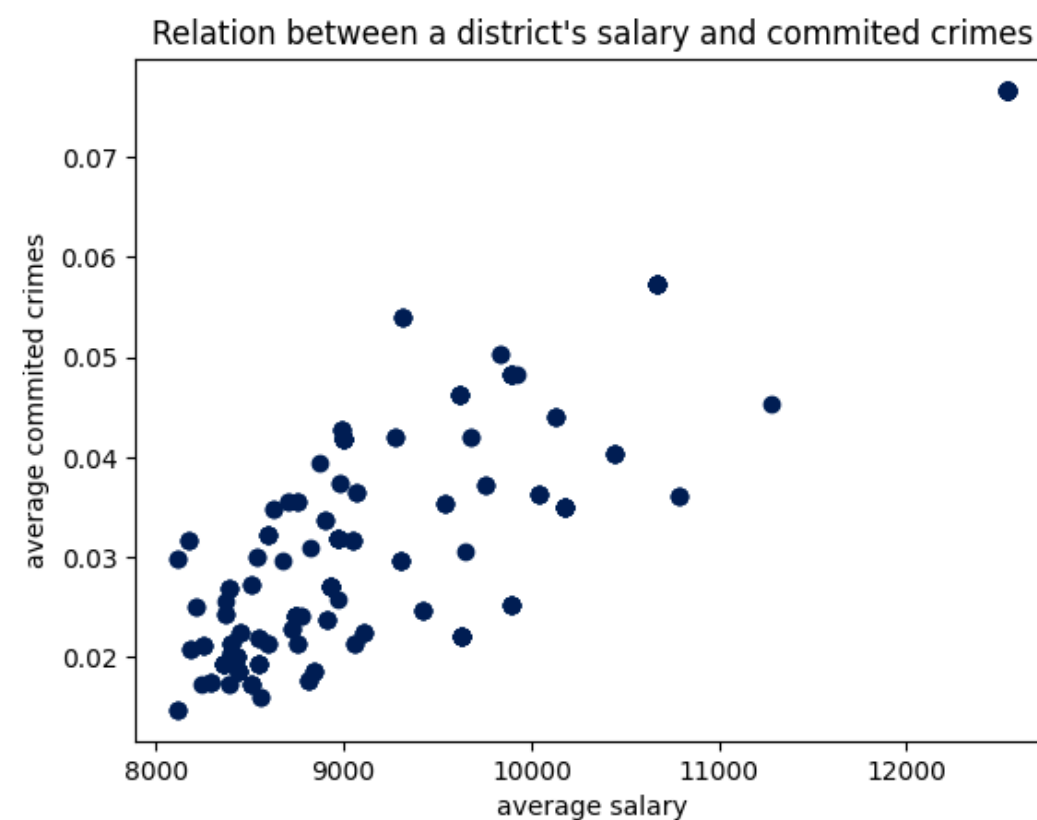
- Format date
- Change **date** to **loan\_date**

# Feature Importance



# District

- Replace missing values of **unemployment\_rate\_95** and **nr\_committed\_crimes\_95** with the correspondent 96 values
- Encode **region** into numerical values



## Feature Importance

# PCA

- Principal Component Analysis was used for reducing the dimensionality of our datasets, increasing interpretability but at the same time minimizing information loss.

# Elbow Method

- For several possible number of clusters k:
  - Calculate the within-cluster SSE, also called distortion;
  - Choose the k so that adding another cluster doesn't yield a much smaller SSE.

# Silhouette Coefficient

- The silhouette coefficient was also calculated for evaluating the algorithm's performance.

# K-Means

- K-Means is a partition-based method that obtains k groups of a dataset.
  - A fast algorithm that scales well;
  - Uses a stochastic approach that frequently works well.

