Bank Loan Prediction

MEIC - Aprendizagem Computacional 2022/2023

Duarte Sardão - up201905497@up.pt | Gabriel Martins - up201906072@up.pt | Miguel Lopes - up201704590@up.pt | Sérgio Estêvão - up201905680@up.pt

Domain Description

Dataset Composition

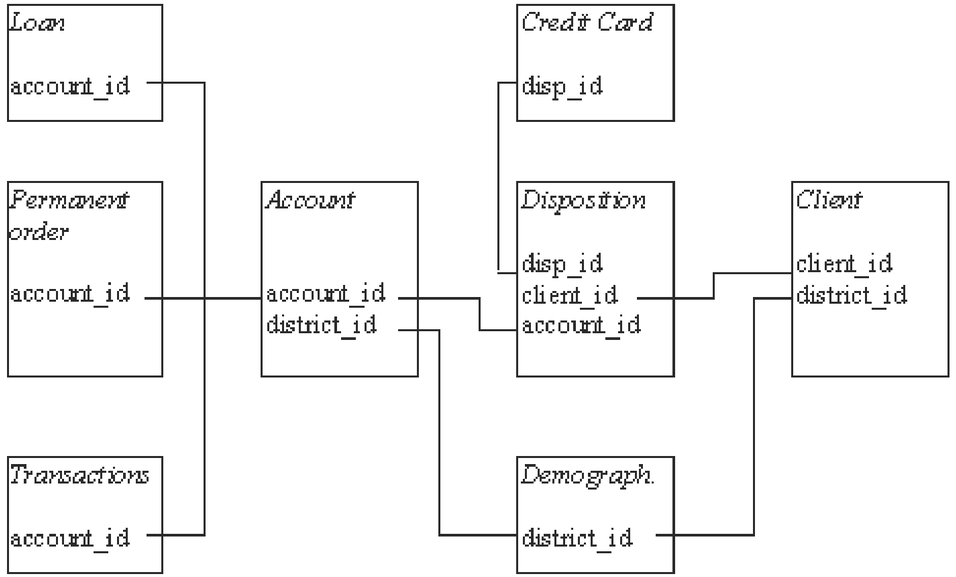
Dataset

The dataset is composed of a series of information describing a Czech bank's activity during the 90s. This includes account, client, credit cards, transaction, loan information, and information regarding the districts where the bank's clients reside.

The records of the Czech bank contain the following entries:

- 4500 accounts
- 5369 clients
- 5369 dispositions
- 396 685 transactions
- 328 loans
- 177 cards
- 77 districts with demographic data

Relational Model of the Dataset

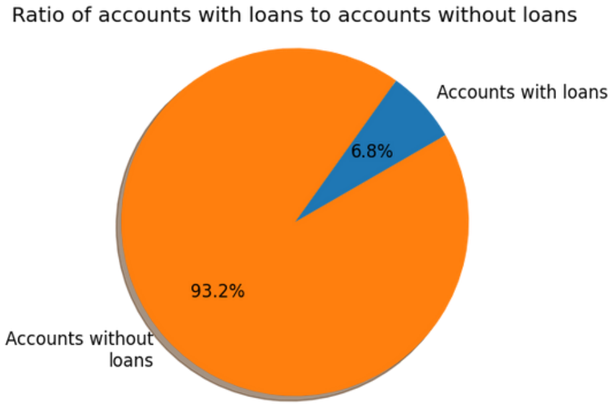


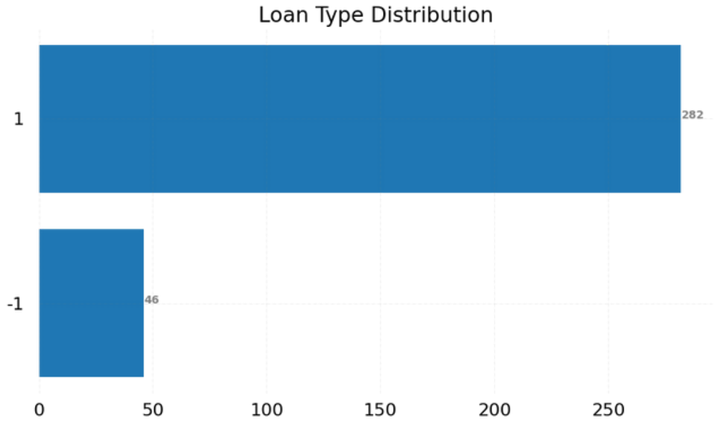
Exploratory Data Analysis

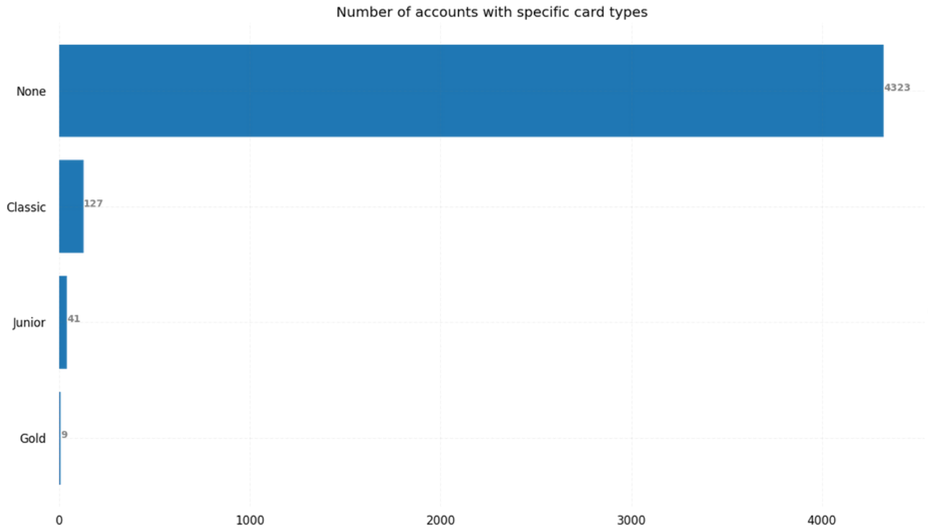
Jupyter Notebook (Python3 & Libraries, e.g. Matplotlib, Pandas, Seaborn), Excel

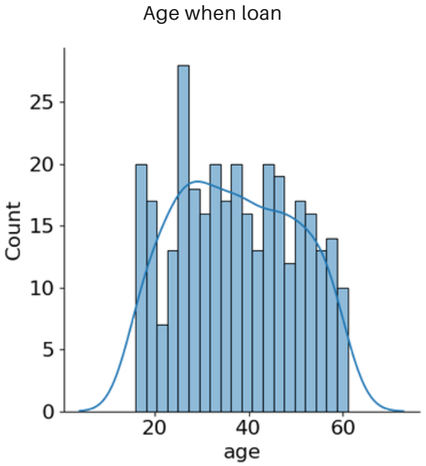
Findings

- The majority of the accounts didn’t have loans, which meant that most of the data couldn't be used
- The loan distribution is unbalanced. Only 14% of the loans were ‘-1’, which hinders the prediction model results
- The great majority of the accounts didn’t have cards associated with them
- The age at which the client takes a loan follows approximately a normal distribution, as expected. The most common age group is between 24 and 40 years old







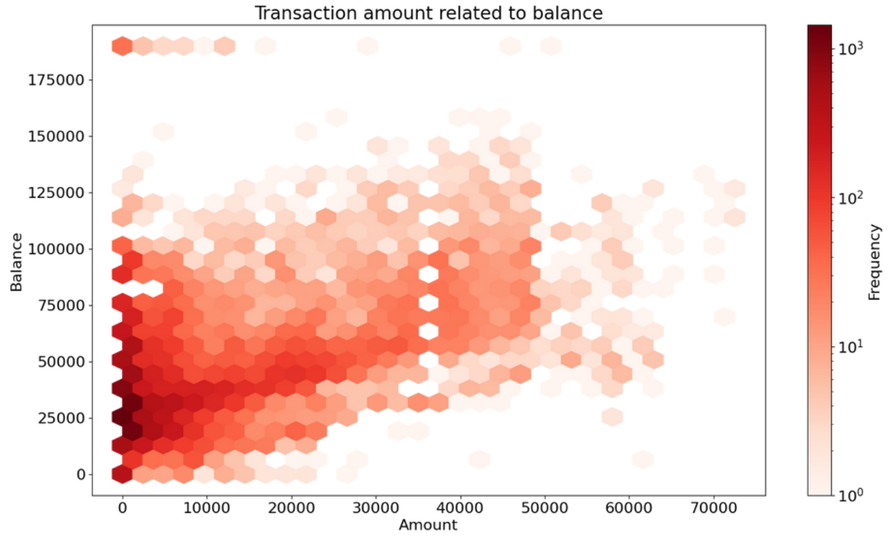


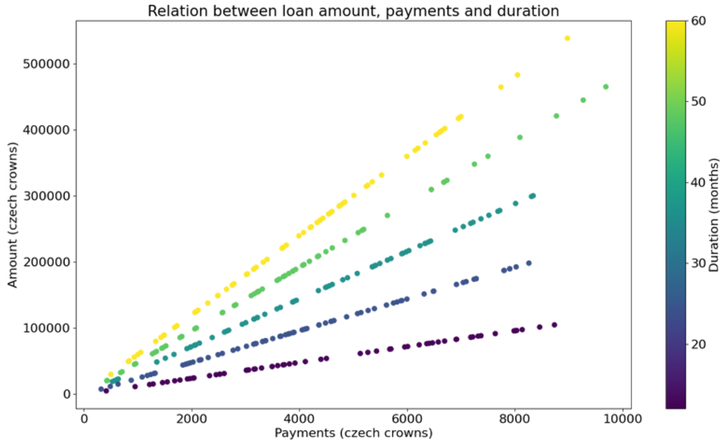
Exploratory Data Analysis

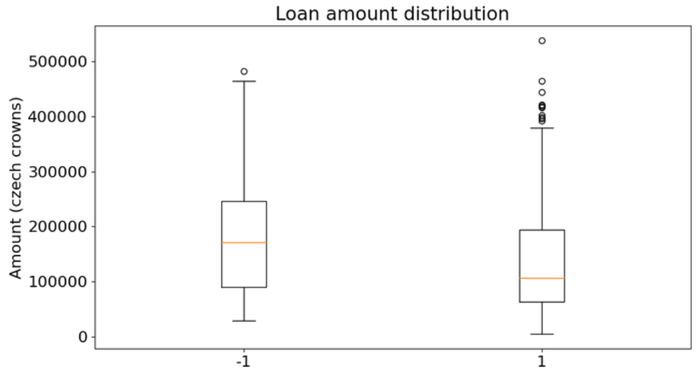
Jupyter Notebook (Python3 & Libraries, e.g. Matplotlib, Pandas, Seaborn), Excel

Findings

- The transactions follow a downward trend. Most are of lower amounts from accounts with a lower balance
- Some outliers can be seen: transactions of lower amounts from accounts with a higher balance and transactions of higher amounts from accounts with a medium balance
- A gap in the transactions of amounts rounding 35.000 <-> 37.500 Kč, no apparent cause
- As expected, a correlation between the loan amount, duration, and payments
- No interest rate is applied by the bank (loan amount = payments ∗ duration)
- The loan amounts related to the paid loans (1) were, on average, smaller and had a higher amount of outliers
- No relevant correlation was found between gender and loan results or balance amounts







Problem Definition

Business Understanding and Data Mining Goals

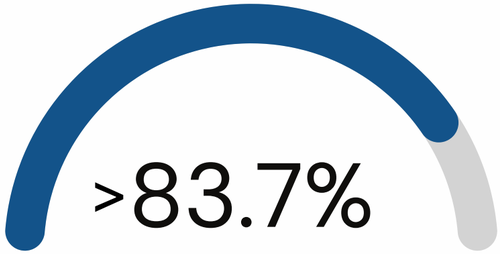
The bank wishes to increase their operating income by using data mining to predict the outcome the loans they grant

1. **Reduce the number of employees and time spent on loan risk analysis**
2. **Increase the number of loans that are paid in full**
3. **Decrease the number of loan defaults**

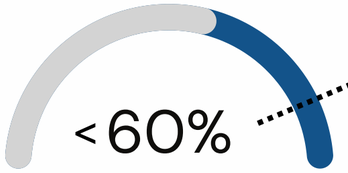
16.3%
Current default rate*

≈8 million Kč
Amount lost from defaulted loans*
(in Czech Korunas)

Our goal is to **decrease the number of actual defaults** (priority), by achieving a higher NPV value, whilst at the same time reducing the number of loans incorrectly marked as defaults (false positives), by achieving a lower FPR value



Negative Predictive Value (NPV)**



False Positive Rate (FPR)**

60%
average denial rate for credit cards in the US.

