MACHINE LEARNING

to loan or not to loan

THAT IS THE QUESTION

group 52

João Baltazar up201905616
Nuno Costa up201906272
Pedro Correia up201905348

Domain Description

Records of bank accounts and their clients from 1993 to 1998.

- *4500* accounts
- *5369* clients
- *5369* dispositions
- *77* districts
- *202* cards *(177 dev)*
- *426885* transactions *(396685 dev)*
- *682* loans *(328 dev)*

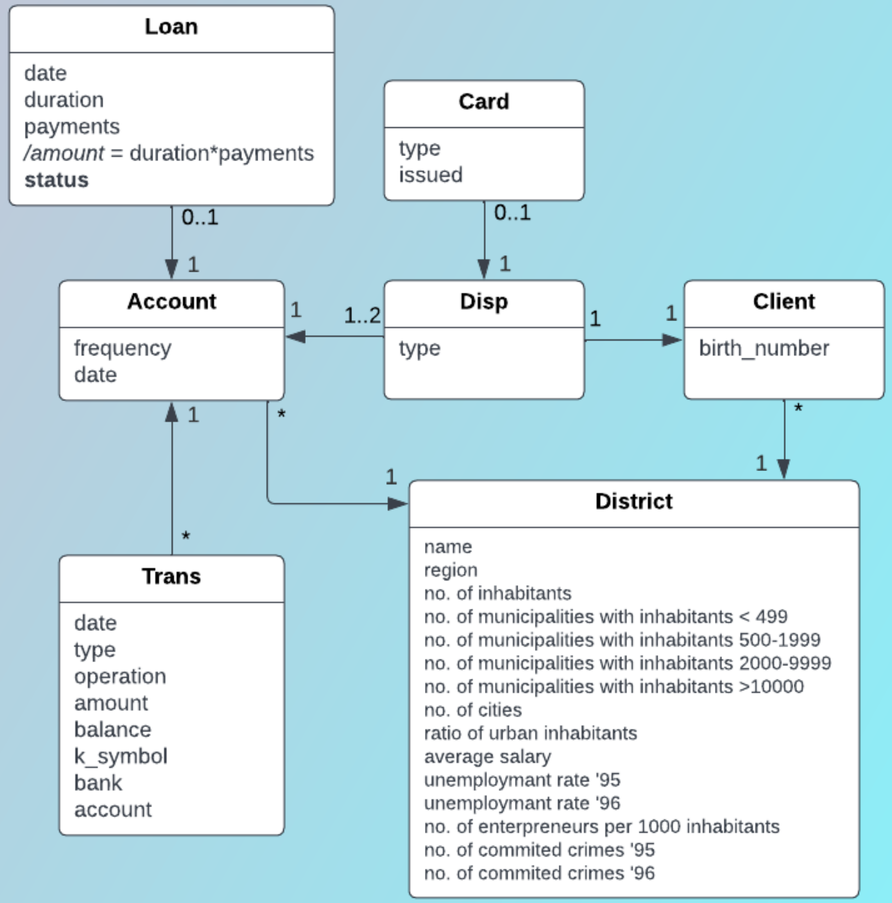
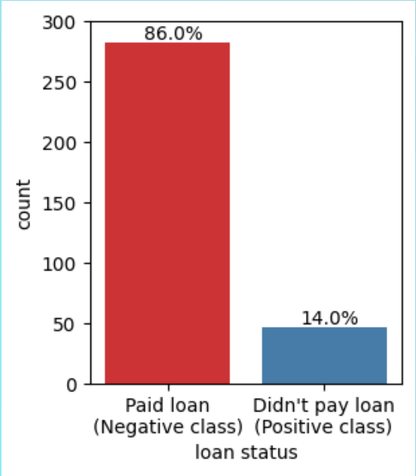


Fig.1 - Relational Model of the Czech Financial Dataset

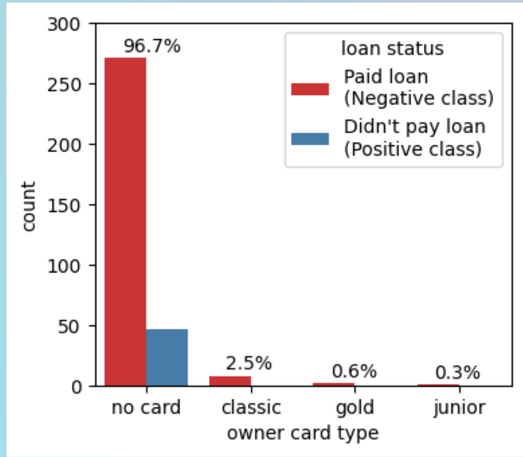
Exploratory Data Analysis

Fig.2 - Imbalanced distribution of loan status



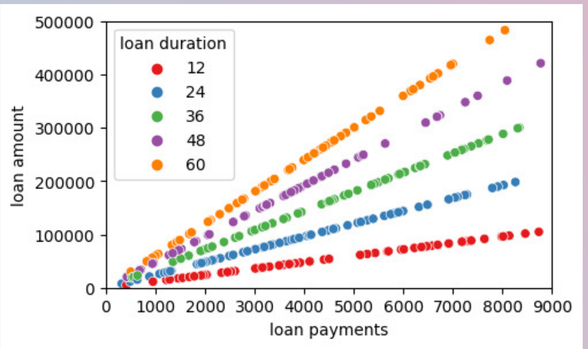
86% of the loans with known outcome were paid off, while *14%* were not.