Cluster	Gender	Age	Average salary	Criminality Growth	Average balance
0	0.732026	68	9127	0.0004	36190
1	0.609756	65	12541	0.0111	42203
2	0.216418	57	9051	0.0008	51735

[Table 3] Average feature values by cluster

# Descriptive Ideas

# Decision Tree

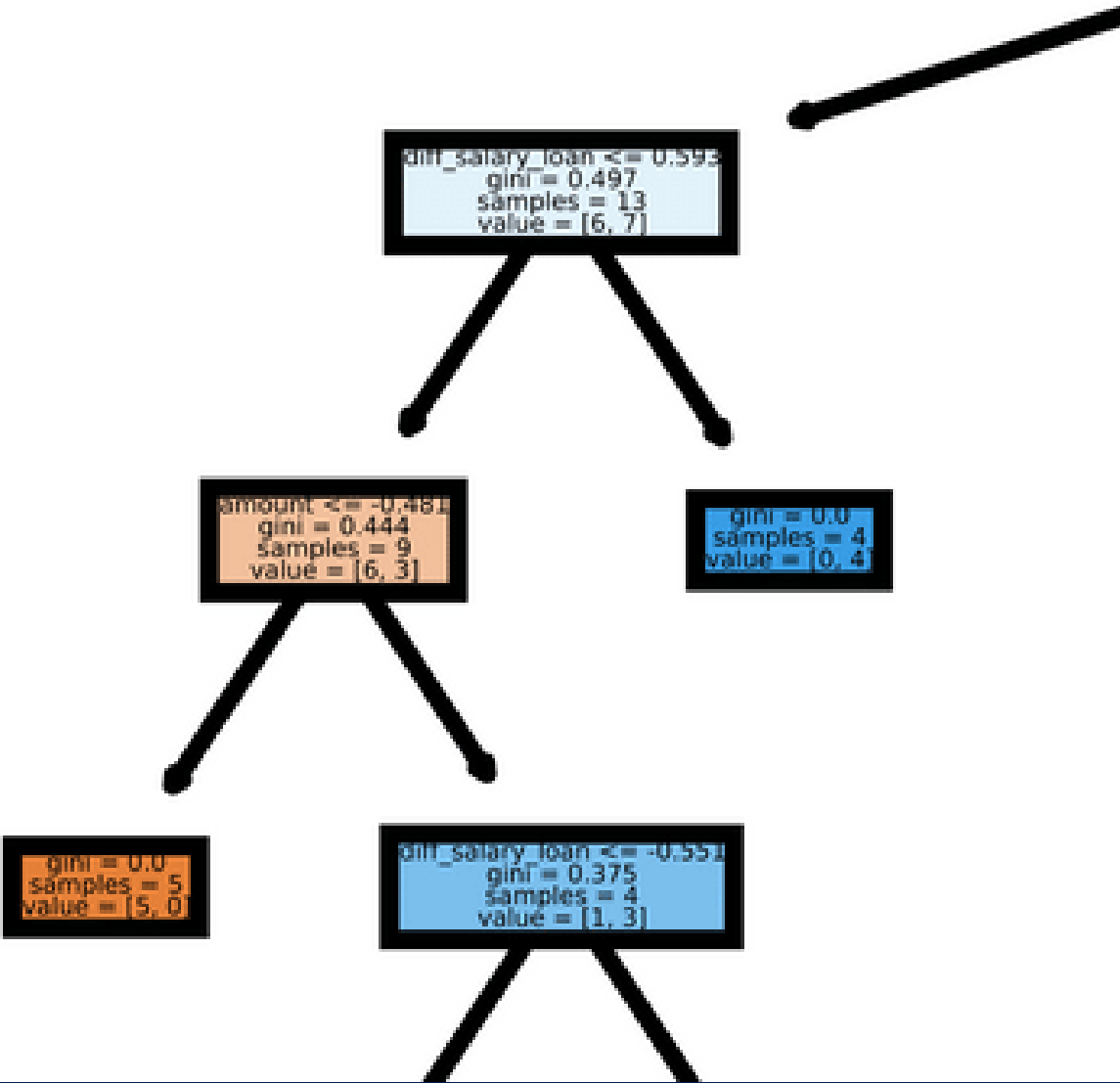
- Decision Trees are easy to understand and interpret - **white-box**;
- Implicitly perform variable screening or feature selection;
- Are prone to creating over-complex trees that do not generalize the data well, i.e., **overfitting**;
- Unstable because small variations in the data might result in a completely different tree being generated. This variance needs to be lowered by methods like bagging and boosting.