* Rate calculated with the data provided initially | ** Considering loan defaults as positives and payed loans as negatives

Data Preparation

Data Preparation Pipeline - Main Operations Performed

Removing information that has no relation to loans

Although we were given a substantial amount of data, a lot of it was about clients that had no loans nor any relation to the loan bearer

Extract the client's birth date and gender from the "birth_number"

The column birth_number contained information regarding both the birthdate and the client's gender

Extracted new parameters from several of the datasets

Created columns like crime rate, recent balance, household payments, insurance payments, sanction payment counter, to name just a few

Converted dates into time intervals

Since most classification models don't accept a date as an input type, we converted dates like, the client's birthdate, the account creation date or the date of issuance of the credit cards into a time delta (age of the account/card)

Using K-Nearest Neighbor algorithm to fill in empty values

Some values related to the number of crimes committed in 95 and 96 were missing, so we used the K-Nearest Neighbour algorithm to fill the gaps

Determining the correlation between values and removing redundant features

Having multiple values with high correlations between them can have a negative impact on the training of the classification model, for that reason we analyzed the values of both **Pearson** and **Spearman** correlations and removed values with a correlation value above 0.95

Combined all the data into a single dataset

Merged the extracted data into the Clients data frame and followed this by merging the new Clients dataset with the Loans and District datasets. After this, it was also added 3 new columns to the dataset (the age of the client at the time of the loan, the age of the account at the time of the loan, and the age of the card at the time of the loan)

Data Preparation

Data Preparation Pipeline - Main Operations Performed

Converting columns with categorical attributes into integers

Columns like sex, and others that had string attributes were converted into integers to make it possible to use them in algorithms

Remove unnecessary or ineligible information

Remove columns that were either used for merging porpuses (ID values) or contained data that was too sparse to be used

Outliers removal

Having a dataset that combines all the initial datasets, we define a function for removing outliers, based on the 1.5x inter-quartile rule. After that, we apply it to a selection of columns that follow a near-normal distribution - the amount, average_balance_fluctuation_per_month, and avg_balance columns

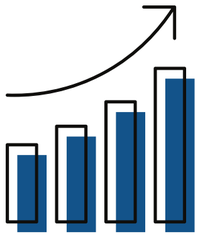
Experimental Setup

Classification Model Pipeline

1. **Standardizing the features** by retrieving the mean and scaling to unit variance (standard score given by z=[x - u]/s, where u is mean and s is std deviation)
2. Addressing the unbalance in the dataset using **oversampling** (we tested SMOTE, SMOTENC, SMOTETOMEK and we also tested undersampling)
3. Used a meta-transformer to **select features** based on their importance weights for each model
4. **Testing different classification models** (SVC, Random Forest, Logistic Regression and Naives Bayes) **with cross-validation**
5. **Hyperparameters are tuned** by doing a cross-validated search over parameter settings
6. **Analyzing the results** - with a main focus on the AUC values but also with the help of plots (precision-recall curve, ROC curve) and the confusion matrix
7. **Selecting our final settings** for classifying the provided test dataset

Results

Results in the Experimental Setup and Kaggle Competition



When testing the data in the experimental context we found that the changes we made reflected very little in the scores and metrics we used to evaluate the performance, so it was hard to determine which algorithm would perform best in the Kaggle competition. The table on the right includes our best results in both the experimental setup and the Kaggle public tests.

Classification Model	Best Results				
	AUC Score Experimental Setup	AUC Score Kaggle Public Tests	AUC Score Kaggle Private Tests	Negative Predictive Value*	False Positive Rate*
Random Forest	0.8696	0.9376	0.8971	89.7%	6.0%
Logistic Regression	0.8941	0.9481	0.8390	98.7%	11.9%

98.7%

Our best Negative Predictive Value used SMOTE-Tomek for balancing the dataset, outlier removal using the 1.5 IQR method, feature selection, hyperparameter tuning, and Logistic Regression as the classification model. Both this value and the associated False Positive Rate exceed the goals set out in our problem definition

* Values calculated in the experimental context

Results

Result Interpretation

Worst Performers

Biggest Impact

Undersampling

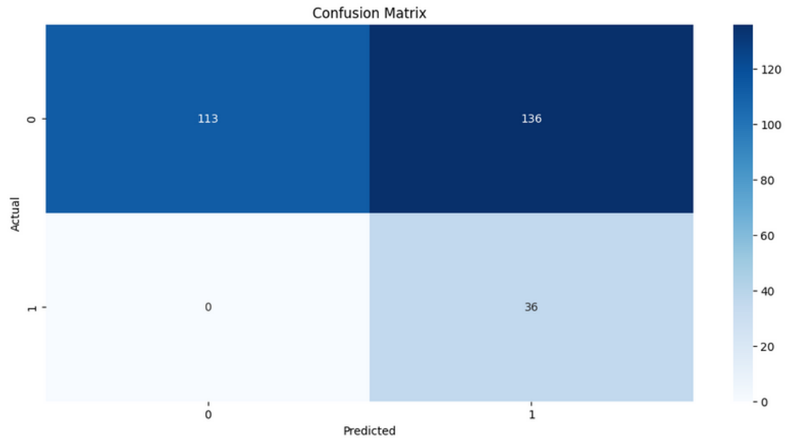
Undersampling proved disappointing, with AUC scores averaging between 60% to 70%, when this method was used

Naives Bayes

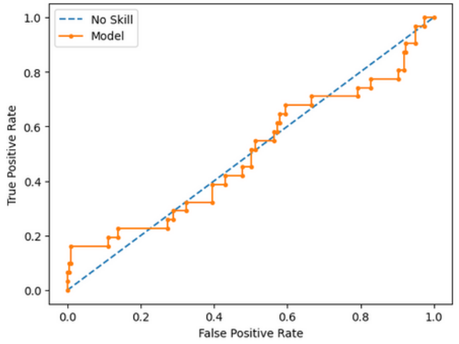
Naives Bayes produced consistently mediocre scores when compared to its counterparts

Hyperparameter Tuning and Feature Selection

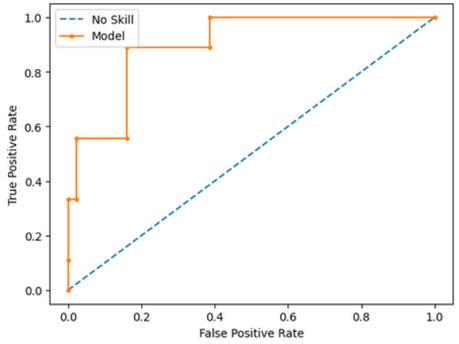
As can be seen from the ROC graphics shown below, when using hyperparameter tuning and feature selection, the quality of the results was higher - there is a clear increase in the value of the area under the curve (AUC)



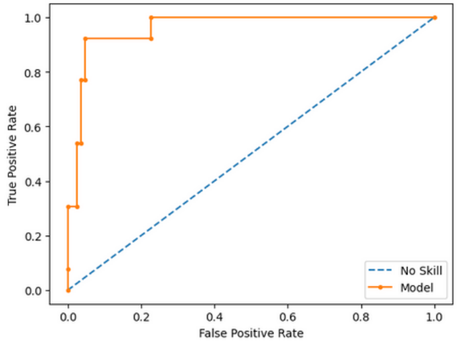
Confusion Matrix - Naives Bayes w/ Undersampling



ROC Curve - Logistic Regression



ROC Curve - Logistic Regression w/ SMOTE and Hyperparameter Tuning



ROC Curve - Logistic Regression w/ SMOTE, Hyperparameter Tuning and Feature Selection

Conclusions, Limitations and Future Work

Conclusions

- Oversampling with SMOTE didn't perform as expected on the private Kaggle tests, even though it had a significant influence on the improvement of the results in the public Kaggle tests and the experimental setup
- Hyperparameter tuning and outlier removal had the most substantial effect on the results
- The goal established at the beginning of this project was accomplished as a higher NPV value was achieved
- We were surprised to find that the Logistic Regression algorithm that had the best AUC score in both the experimental setup and the public Kaggle tests did not perform as well in the private tests. On the other hand, the random forest algorithm had an improvement in performance that was not predicted in the public and experiment setup tests

---

Limitations and Dificulties

- The dataset was fragmented and not very useful as only a minority of the clients had loans, and even in this group, the data was unbalanced. In the end, we ended up working with only a number around 300 entries
- The initial description of the dataset was misleading as it contained wrong information

---

Future Work

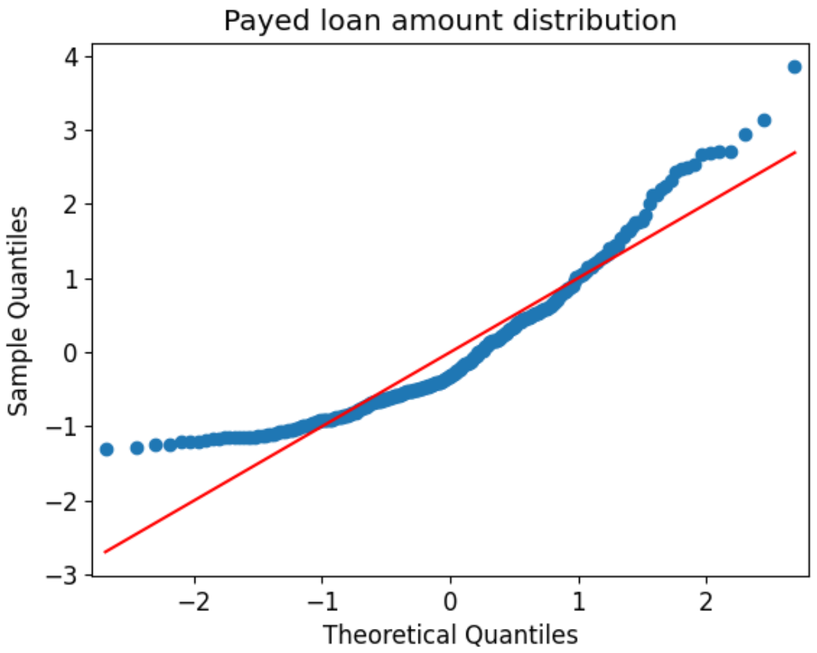
- Explore new classification models
- Closer analysis of the datasets to see if there is any relevant information left that can be extracted

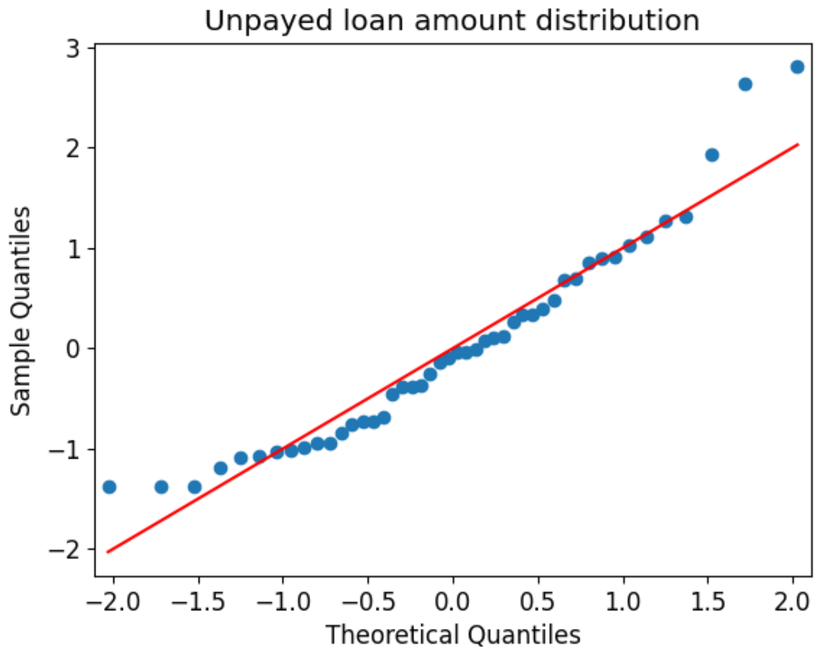
Annexes

Data Understanding

Findings

- Due to the lack of data, the unpaid loan amounts curve is not clearly visible but closely resembles a positive sked distribution curve.
- The Paid loan amounts curve follows a positive sked distribution curve.
- In general, loans with lower amounts are more frequent, regardless of the success of the payment of the loan.