Fig.3 - Card types of owners of accounts who made a loan



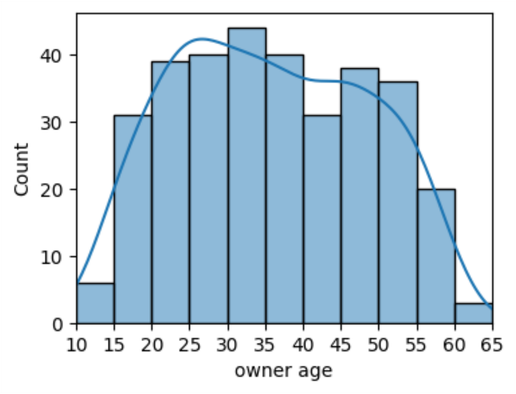
Most clients don't have credit cards, and all who have are owners of an account. The most common card type was *classic*.

Fig.4 - Loan amount is the product of loan payments and loan duration



The bank did not apply interest rate. (*amount = duration×payments*)

Fig.5 - Age of owners of accounts who made a loan

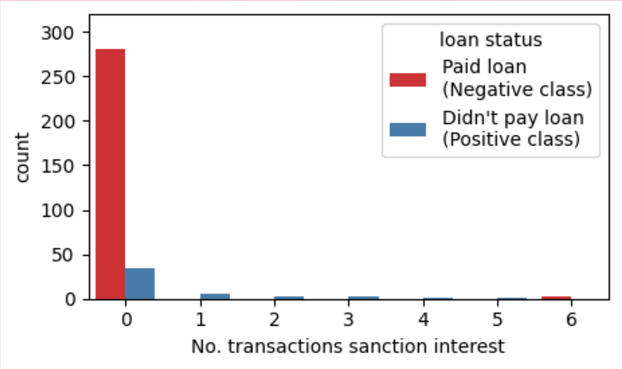


Some loaners are underage and don't have a disponent older than *18* in their account.

group 52

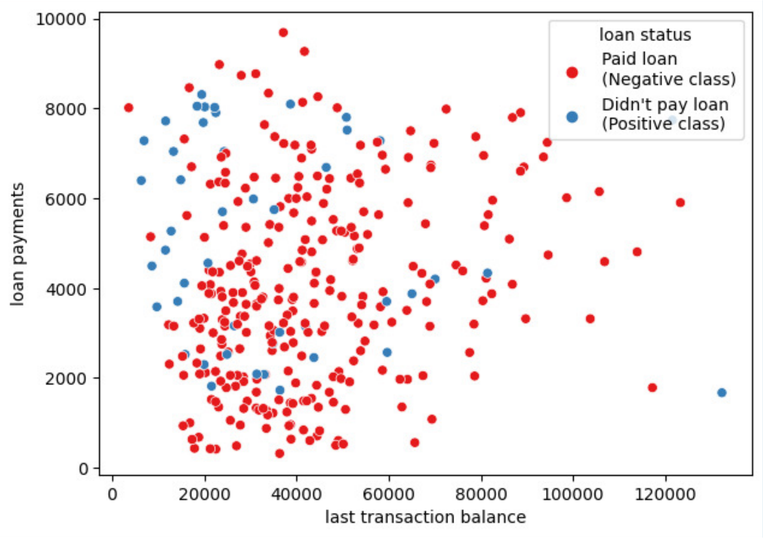
Exploratory Data Analysis

Fig.6 - Number of transactions of sanction interest because of negative balance



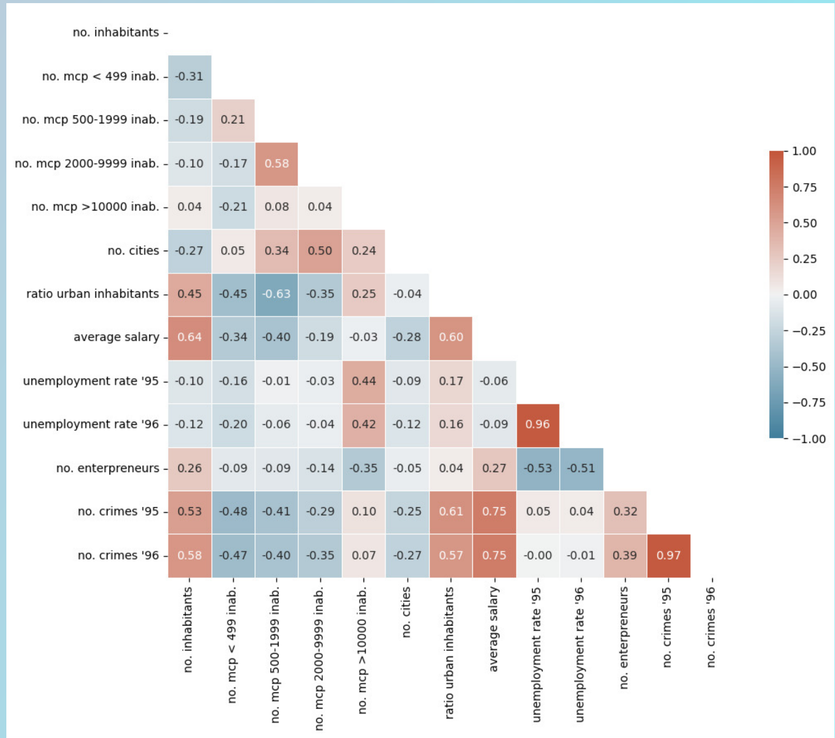
"sanction interest if negative balance" is a good predictor of an unpaid loan.