## Grid search parameters

**criterion:** ['gini', 'entropy', 'log\_loss']  
**splitter:** ['best', 'random']  
**max\_depth:** range(1, 7)

Metrics	Oversample		Feature Selection	Feature Selection & Oversample
Fit time	0.001	0.005	0.001	0.001
Score time	0.001	0.001	0.001	0.001
Accuracy	0.895	0.733	0.908	0.786
ROC AUC	0.686	0.649	0.792	0.772
Precision	0.8	0.24	0.75	0.35

[Table 4] Example output of the Decision Tree Algorithm



# Predictive Algorithms

# Logistic Regression

- Logistic Regression, despite its name, is a linear model for classification rather than regression;
- Analyzes the relationship between one or more independent variables and classifies data into discrete classes;
- Suitable for linearly separable datasets.

## Grid search parameters

**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]  
**max\_iter:** [500, 1000, 5000, 10000]

Metrics	Oversample		Feature Selection	Feature Selection & Oversample
Fit time	0.002	0.230	0.002	0.012
Score time	0.001	0.003	0.001	0.001
Accuracy	0.886	0.834	0.860	0.807
<b>ROC AUC</b>	0.86	0.907	0.793	0.794
Precision	0.613	0.437	0.0	0.408

[Table 5] Example output of the Logistic Regression Algorithm

# Random Forest

- Random Forest is an ensemble learning method that averages multiple decision trees, trained on different parts of the same data;
- It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than of any individual tree.

## Grid search parameters

**criterion:** ['gini', 'entropy']  
**n\_estimators:** [10, 50, 100, 200]  
**max\_depth:** [5, 10, None]  
**n\_jobs:** [-1]

Metrics	Oversample		Feature Selection	Feature Selection & Oversample
Fit time	0.071	0.025	0.031	0.008
Score time	0.023	0.010	0.014	0.006
Accuracy	0.899	0.860	0.882	0.877
<b>ROC AUC</b>	0.791	0.785	0.809	0.787
Precision	1.0	0.422	0.816	0.565

[Table 6] Example output of the Random Forest Algorithm



# Naive Bayes

## Gaussian

- Gaussian Naive Bayes is the extension of naive Bayes. It is simpler as it only calculates the mean and standard deviation for the training data;
- Assumes that the value of **one feature is independent of the value of any other feature**;
- Simple design and application, and, therefore, suitable in many real-life scenarios.

Metrics		Oversample	Feature Selection	Feature Selection & Oversample
Fit time	0.0007	0.003	0.001	0.012
Score time	0.001	0.001	0.001	0.001
Accuracy	0.458	0.475	0.886	0.851
<b>ROC AUC</b>	0.791	0.808	0.766	0.810
Precision	0.189	0.22	0.85	0.4

[Table 7] Example output of the Naïve Bayes Algorithm

# K-Nearest Neighbours

- K-Nearest Neighbours is a non-parametric, supervised learning classifier, which **uses proximity** to make classifications or predictions;
- The goal is to identify the nearest neighbors of a given query point so that it can assign a class label to that point;
- Is a lazy learner algorithm as it does not learn from the training set immediately instead, stores the dataset and only performs actions on it when its actually classifying the new data.