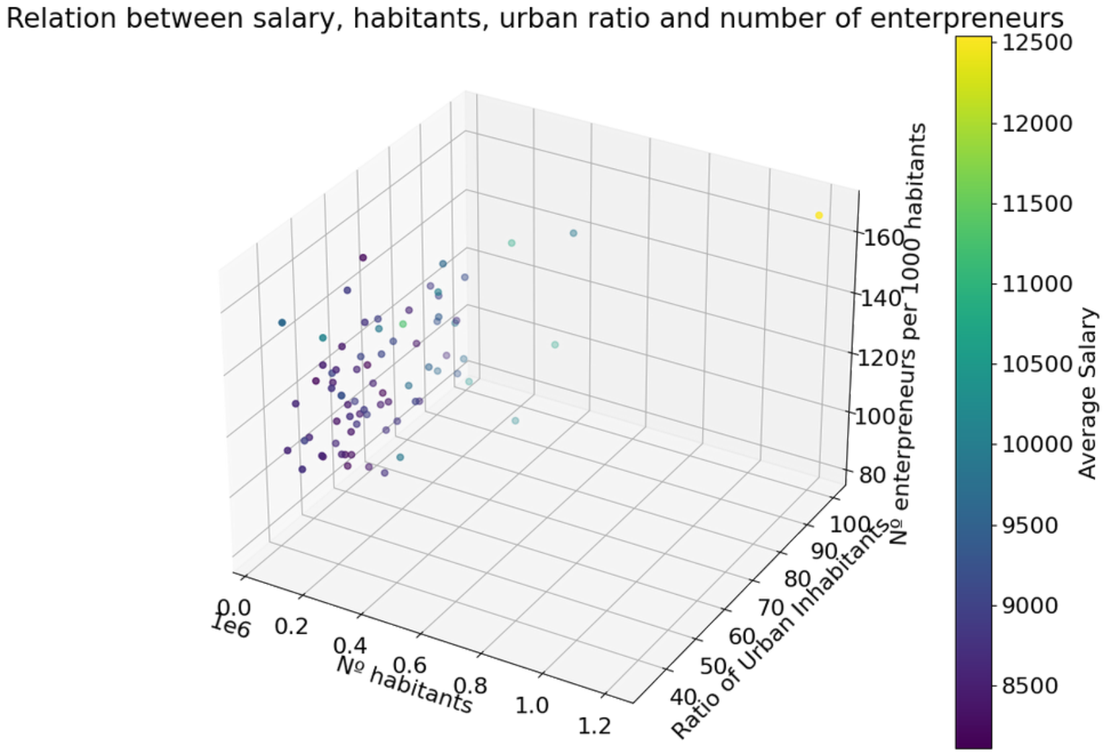




Data Understanding

Findings

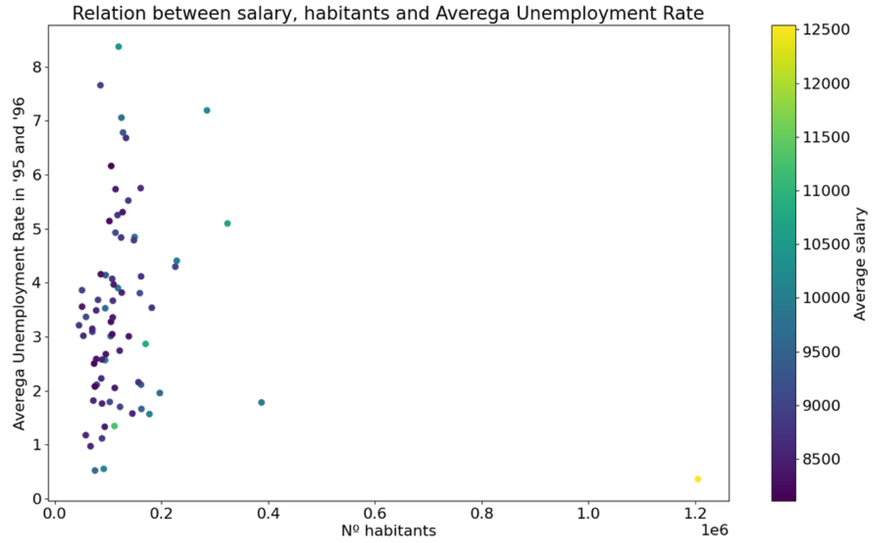
- Less populated areas have significantly lower average salaries.
- The urban ratio slightly affects the average salary: the higher the ratio of urban inhabitants, the higher the average salary.
- The number of entrepreneurs per 1000 habitats in more rural cities with fewer habitats doesn't have a high impact on the average salary.
- One outlier stands out, a city with a high number of inhabitants, urban inhabitants, and entrepreneurs per 1000 habitats, where we can confirm the highest average salary, as expected, which most likely is the capital city.
- Only one larger city is present, while more rural cities are more common.

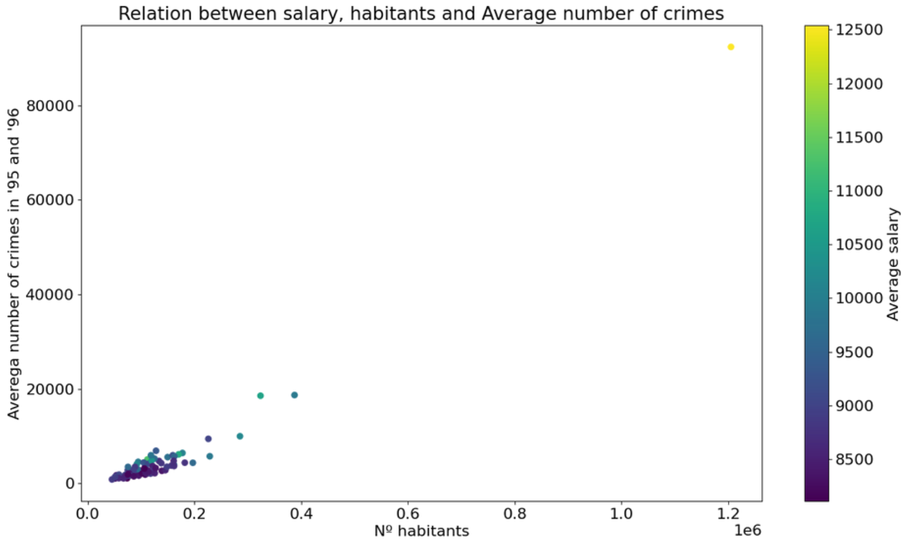


Data Understanding

Findings

- As seen on the last page, cities with a lower number of inhabitants have significantly lower average salaries.
- No correlation was seen between the unemployment rate in cities with fewer inhabitants and the average salary.
- In the top plot, an outlier can be seen on the bottom right, which most likely corresponds to a large urban center with a high population and low unemployment rate, and, as expected, a high average salary value.
- Rural cities with a low number of crimes are the vast majority.
- In the bottom plot, an outlier can be seen on the top right, which most likely corresponds to a large urban center where crimes are more present.





Data Understanding

Findings

- Less populated areas have a significant correlation with lower average salaries
- Significant correlation between client district id and branch district id as in most cases they are the same.
- Data about successive years have a very high correlation
- Absolute values (no of crimes) has a very high correlation with the population
- By analyzing the skewness we verified that most of the data distribution is fairly unsymmetrical
- Looking at the standard deviations allows us to see which data might be represented by its mean or might have outliers.

	code	no. of inhabitants	no. of municipalities with inhabitants < 499	no. of municipalities with inhabitants 500-1999	no. of municipalities with inhabitants 2000-9999	no. of municipalities with inhabitants >10000	no. of cities	ratio of urban inhabitants	average salary	unemployment rate '95	unemployment rate '96	no. of enterpreneurs per 1000 inhabitants	no. of commited crimes '95	no. of commited crimes '96
code	1.000000	0.019858	-0.186831	0.340731	0.318589	0.270787	0.018381	0.022253	-0.250335	0.404195	0.419104	-0.468978	-0.108782	-0.131782
no. of inhabitants	0.019858	1.000000	-0.312467	-0.202190	-0.106207	0.037061	-0.278867	0.463214	0.640440	-0.113837	-0.137509	0.264523	0.978436	0.973129
no. of municipalities with inhabitants < 499	-0.186831	-0.312467	1.000000	0.217136	-0.157365	-0.224381	0.071168	-0.444359	-0.353844	-0.177005	-0.210463	-0.065477	-0.314270	-0.285687
no. of municipalities with inhabitants 500-1999	0.340731	-0.202190	0.217136	1.000000	0.577600	0.066485	0.351456	-0.639956	-0.407941	-0.042703	-0.092587	-0.089230	-0.299030	-0.281315
no. of municipalities with inhabitants 2000-9999	0.318589	-0.106207	-0.157365	0.577600	1.000000	0.032620	0.506122	-0.361284	-0.189635	-0.072288	-0.080391	-0.142781	-0.212998	-0.209699
no. of municipalities with inhabitants >10000	0.270787	0.037061	-0.224381	0.066485	0.032620	1.000000	0.261026	0.285589	-0.032217	0.435900	0.411644	-0.347598	-0.029992	-0.034472
no. of cities	0.018381	-0.278867	0.071168	0.351456	0.506122	0.261026	1.000000	-0.062416	0.279502	-0.095152	-0.133049	-0.063659	-0.325615	-0.307510
ratio of urban inhabitants	0.022253	0.463214	-0.444359	-0.639956	-0.361284	0.285589	-0.062416	1.000000	0.607056	0.190784	0.183726	0.000102	0.450710	0.423680
average salary	-0.250335	0.640440	-0.353844	-0.407941	-0.189635	-0.032217	-0.279502	0.607056	1.000000	-0.068265	-0.096442	0.262970	0.653193	0.634418
unemployment rate '95	0.404195	-0.113837	-0.177005	-0.042703	-0.072288	0.435900	-0.095152	0.190784	-0.068265	1.000000	0.980877	-0.506197	-0.143342	-0.153745
unemployment rate '96	0.419104	-0.137509	-0.210463	-0.092587	-0.080391	0.411644	-0.133049	0.183726	-0.096442	0.980877	1.000000	-0.537565	-0.154203	-0.169144
no. of enterpreneurs per 1000 inhabitants	-0.468978	0.264523	-0.065477	-0.089230	-0.142781	-0.347598	-0.063659	0.000102	0.262970	-0.506197	-0.537565	1.000000	0.336644	0.341629
no. of commited crimes '95	-0.108782	0.978436	-0.314270	-0.299030	-0.212998	-0.029992	-0.325615	0.450710	0.653193	-0.143342	-0.154203	0.336644	1.000000	0.998483
no. of commited crimes '96	-0.131782	0.973129	-0.285687	-0.281315	-0.209699	-0.034472	-0.307510	0.423680	0.634418	-0.153745	-0.169144	0.341629	0.998483	1.000000

	disp_id	client_id	account_id	branch_district_id	num_clients	client_district_id	has_card
disp_id	1.000000	0.999787	0.999983	-0.003577	0.059016	0.052161	0.066191
client_id	0.999787	1.000000	0.999885	-0.003647	0.060825	0.052514	0.067395
account_id	0.999983	0.999885	1.000000	-0.003619	0.059682	0.052254	0.066237
branch_district_id	-0.003577	-0.003647	-0.003619	1.000000	0.002584	0.793821	0.119893
num_clients	0.059016	0.060825	0.059682	0.002584	1.000000	0.010364	0.019547
client_district_id	0.052161	0.052514	0.052254	0.793821	0.010364	1.000000	0.105282
has_card	0.066191	0.067395	0.066237	0.119893	0.019547	0.105282	1.000000

	trans_id	account_id	amount	balance
trans_id	1.000000	0.730371	-0.187568	0.005802
account_id	0.730371	1.000000	0.012000	0.048992
amount	-0.187568	0.012000	1.000000	0.483082
balance	0.005802	0.048992	0.483082	1.000000

```
skewness
trans_id   -0.186464
account_id -0.128467
amount      1.760143
balance     0.965797
account     0.626194
```

```
standard deviation
trans_id              1103419.686515
account_id               3169.7893
date        341 days 03:40:14.670990972
amount                  12454.848929
balance                 24160.74066
```