Fig.7 - Monthly loan payments as a function of balance after last transaction



If the loan is big and the balance of the account after its last transaction is low, it's likely the client will *fail to pay* the rest.

Fig.8 - Correlation matrix of district attributes



Uncovers some nuanced correlations, e.g. a positive correlation between *no. of crimes* and *average salary* — highly populated areas have ***better salaries*** and ***more absolute*** crime.

group 52

Predictive Problem

D E F I N I T I O N

A bank wants to improve their loaning services.

- Better customer understanding
- More thorough and efficient credit analysis
- Confident prediction of loan fulfillment

Task

- **Predict** whether a **future** loan will fail to be paid
- Most loans are paid off → **imbalanced dataset**
- Positive class: **not paid**

Experience

- Loan records until *1996*

Performance Measure

- AUC from predictions on loans from *1997* and *1998*

Predictive Problem

DATA PREPARATION

Join tables based on IDs

> Join *loan*, *account*, *disposition* (*owner*), *card*, *client* and *district*
> Take time into account so *loan records* don't include information from the future

Feature Engineering

> Extract *birthdate* and *gender* from client *birthnumber*
> Use *birthdate* and *loan date* to get *age of the client* (at the *loan*)
> Aggregate features from *transactions* (*counts, means, last transaction balance*)

Missing Values

> Remove columns with **mostly NaN values** (eg. *card issue date*)
> Replace with **mean** — *crime* and *unemployment rate*
> Replace with **0** — *mean withdrawal amount when no transactions*
> Replace *card type* with **"no card"** when there is no *card*

'count_trans_credits',
'count_trans_withdrawals',
'count_trans_credit_cash',
'count_trans_withdrawal_cash',
'count_trans_withdrawal_card', 'count_trans_collection_other_bank',
'count_trans_remittance_other_bank',
'count_trans_ksymbol_interest_credited',
'count_trans_ksymbol_household',
'count_trans_ksymbol_payment_for_statement',
'count_trans_ksymbol_insurance_payment',
'count_trans_ksymbol_sanction_interest_if_negative_balance',
count_trans_ksymbol_oldage_pension',
'last_trans_balance',
'mean_trans_balance',
'mean_trans_amount_absolute',
'mean_trans_amount_credit',
'mean_trans_amount_withdrawal',
'mean_trans_amount_signed',
'owner_male',
'owner_age',
'account_age_months',
'has_disponent',
'owner_profile'

Fig.9 -Engineered Features List

Predictive Problem

DATA PREPARATION

Transformation

- Standardization — *centering* and *scaling*
- Non-linear transformation — for *lognormal/chi-squared* like distributions
- Discretization — *categorical feature encoding* and *binarization*

Feature Selection

- Remove **redundant** attributes using *correlation threshold* (*>0.8*)
- Recursive Feature Selection — *RFECV* and *SequentialFeatureSelector* (*backward*)

Outlier Detection

- **Standard Deviation** — *upper bound = 4 × std*
- **Z-Score** — *upper_bound = z-score + 4*

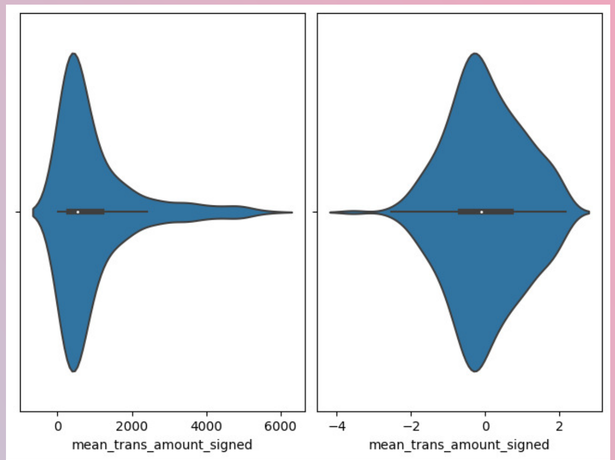


Fig.10 - Mean transaction signed amount before and after PowerTransform non-linear transformation

group 52

Predictive Problem

E X P E R I M E N T A L S E T U P

Feature Definition

Take all previous results into account
Define the input columns and target label

Model Tuning

Past-future train-test split - *TimeSeriesSplit*
Function wrapper to train different models under parameterized conditions