## Grid search parameters

**n\_neighbors:** [4, 5, 6, 7, 10, 15]

**leaf\_size:** [5, 10, 15, 20, 50, 100]

Metrics	Oversample		Feature Selection	Feature Selection & Oversample
Fit time	0.0006	0.001	0.001	0.001
Score time	0.003	0.005	0.007	0.008
Accuracy	0.882	0.62	0.895	0.790
<b>ROC AUC</b>	0.741	0.681	0.816	0.858
Precision	0.9	0.223	0.8	0.388

[Table 8] Example output of the K-Nearest Neighbours Algorithm

# XGBClassifier

- XGBClassifier is a boosting algorithm based on gradient-boosted decision trees algorithm;
- It works by building weak prediction models sequentially, minimizing the error in each next attempt;
- Needs proper **parameter tuning** to avoid overfitting;
- Difficul to interpret, **black-box** model.

## Grid search parameters

**min\_child\_weight:** [4, 5, 6, 7, 10, 15]  
**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]

Metrics	Oversample		Feature Selection	Feature Selection & Oversample
Fit time	0.237	0.449	0.207	0.28
Score time	0.003	0.003	0.003	0.003
Accuracy	0.877	0.838	0.869	0.86
<b>ROC AUC</b>	0.805	0.770	0.772	0.78
Precision	0.686	0.332	0.4	0.59

[Table 9] Example output of the XGBClassifier Algorithm

# SVC

- Support Vector Classification's goal is to find a hyperplane in an N-dimensional space that distinctly classifies the data points;
- Suitable for linearly separable data.

## Grid search parameters

**C:** [1, 10, 50]  
**gamma:** [0.001, 0.0001, 0.01, 1, 'scale', 'auto']  
**kernel:** ['linear', 'poly', 'rbf', 'sigmoid']  
**degree:** [1, 2, 3, 4, 5, 6, 7]  
**coef0:** [0.0, 0.1, 0.3, 0.5, 0.7]  
**max\_iter:** [1, 2, 3, 5, 7, 10]  
**decision\_function\_shape:** ['ovo', 'ovr']  
**class\_weight:** [None, 'balanced', dict]

Metrics	Oversample		Feature Selection	Feature Selection & Oversample
Fit time	0.0006	0.004	0.001	0.003
Score time	0.001	0.001	0.001	0.001
Accuracy	0.41	0.58	0.71	0.79
<b>ROC AUC</b>	0.55	0.64	0.64	0.81
Precision	0.15	0.18	0.22	0.77

[Table 10] Example output of the SVC Algorithm

# Assumptions


- Decision Tree's results were due to its sensitivity to small perturbations and easy overfitting.
- Having Logistic Regression and SVC with the best results leads us to assume our data is linearly separated.
- KNN has significantly better results with feature selection which means that some of the features used weaken the distribution of the dataset in classes.

# Conclusions

- Using oversample retrieves the best results. Without it, classifiers tend to predict a probability towards the negative class (status=1), due to the lack of comparison data.
- RandomForest and XGBClassifier take the longest as they are ensemble learning methods.

# Algorithms Analysis

# Project Management and Collaboration

Organization and methodology are crucial in a collaborative project. We used  as the hosting platform for version control and as the project management tool (main issues definition).

 **Discord** was used for communication, remote collaboration sessions, and week planning TODOs.

Finally, we followed a **CRISP-DM** methodology as the base of our data science process. The six commonly described phases were revised cyclically allowing for a sequential improvement over the end-results.

# Data visualization, cleaning and training



Matplotlib & Seaborn

Numpy

Pandas

Scikit-learn

Plotly

# Tools

# Future Work

- Invest even more in feature engineer - features are never enough
- Cost-sensitive learning
- A deeper exploration of feature selection
- Help the bank managers to create different client profiles with the results obtained from the descriptive analysis



# Conclusions

- By adding new features, we were able to iteratively evaluate and test different algorithms with different datasets, greatly improving our understanding of the problem and the results
- Feature engineering over unused tables brought us the biggest improvements
- Oversampling techniques were crucial since the two classes were imbalanced