	disp_id	client_id	account_id	branch_district_id	num_clients	client_district_id	has_card
count	328.000000	328.000000	328.000000	328.000000	328.000000	328.000000	328.000000
mean	7191.899390	7316.789634	5982.085366	36.960366	1.228659	38.112805	0.033537
std	3847.627501	3975.749806	3213.262492	25.242099	0.420610	24.930903	0.180308
min	2.000000	2.000000	2.000000	1.000000	1.000000	1.000000	0.000000
25%	3723.750000	3723.750000	3079.000000	12.750000	1.000000	14.000000	0.000000
50%	7288.500000	7288.500000	6032.000000	38.000000	1.000000	40.000000	0.000000
75%	10255.750000	10563.750000	8564.500000	62.000000	1.000000	61.250000	0.000000
max	13663.000000	13971.000000	11362.000000	77.000000	2.000000	77.000000	1.000000

```
skewness
disp_id            -0.105530
client_id          -0.078805
account_id         -0.099938
branch_district_id -0.048415
num_clients         1.298142
client_district_id -0.122242
has_card            5.205814
```

```
standard deviation
disp_id                          3847.627501
client_id                        3975.749806
account_id                       3213.262492
branch_district_id                 25.242099
creation           369 days 18:19:11.644103016
num_clients                         0.42061
client_district_id                 24.930903
birthdate          4582 days 02:37:16.910259264
has_card                            0.180308
card_issue         345 days 08:31:22.959187544
```

	trans_id	account_id	amount	balance
count	2.449400e+04	24494.000000	24494.000000	24494.000000
mean	2.114311e+06	6064.584021	9253.233049	44398.662027
std	1.103420e+06	3169.789300	12454.848929	24160.740660
min	2.760000e+02	2.000000	0.100000	-3424.600000
25%	1.121973e+06	3115.000000	218.000000	27323.925000
50%	2.287766e+06	6118.000000	4095.000000	39261.450000
75%	3.145645e+06	8625.000000	13192.750000	57014.725000
max	3.682934e+06	11362.000000	74522.000000	193909.900000

```
skewness
code                                    -0.033013
no. of inhabitants                       6.443703
no. of municipalities with inhabitants < 499    0.545120
no. of municipalities with inhabitants 500-1999    0.424873
no. of municipalities with inhabitants 2000-9999    1.224662
no. of municipalities with inhabitants >10000    0.756958
no. of cities                           -0.067662
ratio of urban inhabitants               0.640568
average salary                           1.761712
unemployment rate '95                    0.650900
...
unemployment rate '96                    0.692654
no. of enterpreneurs per 1000 inhabitants    0.593651
no. of commited crimes '95               7.355724
no. of commited crimes '96               7.699699
```

```
standard deviation
code                                    22.469044
no. of inhabitants                  140241.845056
no. of municipalities with inhabitants < 499    33.118049
no. of municipalities with inhabitants 500-1999    13.025747
no. of municipalities with inhabitants 2000-9999    4.030987
no. of municipalities with inhabitants >10000    1.024435
no. of cities                            2.477448
ratio of urban inhabitants              16.221325
average salary                         806.025408
unemployment rate '95                    1.665144
...
unemployment rate '96                    1.896729
no. of enterpreneurs per 1000 inhabitants    16.752985
no. of commited crimes '95           10150.636376
no. of commited crimes '96           11565.219310
```

Data Processing

Data Quality Assessment

Accuracy

We found the dataset to be accurate. That being said, we didn't have a lot of information in order to assess this dimension, so our evaluation is merely based on our perception of reality and our expectations for this type of dataset

Completeness

Some values related to the number of crimes committed in 95 and 96 were missing

Consistency

We determined the dataset is consistent given that all the information related to loans could be matched with other information such as transactions, cards, etc by simply matching identifying columns (loan_id, client_id, ...)

Timeliness

Both the training dataset and the test dataset provided were from sequential periods of time, so timeliness in the analysis was ensured

Validity

Most of the data in the datasets followed business rules and fulfilled our expectations with exception of the age of the clients who took out loans. Some clients were underaged according to current legal standards

Uniqueness

There were no duplicate values in any of the datasets.

Data Processing

Redundancy, Missing Data, Outliers and Feature Selection

Determining the correlation between values and removing redundant features

Having multiple values with high correlations between them can have a negative impact on the training of the classification model, for that reason we analyzed the values of both **Pearson** and **Spearman** correlations for features that followed a near-normal distribution, and removed values with a correlation value above 0.95

Feature Selection

Throughout the data mining process, we did feature selection at two separate stages, one of them was here, by removing features with an elevated correlation value **(filter-based)**, and another was during the classification process **(wrapper-based)**, using a meta-transformer to select features based on their importance weights (with a Linear Support Vector Classification estimator)

Using K-Nearest Neighbor algorithm to fill in empty values

Some values related to the number of crimes committed in 95 and 96 were missing, so we used the K-Nearest Neighbour algorithm to fill the gaps

Outliers removal

Having a dataset that combines all the initial datasets, we define a function for removing outliers, based on the 1.5x inter-quartile rule. After that, we apply it to a selection of columns that follow a near-normal distribution - the amount, average_balance_fluctuation_per_month, and avg_balance columns

Data Processing

Sampling, Data Transformation, Data Integration and Feature Engineering

Removing information that has no relation to loans

Although we were given a substantial amount of data, a lot of it was about clients that had no loans nor any relation to the loan bearer

Extract the client's birth date and gender from the "birth_number"

The column birth_number contained information regarding both the birthdate and the client's gender

Extracted new parameters from several of the datasets

Created columns like crime rate, recent balance, household payments, insurance payments, sanction payment counter, to name just a few

Converted dates into time intervals

Since most classification models don't accept a date as an input type, we converted dates like, the client's birthdate, the account creation date or the date of issuance of the credit cards into a time delta (age of the account/card)

Converting columns with categorical attributes into integers

Columns like sex, and others that had string attributes were converted into integers to make it possible to use them in algorithms

Remove unnecessary or ineligible information

Remove columns that were either used for merging porpuses (ID values) or contained data that was too sparse to be used

Combined all the data into a single dataset

Merged the extracted data into the Clients data frame and followed this by merging the new Clients dataset with the Loans and District datasets. After this, it was also added 3 new columns to the dataset (the age of the client at the time of the loan, the age of the account at the time of the loan, and the age of the card at the time of the loan)

Data Processing

Sampling, Data Transformation, Data Integration and Feature Engineering

Extracting Data from transactions

- From the transactions table, it was possible to calculate the following monthly information about the clients: Average variation of the balance per month (average_balance_fluctuation_per_month) and the common amount credited on specific days in 80% of the months registered (salary). These two values are a good indicator of the financial stability of the client.
- We were also able to extract information regarding all client transactions like the number of interests credited (num_interests_credited), the total number of transactions (num_transactions), the ratio of credit-type transactions (ratio_credits), average balance (avg_balance), minimum balance (min_balance), maximum balance (max_balance), etc. As one can see these columns focus on giving us an overall view of the activities of the clients so one can know how a client managed his finances in the past.

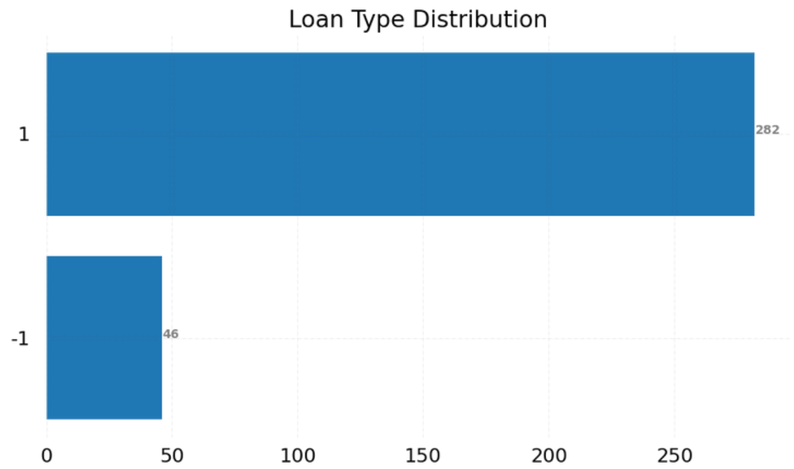
Dealing with Data from transactions

- Now that we have data that is representative of all the transactions of one client we can ditch the transaction table and merge the extracted information into the clients dataframe.
- We thought these new columns that were added to clients are good representatives of the financial capacity and stability of the client which would influence the prediction of granting or not a load.

Data Processing

Imbalance Treatment

As part of the exploratory analysis, we discovered a significant imbalance in the loan outcome distribution, with a majority of loans being paid and a small amount defaulting



Undersampling

- Random Undersampling - undersample the majority class by randomly picking samples

Oversampling

- SMOTE (Synthetic Minority Oversampling Technique) - draws a random sample from the minority class, identifies the k nearest neighbors and takes one of those neighbors, identifies the vector between the current data point and the selected neighbor, multiplies the vector by a random number between 0 and 1 and adds this to the current data point to obtain the synthetic data point
- SMOTE-NC (Synthetic Minority Over-sampling Technique for Nominal and Continuous) - unlike SMOTE, SMOTE-NC for dataset containing numerical and categorical features

Combined

- SMOTE-Tomek (SMOTE with Tomek Links) - Combine over and under-sampling using SMOTE and Tomek links

Predictive Task

Overview

Defining the Problem Type and Algorithms

Supervised Learning

A type of machine learning in which a model is trained on labeled data, meaning that the correct output is provided for each example in the training set. The model uses this training data to make predictions on new, unseen examples

Defining the Algorithms

We were limited by both the time we had to complete this task and the limit of 1 submission per day imposed for submitting predictions on Kaggle. For that reason we decided to limit our analysis to four algorithms:

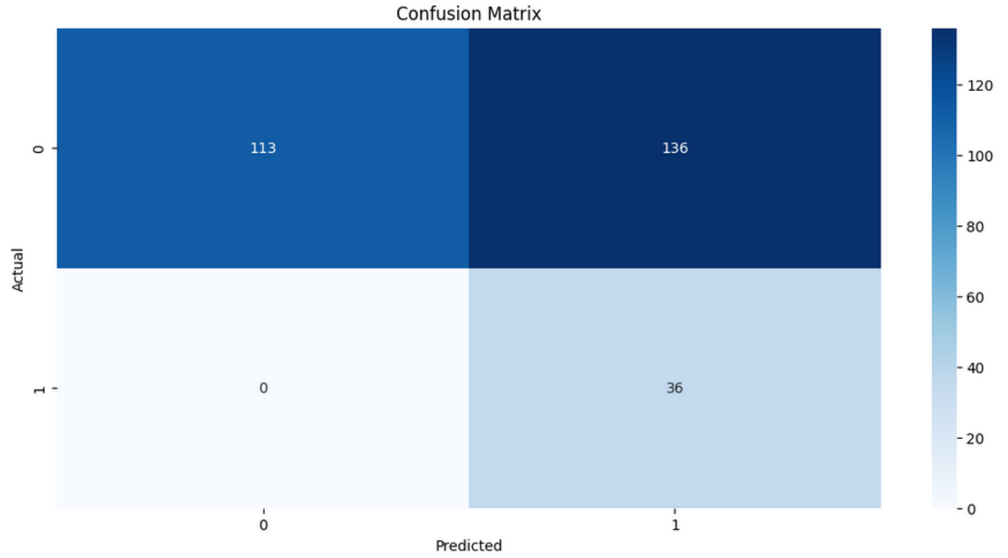
- Naive-Bayes
- Support Vector Machines
- Logistic Regression
- Random Forest

Predictive Task

Naïve Bayes

The algorithm

- Naive Bayes is a type of classification algorithm based on the Bayes theorem, which states that the probability of an event is equal to the prior probability of the event multiplied by the likelihood of the event given certain conditions. In the case of the naive Bayes algorithm, the "event" we are interested in is the class of a given data point, and the "conditions" are the attributes of that data point
- The algorithm is called "naive" because it makes a very strong assumption about the data: that all of the attributes are independent of one another. This means that the presence or absence of a particular attribute does not affect the presence or absence of any other attribute. While this assumption is often not true in practice, the naive Bayes algorithm still performs well in many real-world applications, for that reason we decided to give it a try