Scoring Metric (the used one was *roc_auc_score*)
Exhaustive (*Grid*) or *Randomized Search* for **hyperparameters** (given the *param_grid*)
Whether to use a Resampler or not (*SMOTETomek*)
used within a **Pipeline** for proper resampling

Try a diversity of Models

DecisionTree (baseline)
RandomForest (robust)
GaussianNB (good for clustered datasets, not the best since it isn't)
LogisticRegression (good for partitioned datasets, PCA indicated it isn't)
GradientBoosting (good to avoid bias)
AdaBoost (good for imbalanced data)
XGB and *LGBM Boost* (not the best given small dataset)
StackingClassifier (good to reassure the output of two or more models)

Calculate and Plot Metrics

Learning Curve, *ROC_AUC* and *Confusion Matrix* plots
Classification Report, *Accuracy* and *ROC_AUC* scores

Predictive Problem

R E S U L T S

Best Results: Stacking Classifier

> Uses *AdaBoost* and *RandomForest*
> Final estimator is *Logistic Regression*

The recall problem

> Since our positive class is so small, models will struggle to explore the positive prediction. Performance during testing was mostly dictated by the model's ability to overcome this imbalance — *AdaBoost is great in these scenarios*. Resampling made it worse - while the produced model would be more general, the FPR would increase so much so that it would lead to lower *roc_auc* scores in already robust models — simpler ones as *Decision Tree* did get a noticeable bump in performance.

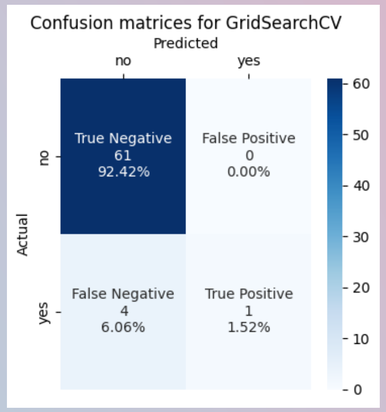


Fig.11 - Confusion matrix for the best classifier obtained

AUC = 0.9115, Accuracy = 0.9394

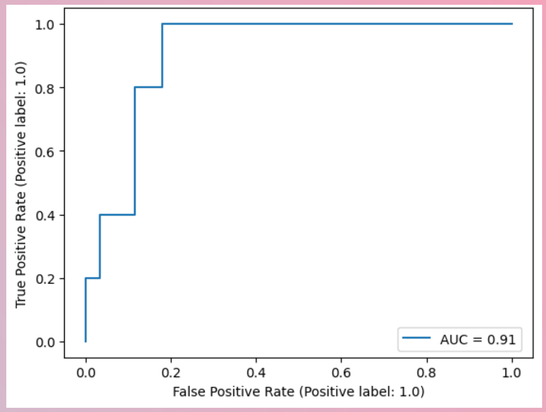


Fig.12 - ROC curve for the best classifier obtained

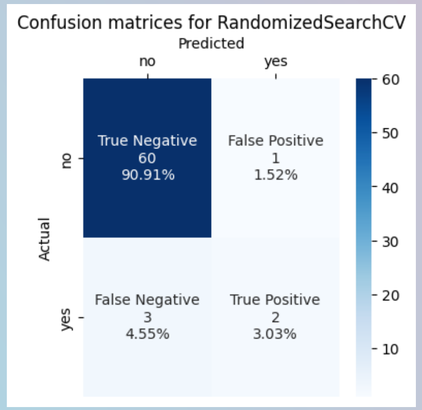


Fig.13 - Confusion matrix for AdaBoostClassifier (no oversampling)

AUC = 0.8721, Accuracy = 0.9394

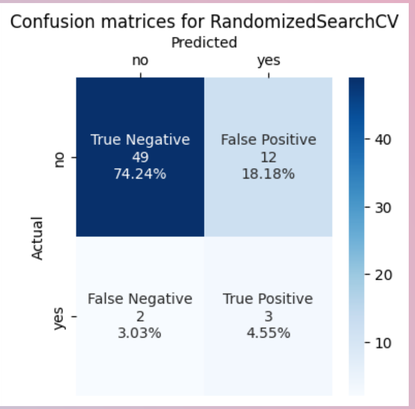


Fig.14 - Confusion matrix for AdaBoostClassifier (oversampled)

AUC = 0.8705, Accuracy = 0.7879

Predictive Problem

R E S U L T S

Performance of other Models

- RandomForest gave good performance out of the box
- GradientBoosting, AdaBoost and LGBM gave reasonable scores
- DecisionTree required a lot of processing to give reasonable scores
- GaussianNB, LogisticRegression and XGB did not give good results

Impact of Preprocessing

- Oversampling helped generalize the model — *good for less robust models*
- Transforming the data had a good impact on the result — *made the data easier for the model to interpret*
- SequentialFeatureSelector improved performance, while all other feature selection techniques were worse than using all available features
- Outlier detection did not produce noticeable changes