Confusion Matrix - Naives Bayes

Results and Interpretation

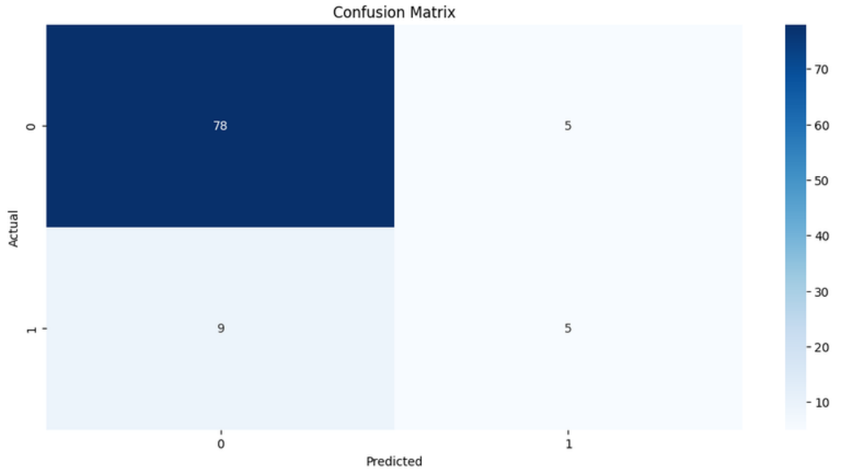
Naive-Bayes produced consistently mediocre scores when compared to its counterparts. Besides the assumption of independence, which might have played a big part in the negative results, since some of the parameters of the dataset are directly intertwined, another weakness of the naive Bayes algorithm is that it can be sensitive to noisy or irrelevant attributes in the data. Because the algorithm calculates probabilities for each attribute independently, the presence of noisy or irrelevant attributes can have a disproportionate effect on the final predicted class

Predictive Task

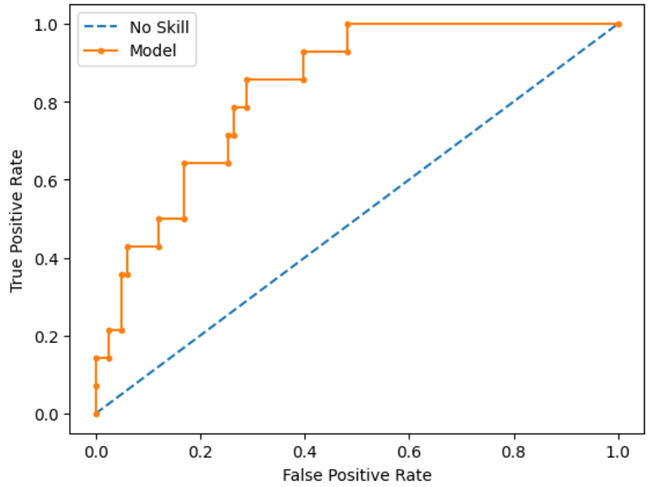
Random Forest

The algorithm

- The Random Forest is an ensemble learning algorithm that is used for classification and regression. It is called a "random forest" because it trains multiple decision trees on random subsets of the data and then averages the predictions of these trees to make a final prediction
- The decision trees in a random forest are trained using a technique called bagging, which involves training each tree on a different random subset of the data. This helps to reduce overfitting, which is a common problem in decision trees that can cause them to perform poorly on new data
- To make a prediction for a new data point using a random forest, the algorithm first feeds the data point to each of the decision trees in the forest. Each tree then makes a prediction, and the final prediction of the random forest is calculated by averaging the predictions of all the trees



Confusion Matrix - Random Forest



ROC Graph - Random Forest

Results and Interpretation

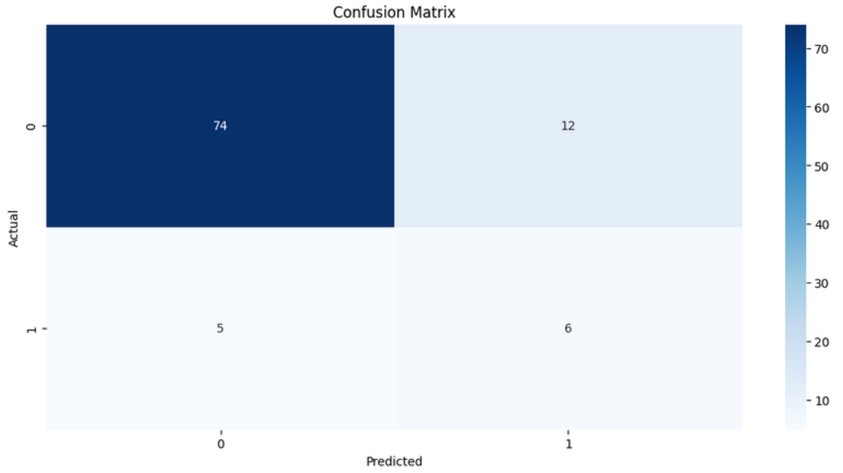
The RF algorithm produced some of the best results we had with a 0.87 AUC Score in our experimental setup tests and 0.94 and 0.90 in our public and private Kaggle tests, respectively. The RF can be considered a blackbox model since it's hard to interpret the predictions it makes (RF is an ensemble of many decision trees, so it can be hard to understand how the individual trees are contributing to the final prediction). This said the positive results we encountered might be attributed to the fact that RF doesn't make any assumptions about the data or its distribution. Hence it generally requires minimal data transformations. Also, RF makes use of random subsets of features and hence it can perform quite well with a high dimensional dataset (datasets with many features), like the one we have

Predictive Task

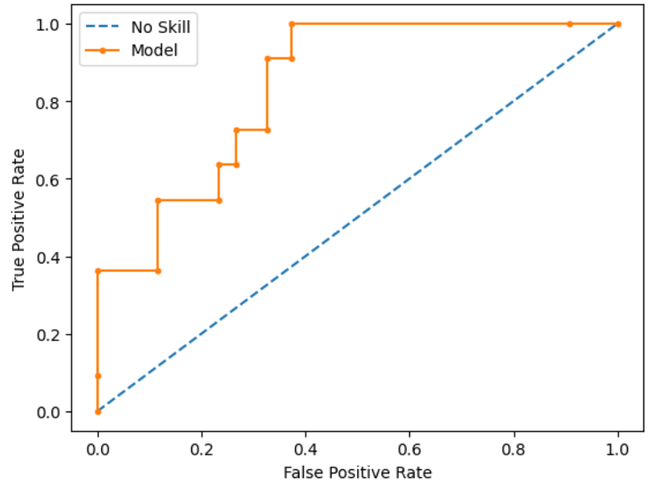
Support Vector Machine

The algorithm

- Support Vector Machines are a type of supervised learning algorithm that can be used for classification or regression. The goal of an SVM is to find the line or hyperplane that best separates the data points in a dataset into different classes. This line or hyperplane is called the decision boundary, and the points that are closest to the decision boundary are called support vectors
- To find the decision boundary, SVMs use a concept called the margin, which is the distance between the decision boundary and the closest data points. The goal of the SVM algorithm is to find the decision boundary that has the greatest margin, which helps to improve the generalization of the model to new data.
- Once the decision boundary has been found, new data points can be classified by determining which side of the boundary they fall on.



Confusion Matrix - SVM



ROC Graph - SVM

Results and Interpretation

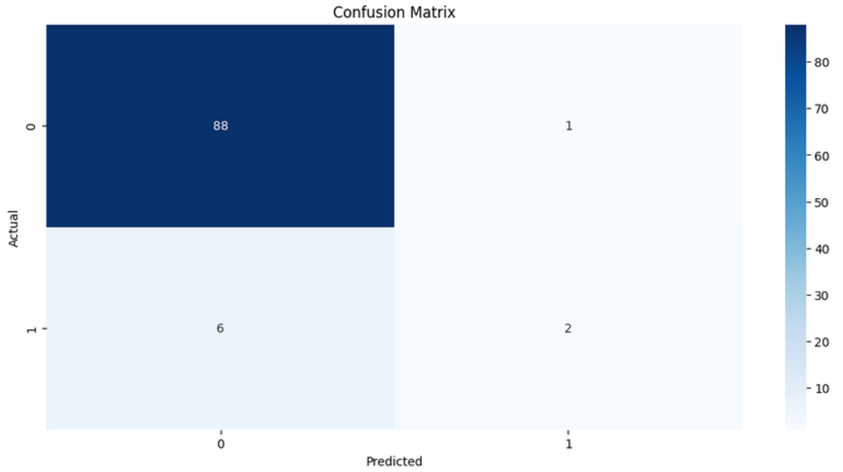
Although it produced great results, with an AUC score of 0.86 in our experimental setup tests and 0.89 and 0.84 in the Kaggle public and private tests respectively, the SVM algorithm slightly underperformed the Random Forest and Logistic Regression algorithms. Since the dataset we worked with is relatively small and the difference in scores was a matter of a few decimal points, no definitive conclusions may be drawn from these values.

Predictive Task

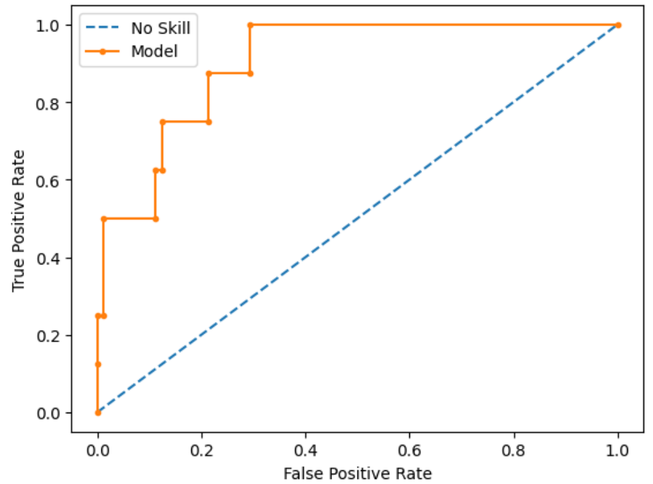
Logistic Regression

The algorithm

- Logistic regression is a type of supervised learning algorithm used for classification. It is called "logistic regression" because it uses a logistic function to model the relationship between the dependent variable and one or more independent variables. The logistic function is a mathematical function that takes in an input and produces an output value between 0 and 1, which can be interpreted as a probability
- To train a logistic regression model, the algorithm uses an optimization procedure to find the coefficients of the independent variables that best predict the dependent variable. Once the model has been trained, it can be used to make predictions for new data points by applying the logistic function to the coefficients and independent variables of the new data point



Confusion Matrix - Logistic Regression



ROC Graph - Logistic Regression

Results and Interpretation

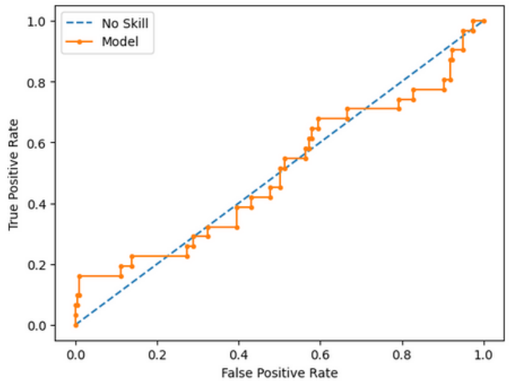
The LR algorithm produced some of the best results we had with a 0.89 AUC Score in our experimental setup tests and 0.95 and 0.84 in our public and private Kaggle tests, respectively. This might be attributable to our data-preparation work, namely the removal of variables with high correlation value. One of the main weaknesses of LR is that it is sensitive to the presence of irrelevant or highly correlated independent variables in the data. Because the model uses a linear combination of the independent variables to make predictions, the presence of irrelevant variables or strong correlations between variables can lead to unstable and misleading coefficients. This can cause the model to make poor predictions on new data. By removing those cases, we might have improved the model's performance

Predictive Task

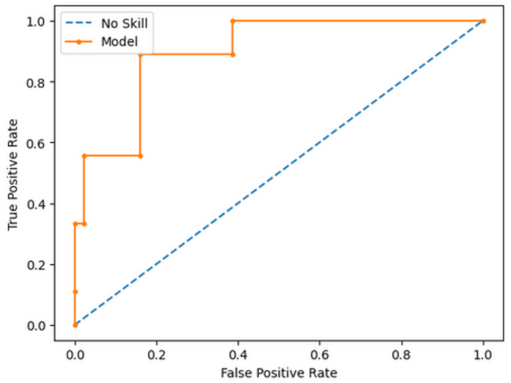
Parameter Tuning, Wrapper-based Feature Selection