  - Given regular thresholds, the 'outlier' sample would be too big, impacting the model negatively
  - Given smaller thresholds, the sample would be smaller but the impact was unoticeable

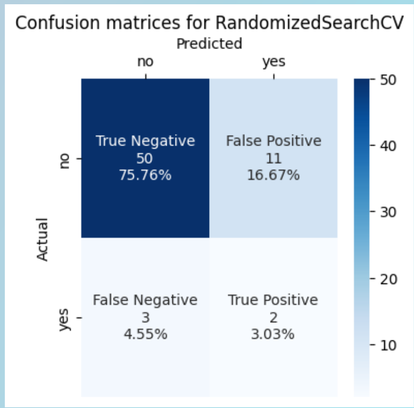


Fig.15 - Confusion matrix for GaussianNB

AUC = 0.7246 Accuracy = 0.7879

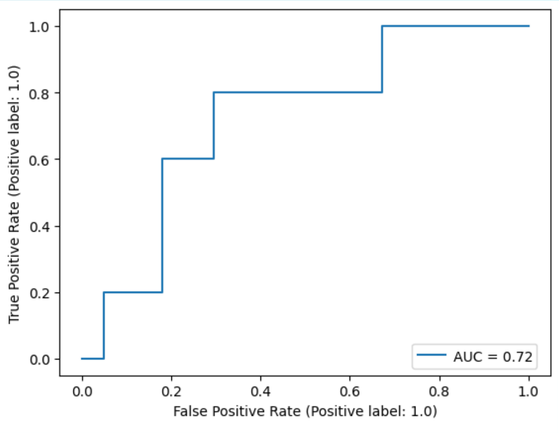


Fig.16 - ROC curve for GaussianNB

Descriptive Problem

Finding socio-demographic profile of account owners who took loans

Metric: *Euclidean*

Algorithms

- KMeans
- KMedoids (PAM)
- AgglomerativeClustering (average-link)

Tuning number of clusters

- Silhouette method
- Elbow method

Evaluation

- Total average silhouette score
- Variance ratio criterion
- Davies-Bouldin score

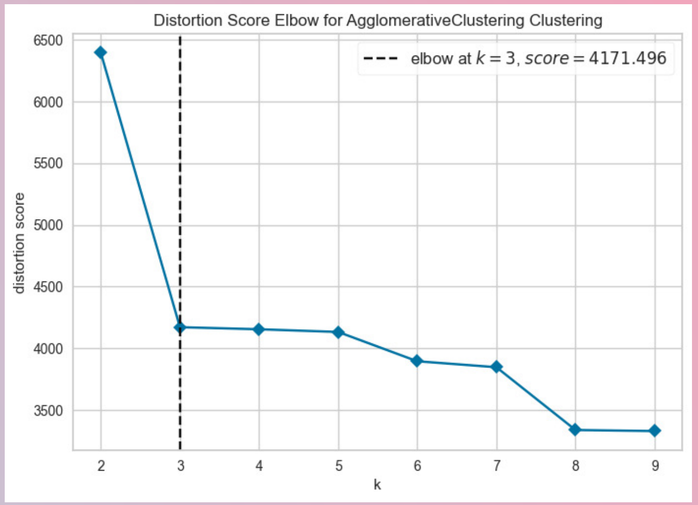


Fig.17 - Distortion Score Elbow for AgglomerativeClustering

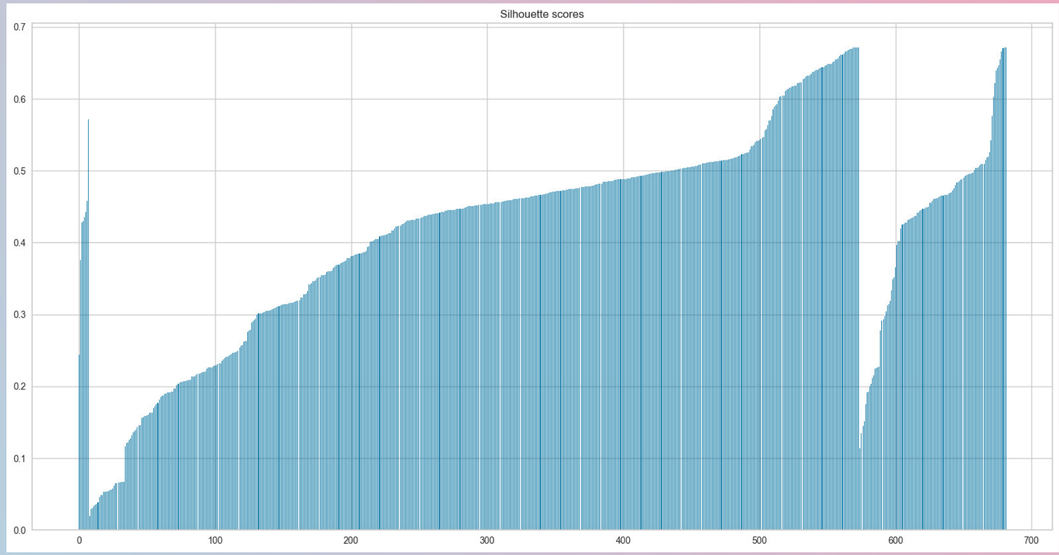


Fig.18 - Silhouette scores for AgglomerativeClustering (3 clusters)

group 52

Conclusions, Limitations And Future Work

DM goals were achieved

> *Customer characterization* and *loan prediction*
> Good results on Kaggle — *0.96213* (public) and *0.93209* (private)

Iteration is essential

> Knowledge about the project and its data evolves over time, so refining previous steps is key

Data is king

> Processing can only do so much, and enhancements often lead to decreases in performance because there is very little data, and heavily imbalanced
> Complementing this dataset would be one of the top priorities for the future

MACHINE LEARNING

to loan or not to loan

THAT IS THE QUESTION

APPENDIX

group 52

Data Quality

Accuracy - to this dataset pertains anonymized Czech bank transactions, account information, and loan records from years 1993 to 1998. However, it is not possible to assert its accuracy with complete certainty given only the information we have.

Completeness - the data fulfills expected comprehensiveness: no missing references in the relation model, and *very* few missing values. It is almost complete.

Consistency - in the few situations where the same information has redundancy (such as *amount = duration×payments*), the data is consistent.

Timeliness - the dataset contains loan information from 1993-1998, and was released in 1999. It is timely.

Uniqueness - after all joins are properly done, there are no duplicate entries. The data is unique.

Validity - although all data conforms to their specified formats, there is a good chance at least some doesn't follow business rules, since, for instance, some loan owners are 14 years old without another disponent, much less a disponent of age.

Predictive Problem

D A T A P R E P A R A T I O N

Join tables based on IDs

> *disposition* and *client* have 1-to-1 relationship: merge
> disposition has at most one card: merge with null values
> account has exactly one owner, but may have more disponents: merge account with its owner, create feature on whether it has another disponent
> account and client have each 1-to-many relationships with district: merge
> transaction has 1-to-many relationship with account: aggregate data into new features
> loan has 1-to-1 relationship with account: merge

> Time taken into account so that *loan records* don't include information from the future (for example cards or transactions).

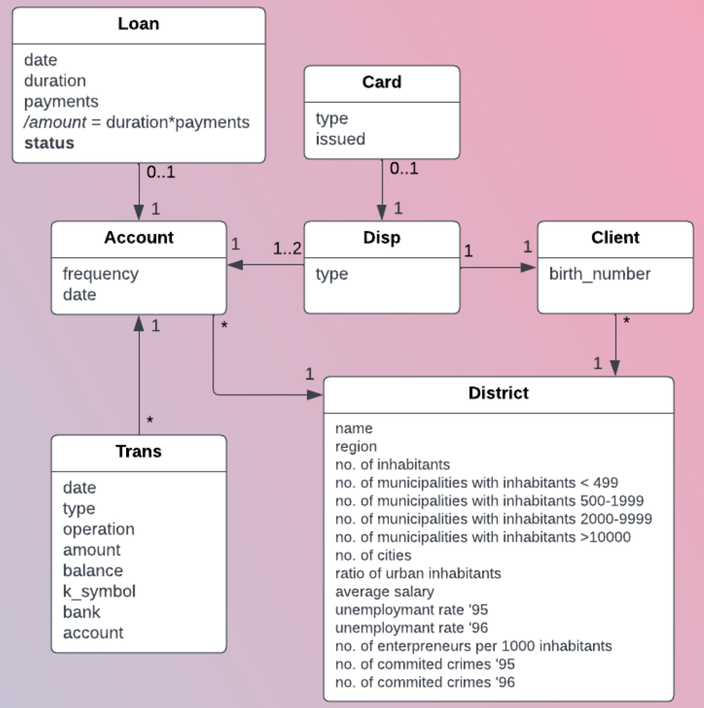


Fig.19 - Relational Model of the Czech Financial Dataset

Predictive Problem

DATA PREPARATION

Feature Engineering

Extract *birthdate* and *gender* from client *birthnumber*

Create has_disponent for whether the account has a disponent other than the owner

Use *birthdate* and *loan date* to get *age of the client* (at the *loan*)

Use *account date* and *loan date* to get *age of the account* (at the *loan*)

Aggregate features from *transactions*

- count categorical features
- balance after last transaction
- mean balance
- mean amount in transaction
  - only credit
  - only withdrawal
  - absolute (credit + withdrawal)
  - signed (credit - withdrawal)

'count_trans_credits',
'count_trans_withdrawals',
'count_trans_credit_cash',
'count_trans_withdrawal_cash',
'count_trans_withdrawal_card', 'count_trans_collection_other_bank',
'count_trans_remittance_other_bank',
'count_trans_ksymbol_interest_credited',
'count_trans_ksymbol_household',
'count_trans_ksymbol_payment_for_statement',
'count_trans_ksymbol_insurance_payment',
'count_trans_ksymbol_sanction_interest_if_negative_balance',
count_trans_ksymbol_oldage_pension',
'last_trans_balance',
'mean_trans_balance',
'mean_trans_amount_absolute',
'mean_trans_amount_credit',
'mean_trans_amount_withdrawal',
'mean_trans_amount_signed',
'owner_male',
'owner_age',
'account_age_months',
'has_disponent',
'owner_profile'