Hyperparameter Tuning

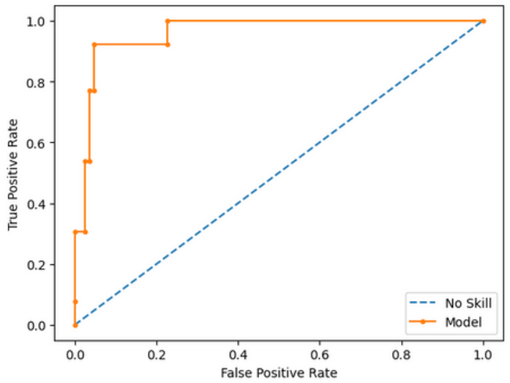
- Hyperparameter tuning involves training the model multiple times with different values of the hyperparameters and then evaluating the performance of the model on a validation dataset
- The goal of hyperparameter tuning is to find the values of the hyperparameters that result in the best performance of the model on the validation data. To evaluate performance, we used the AUC value
- There are several different methods for hyperparameter tuning, including grid search, random search, and Bayesian optimization. In the end **opted for Random Search**



ROC Curve - Logistic Regression



ROC Curve - Logistic Regression w/ SMOTE and Hyperparameter Tuning



ROC Curve - Logistic Regression w/ SMOTE, Hyperparameter Tuning and Feature Selection

Wrapper-based Feature Selection

- As mentioned earlier we performed two types of feature selection, filter-based, during the data preparation phase and wrapper-based during the predictive phase
- In wrapper methods, the feature selection process is based on the specific machine learning algorithm that we are trying to fit on a given dataset. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion
- We used a meta-transformer from SKLearn (SelectFromModel) to select features based on their importance weights for each model we used (Random Forest, Logistic Regression, and SVM)

Predictive Task

Performance Estimation - Train vs Testing

Cross-validation

- Cross-validation is a method for evaluating the performance of a machine-learning model. It is called "cross-validation" because it involves splitting the training data into multiple "folds" and training the model on different combinations of these folds
- The main goal of cross-validation is to estimate the performance of the model on new data, by measuring its performance on different subsets of the training data. This is important because the performance of a machine learning model can vary depending on the data that it is trained on. By using cross-validation to evaluate the model's performance on multiple different subsets of the data, it is possible to get a more accurate estimate of how the model will perform on new data
- We **performed cross-validation with all predictive models**

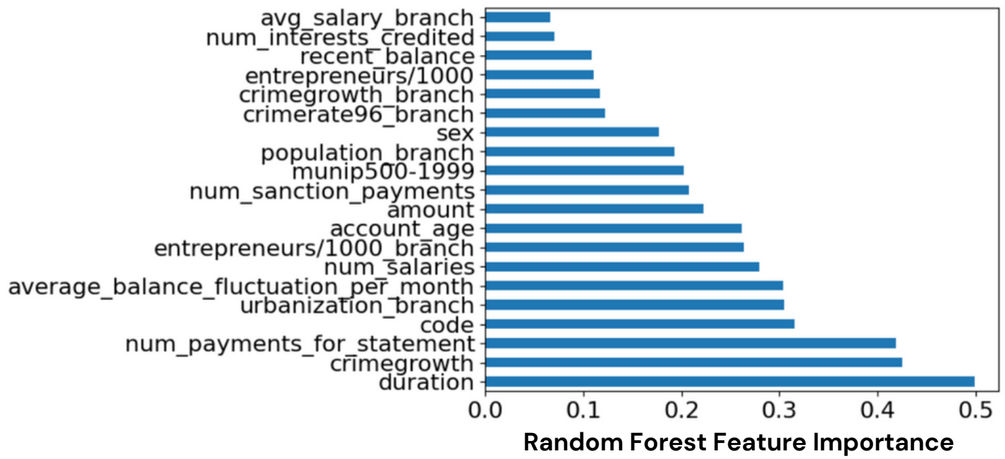
Stratified K-Folds

The splitting strategy we opted for was Stratified K-Folds cross-validation. In stratified cross-validation, the data is split into folds in such a way that the proportions of different classes in the data are preserved in each fold. This can be important for imbalanced datasets, where some classes are much more common than others, because it ensures that the model is trained and evaluated on a representative sample of the data. Since our dataset is marked by a significant imbalance in the type of loan (more loans labeled as defaults, than as paid loans), this splitting strategy felt like the natural choice

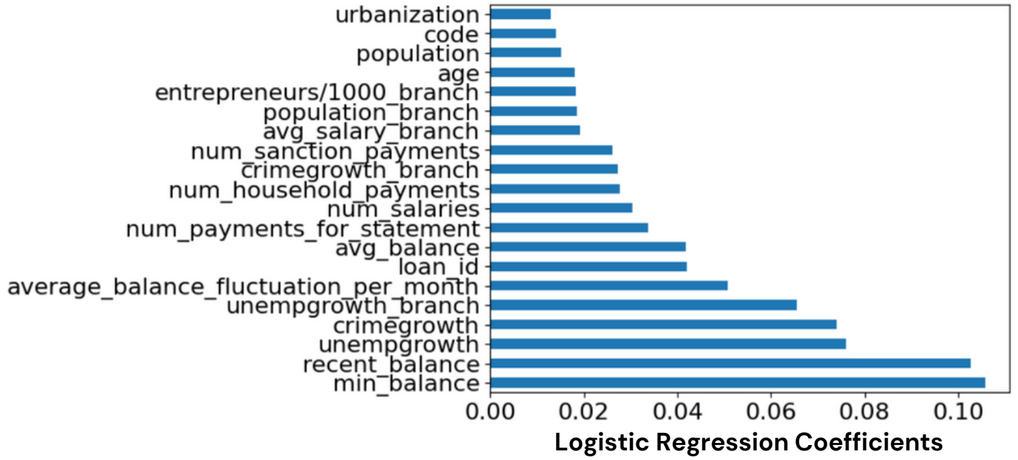
Predictive Task

Feature Importance and Model Improvement

For feature importance we analysed the results from our first runs of logistic regression and random forest. Putting it into use, the results of a l1 Linear SVC were applied to on hypertuning for feature selection.



Random Forest Feature Importance



Logistic Regression Coefficients

In our effort to improve our model (taking into account kaggle results), we sought to maximize the value of our hypertuning models: In this proccess we elected to:

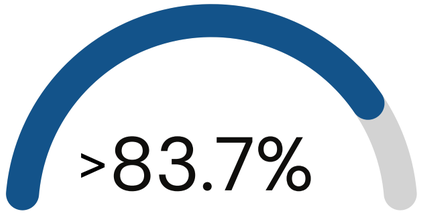
- Alter the target score, from F1 to AUC
- Apply hypertuning to SMOTE, rather than just the model.
- Switch SMOTE for SMOTETomek
- Apply outlier removal
- Reduce limits for outlier removal
- Attempt undersampling rather than smote (weaker results)

Predictive Task

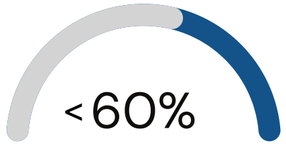
Performance Estimation - Performance Measures and Result Analysis

Defined Goals

We were able to reach our initially defined goals with 3 of the algorithms: Random Forest, SVM and Logistic Regression



Negative Predictive Value (NPV)*



False Positive Rate (FPR)*

General Data Mining Metrics

After reaching our defined goals, we started using other metrics to rank the algorithms from best to worse. We used metrics such as F1 score, Accuracy, AUC (Area Under the Curve), with a focus on AUC. We also used some graphs to help us visualize the results: ROC curve and Confusion Matrix

Other Metrics

Apart from the metrics mentioned previously, we measured the time it took to run each model. Since the dataset we worked with was relatively small, we didn't have to worry much about this metric, but in a bigger more realistic dataset, it would definitively be a measure to consider

Evaluation Metric	Classification Model		
	Random Forest	Logistic Regression	SVM
AUC Score Experimental Setup (Cross-Validation Mean)	0.8696	0.8941	0.8648
Experimental Setup Standard Deviation	4.35%	4.78%	3.63%
Negative Predictive Value*	89.7%	98.7%	97.4%
False Positive Rate*	6.0%	11.9%	10.4%
AUC Score Kaggle Public Tests	0.9376	0.9481	0.8945
AUC Score Kaggle Private Tests	0.8971	0.8390	0.8436
Runtime	113s	15s	8.5s

As can be seen above, AUC isn't always the best predictor of a model's quality. Despite being the model with the best AUC in our experimental setup, the RF algorithm was outperformed by the Logistic Regression when it comes to the business goals metrics (NPV and FPR), as well as in terms of runtime.

* Considering loan defaults as positives and payed loans as negatives

Descriptive Task

Overview

Objectives and Algorithms

Objectives - Profiling

To identify patterns and relationships in the data we use Clustering, a type of unsupervised learning in machine learning, that groups similar items together into clusters, making it possible to analyze and profile the different types of clients present in the final dataset. For this we used the socio-demographic data of the user (client and district datasets).

Defining the Algorithms

To complete these tasks we used the three different Clustering algorithms with distinct approaches to the clustering problem:

- K-Means
- Agglomerative Clustering
- DBSCAN

Descriptive Task

Algorithms

K-Means	DBSCAN	Agglomerative Clustering
K-means is a centroid-based clustering algorithm that is used to group similar items into clusters. This algorithm is based on the principle of dividing a dataset into a specified number of clusters, and then iteratively refining the clusters by reassigning items to the cluster that is closest to their mean.	DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm. It is designed to find clusters of arbitrary shape in a dataset by identifying high-density regions of the data.	Agglomerative Clustering is a hierarchical clustering algorithm that uses a bottom-up approach to build a hierarchy of clusters. In this algorithm, each data point is initially considered to be its own cluster, and then the clusters are iteratively merged together based on some measure of similarity or distance. This process continues until all data points are in the same cluster, or until a specified number of clusters is reached.

Descriptive Task

Performance Evaluation Criterea

To evaluate the performance of the different iterations of the clustering algorithms we used:

Silhouette Score

The measure of how well each data point is assigned to its cluster. It ranges from -1 to 1, with a high value indicating that the data point is well-matched to its cluster and a low value indicating that it is poorly matched.

Inertia

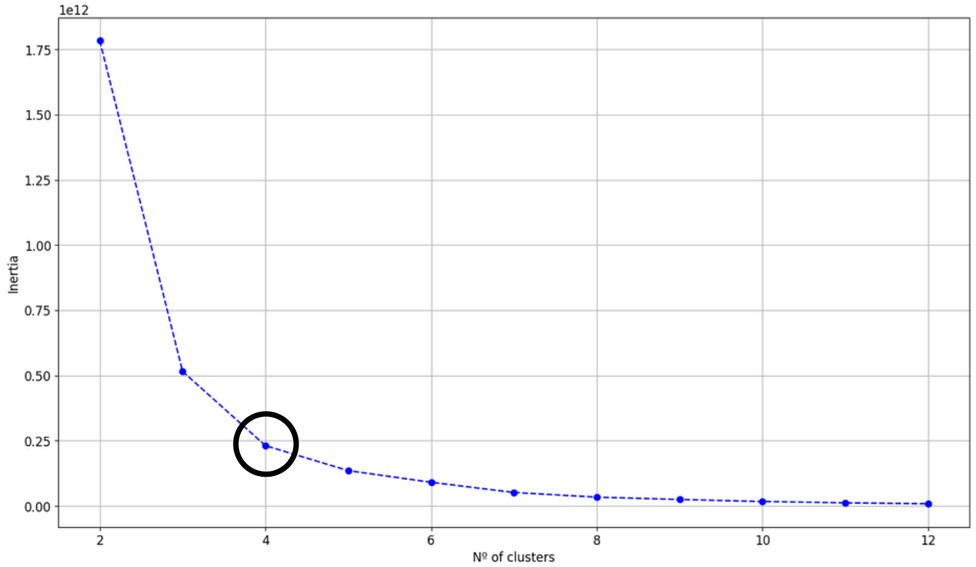
The measure of the sum of the squared distances between each data point and the center of its cluster. It is used to evaluate the quality of a clustering algorithm. In general, a clustering algorithm with lower inertia is considered to be better, because it means that the data points are closer to the centers of their clusters.