Fig.20 -Engineered Features List

Predictive Problem

DATA PREPARATION

Missing Values

Remove columns with **mostly NaN values** (eg. *card issue date*)

- These columns did not have enough values to give useful information

Replace with **mean** — *crime* and *unemployment rate*

- *Only one row is missing, and these attributes are expected to have values close to the mean*

Replace with **0** — *mean withdrawal amount when no transactions*

- *The meaning of the attribute suggests 0 as the most natural number when there are no transactions, since there are no withdrawals.*

Replace *card type* with **"no card"** when there is no *card*

- *This missing value has significant meaning and can be incorporated as a categorical value on its own, along with the types "junior", "gold" and "classic"*

Predictive Problem

DATA PREPARATION

Transformation

Standardization *(transformation to Gaussian distribution with zero mean and unit variance)* of datasets is a common requirement for many machine learning estimators - if the individual features do not resemble standard normally distributed data, good performance cannot be assured.

Sometimes, due to the nature of the value distribution, simple standardization using *StandardScaler* may not achieve a meaningful transformation, as distance significance may decrease proportionally to magnitude. In these cases, **Non-linear transformation** using *PowerTransform* is applied before standardizing.

→

Data visualization and validation of all columns (before and after)

Fitting on dev set and applying on entire set

Improved performance

Some non-standard columns exhibit binary-like behaviour: from experience, it was understood that counting transactions per type was mostly insignificant and risky due to lack of samples per value, but the *existence* of said types of transactions was very relevant. Thus, these columns were subject to **Binarization**.

Predictive Problem

OUTLIER DETECTION

Several Metrics to characterize features

> *Min, Max, Mean, Median, Standard Deviation, Skewness, Variance, Kurtosis, Interquartile Range*

Detect outliers by defining a range of acceptable values for features

Standard Deviation

> Measures how close to the mean values are
> Range from [-4*std, 4*std]
> Usual cutoff is 3*std, but by applying it a lot of values would've been cut - making the small dataset even smaller

Z-Score

> Scores where the value lies in the normal distribution curve
> Range from [-4, 4]
> Usual cutoff is 3, but the same problem would be found as in the standard deviation

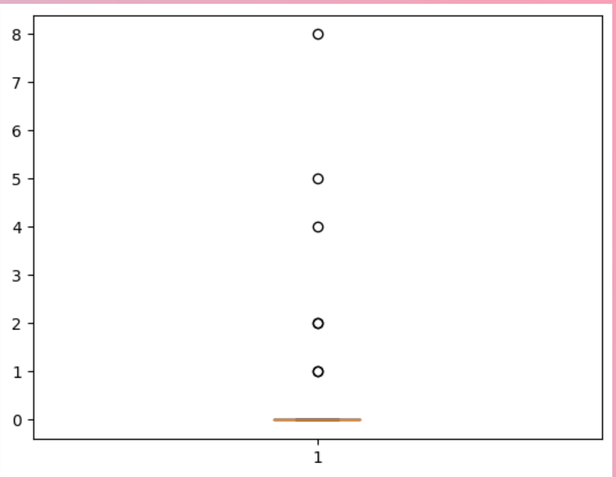


Fig.21 -Box Plot for the *count_trans_withdrawal* **feature**

> Detected outliers belong to categorical variables, that is, integers - mean and standard deviation do not apply the same as they would for continuous variables
> Deleting them would empirically not improve results - which was comproved experimentally

Predictive Problem

F E A T U R E S E L E C T I O N

Main Focus

Remove correlated features
Remove unnecessary features

→

Diminishes the model's complexity
Eases learning/training

Several Approaches

Remove features with a correlation factor > 0.8
Evaluate impact of feature on target variable
Use Recursive Feature Elimination with cross-validation
Use Sequential Feature Selector with:

- forward elimination
- backward elimination
- different estimators

Fig.22 - Correlation Matrix

Several features are correlated with each other, building up a lot of noise for the model training