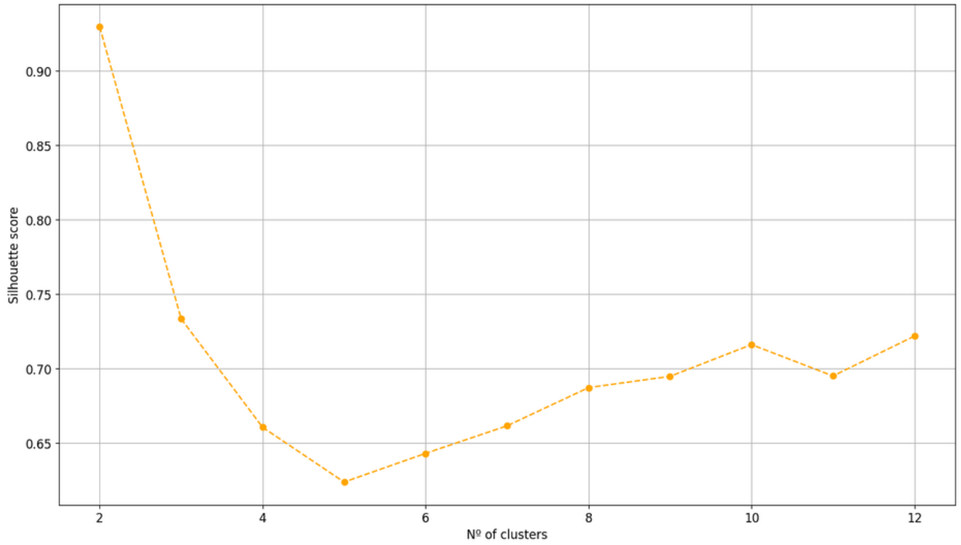
Descriptive Task

Parameter Tuning and Performance Evaluation

K-Means

- The k-means algorithm requires that the number of clusters to group up the data has to be specified.
- To optimize the algorithm, various iterations were run varying the number of clusters.
- On the right, we can see the variation of Inertia and Silhouette scores with the increase in the number of clusters.
- By applying the Elbow heuristic, we can conclude that regarding Inertia, the ideal number of clusters would be 4.
- Even though the silhouette score of the iteration with 4 clusters is not the best performing, we decided to stay with this option since the elbow heuristic is a method for choosing the optimal number of clusters.



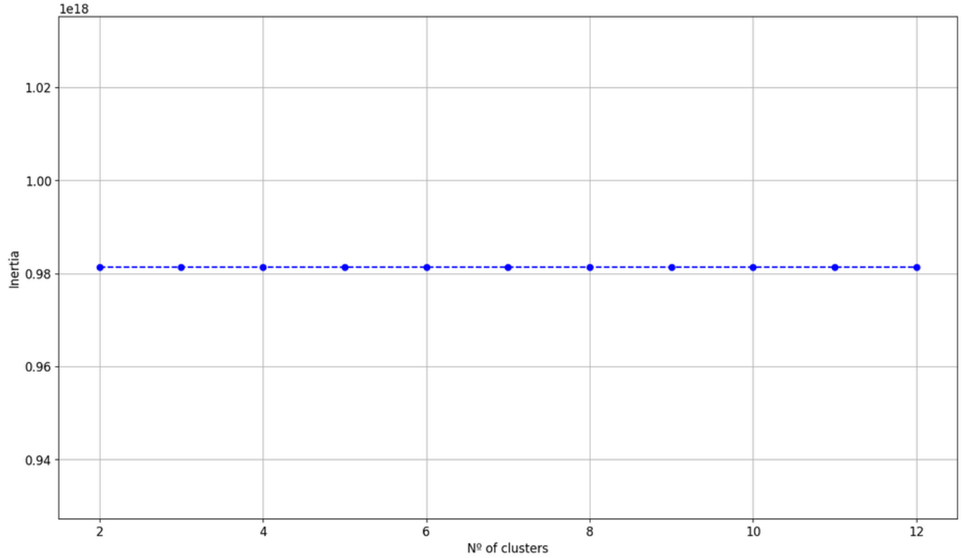


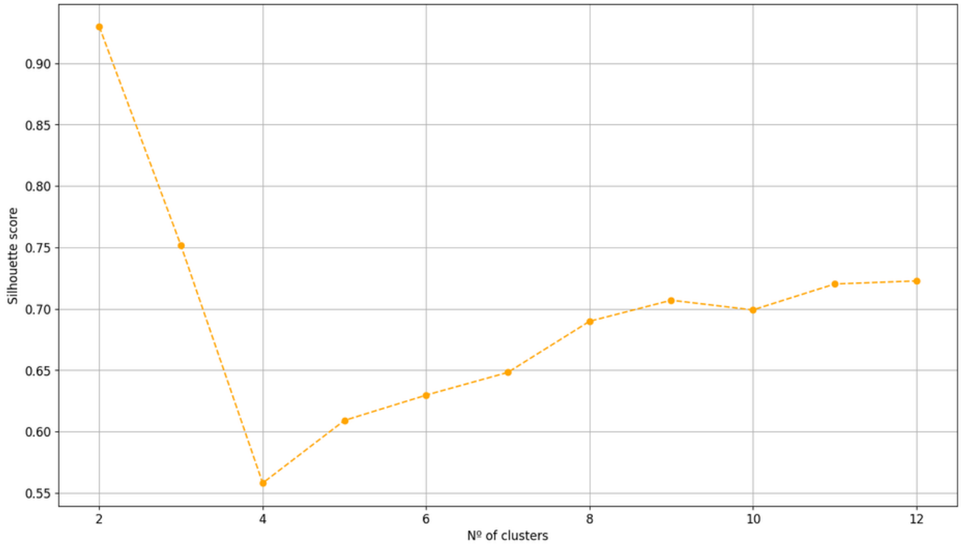
Descriptive Task

Parameter Tuning and Performance Evaluation

Agglomerative Clustering

- Like the k-means algorithm, Agglomerative Clustering requires that the number of clusters to group up the data has to be specified and the same procedure was taken.
- The Inertia value remained relatively constant across all iterations, the reason why this happened is related to the clusters at each level being similar in size and compactness. And since the value is relatively constant, the Elbow heuristic can't be applied and the ideal number of clusters to use is inconclusive.
- Regarding the Silhouette score, the iteration with the number of clusters equal to 2 was the one that outperformed, so we can conclude that it is the best parameter value for this algorithm.



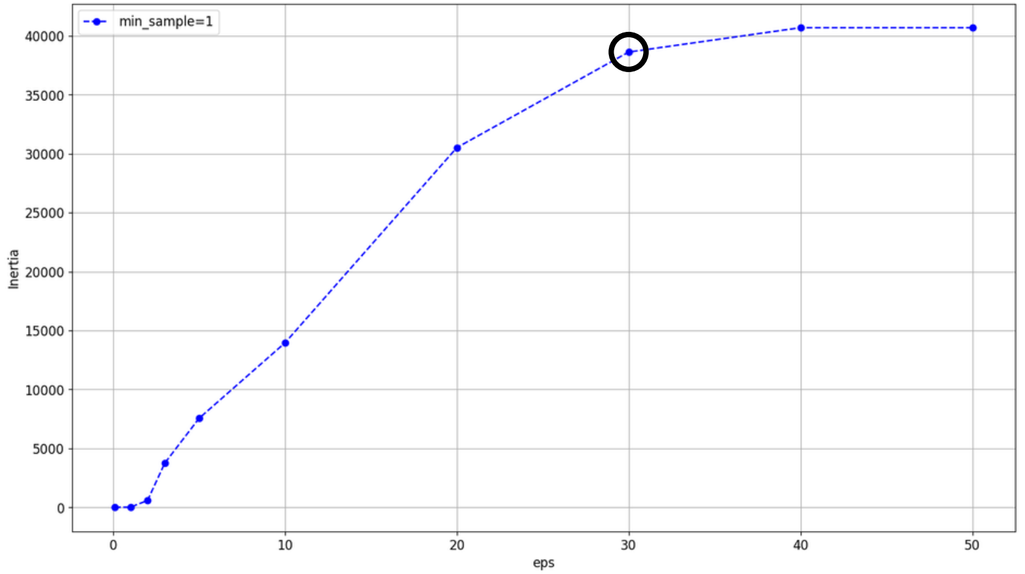


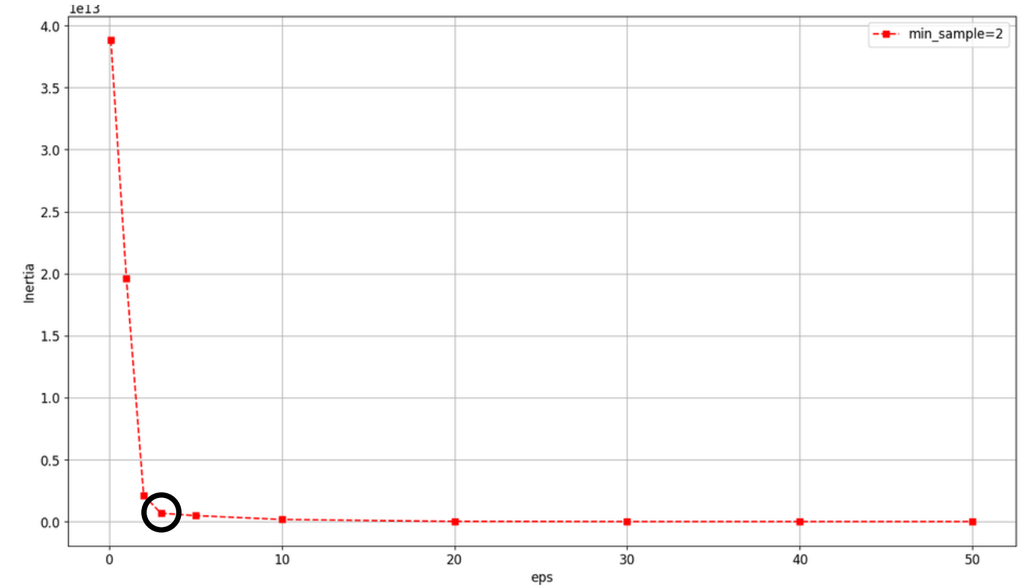
Descriptive Task

Parameter Tuning and Performance Evaluation

DBSCAN

- DBSCAN algorithm uses the parameters 'eps', which corresponds to the maximum distance between two samples for one to be considered as in the neighborhood of the other, and 'min_samples', which corresponds to the number of samples (or total weight) in a neighborhood for a point to be considered as a core point.
- Min_sample values 1 and 2 were chosen since other values would generate a single cluster system.
- For the min_sample=1, unlike other Inertia curves, this one we can verify logarithmic ascension, having the optimal point to satisfy the Elbow heuristic at eps=30.
- For the min_sample=2, the Inertia - eps function has a noticeable steep decline, but with careful selection of eps values, we could find the optimal point to satisfy the Elbow heuristic, which corresponds to eps equal to 3.
- Comparing both min_sample cases, based on Inertia, min_sample=1 (with eps=30) is the overall better parameter because it has overall significantly lower values.



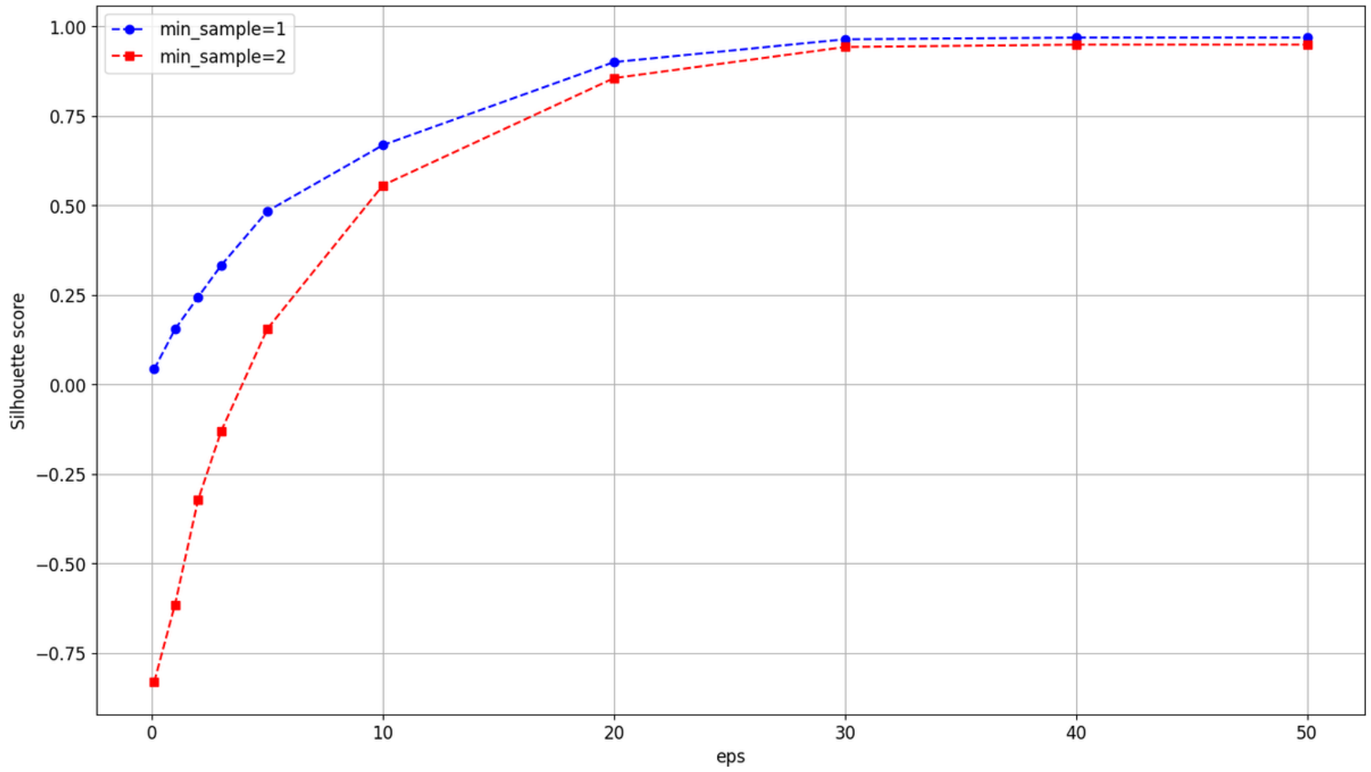


Descriptive Task

Parameter Tuning and Performance Evaluation

DBSCAN

- Regarding Silhouette score, both min_sample values have a similar progression along the eps values.
- The min_sample=1 curve has again overall a better Silhouette score across all eps values, starting with a better value but converging to the same +/- 0.95 Silhouette score alongside min_sample=2.



Descriptive Task

Performance Evaluation

After running the best iterations of each algorithm we concluded the following:

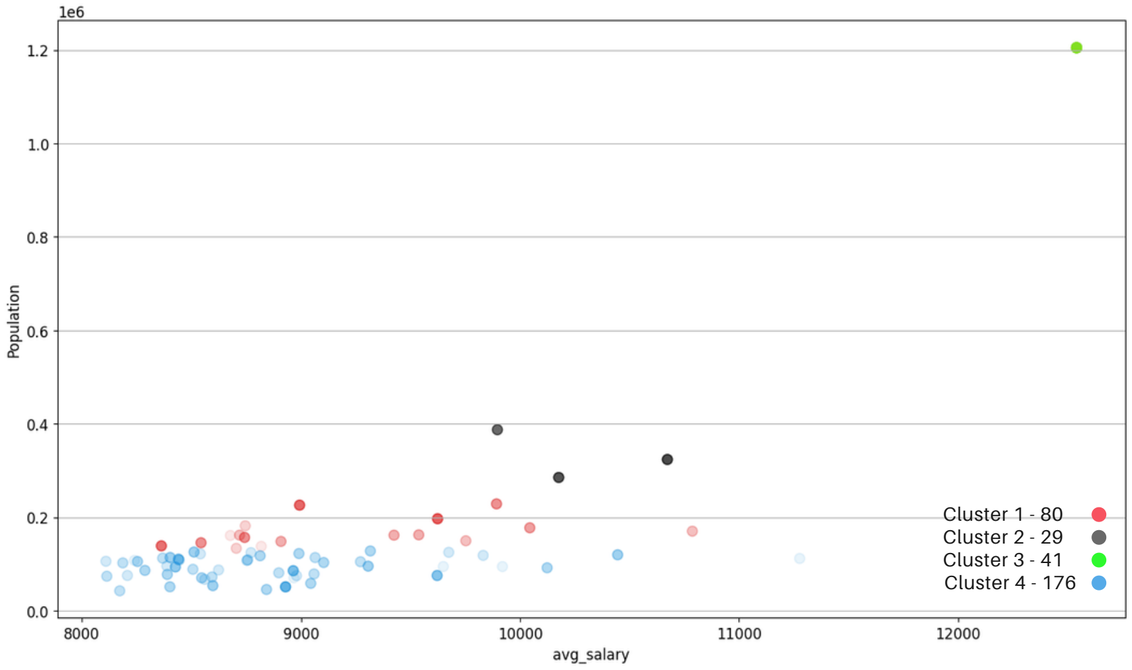
K-Means	DBSCAN	Agglomerative Clustering
• Davies Bouldin Score = 0.3927 The visual results of the K-Means were coherent and could be seen a clear distinguishment between all clusters, being the population the clear divider between them.	• Davies Bouldin Score = 0.0176 DBSCAN doesn't produce a fixed number of clusters, so the result ended up being too overfitted, 20+ clusters were generated. This ended up being the worst result.	• Davis Bouldin Score = 0.0556 Agglomerative Clustering, like K-Means, ended up being the best results, only two clusters created and both visualy coherent and well noticeable, defining two clear client profiles.

Descriptive Task

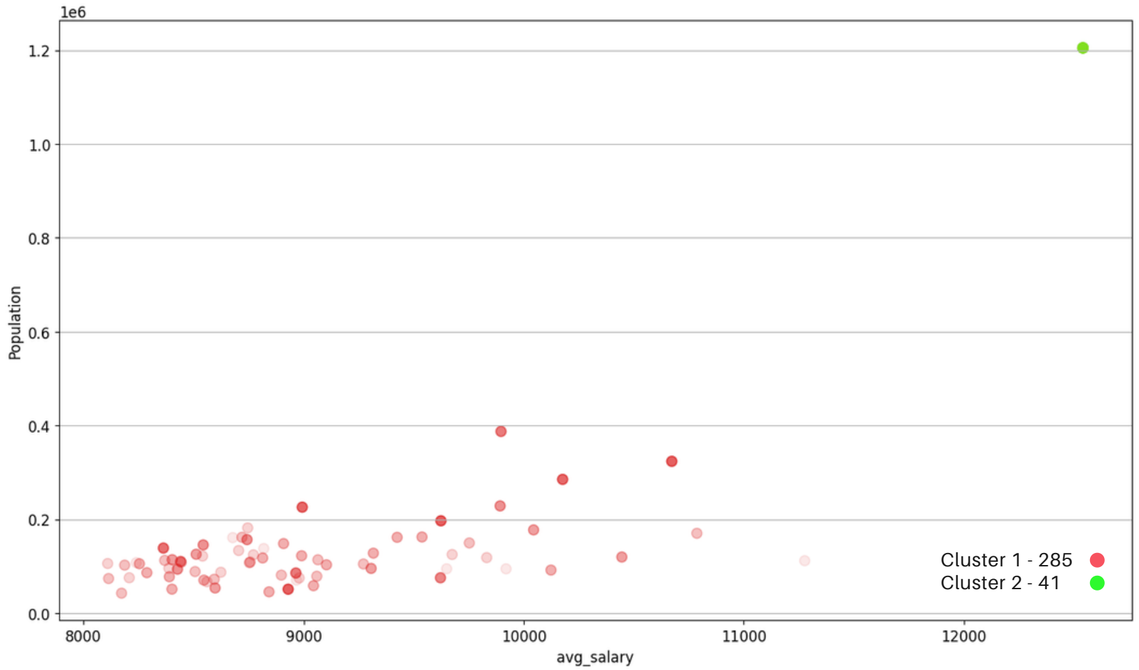
Results

The following scatter plots show how both algorithms distribute people. In either of them where the population is more condensed the distributions are the same and looking at the other tree clusters on the left, we can see that their distribution is very similar to the second cluster on the right

K-Means with 4 clusters

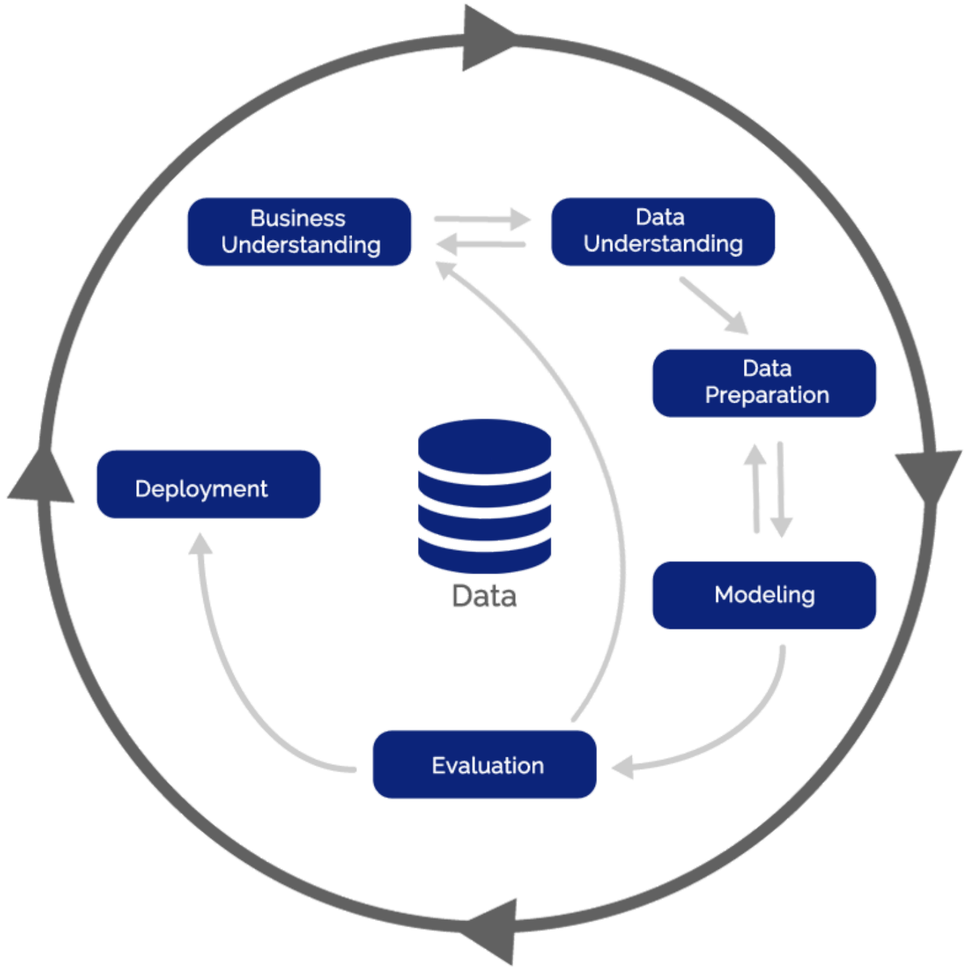


Agglomerete Clustering with 2 clusters



Project Management/Tools

- The way we viewed and approached the project mutated during the course of its development but one can say that we followed a Cross-Industry standard process for data mining (CRISP-DM).
- At first, we focused on understanding the business and data mining goals. These goals were defined after reading the project definition and quickly scanning the dataset because doing so gave us a sense of the actual values and size.
- The Data Understandig step is where we became more familiar with the reality of the dataset and made us rethink about our goals and if they were adjusted for the data.
- Data Preparation suffered changes during the maturing of the project as one understood better the dataset and how the data influenced the path to reach our goals. Therefore some information from the dataset was extracted and other was cut out to make the information in every column relevant.
- In Modeling, we started to apply algorithms to the final dataset but the results weren't satisfying our goals so we relied on outlier removal, hyperparameter tuning, and SMOTE to help us improve our results.



Tools and Individual Factor

Tools Used

- Python on Jupiter NoteBook with the following libraries: pandas, dataframe_image, datetime, matplotlib, numpy, math, sklearn, and imblearn.
- Git and Google Colab were used as the main tools for collaboration as they were the home for all the code and materials used and created during this project.
- Discord was the chosen tool to use for communication within the team where meetings were set to work on the project, distribute tasks and share thoughts and opinions.

Individual Participation

- Miguel Lopes - 25%
- Sérgio Estêvão - 25%
- Duarte Sardão - 25%
- Gabriel Martins - 25%