group 52

Predictive Problem

F E A T U R E S E L E C T I O N

Fig.23 - Different Distributions vs Target Variable

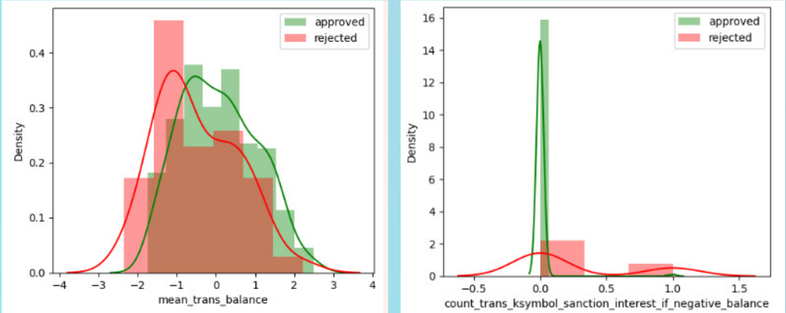


Fig.24 - Target Variable Impact per Feature - 1

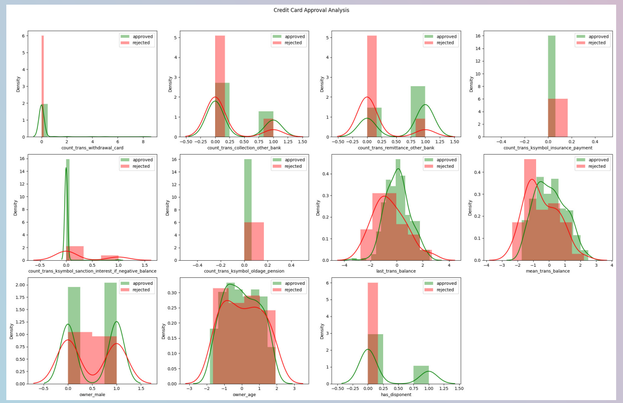


Fig.25 - Target Variable Impact per Feature - 2

Most features' impact on the target variable is **minimal**

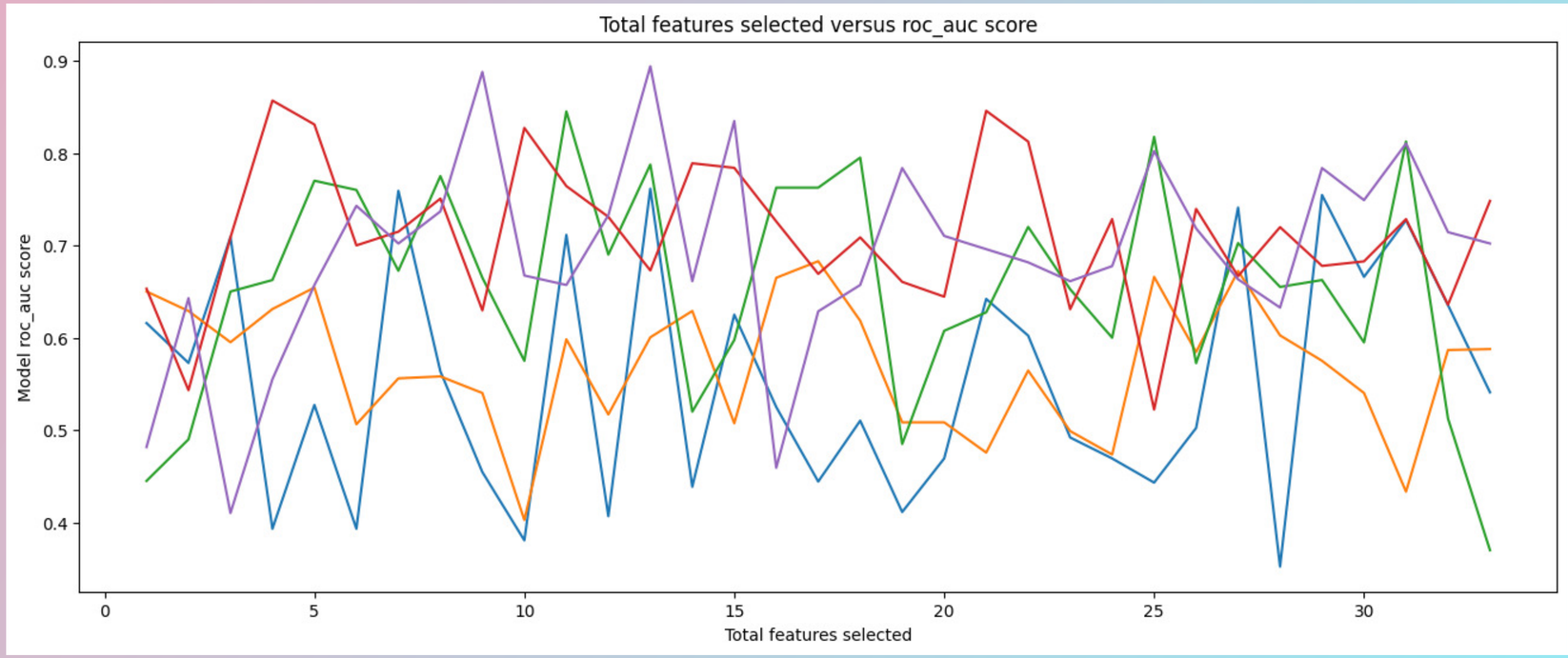
mean_trans_balance has impact as the negative values produce a similar distribution but with different skews

count_trans_ksymbol_sanction_interest_if_negative_balance has impact as the distributions of the target variable are very different

Predictive Problem

F E A T U R E S E L E C T I O N

Fig.26 - Model Performance per number of selected features on each CV run in RFECV using a Decision Tree as its estimator



Difficulty achieving reliable results using RFECV, as there was no tendency towards a certain number of selected features

Usage of different models didn't change the inconsistency of the results

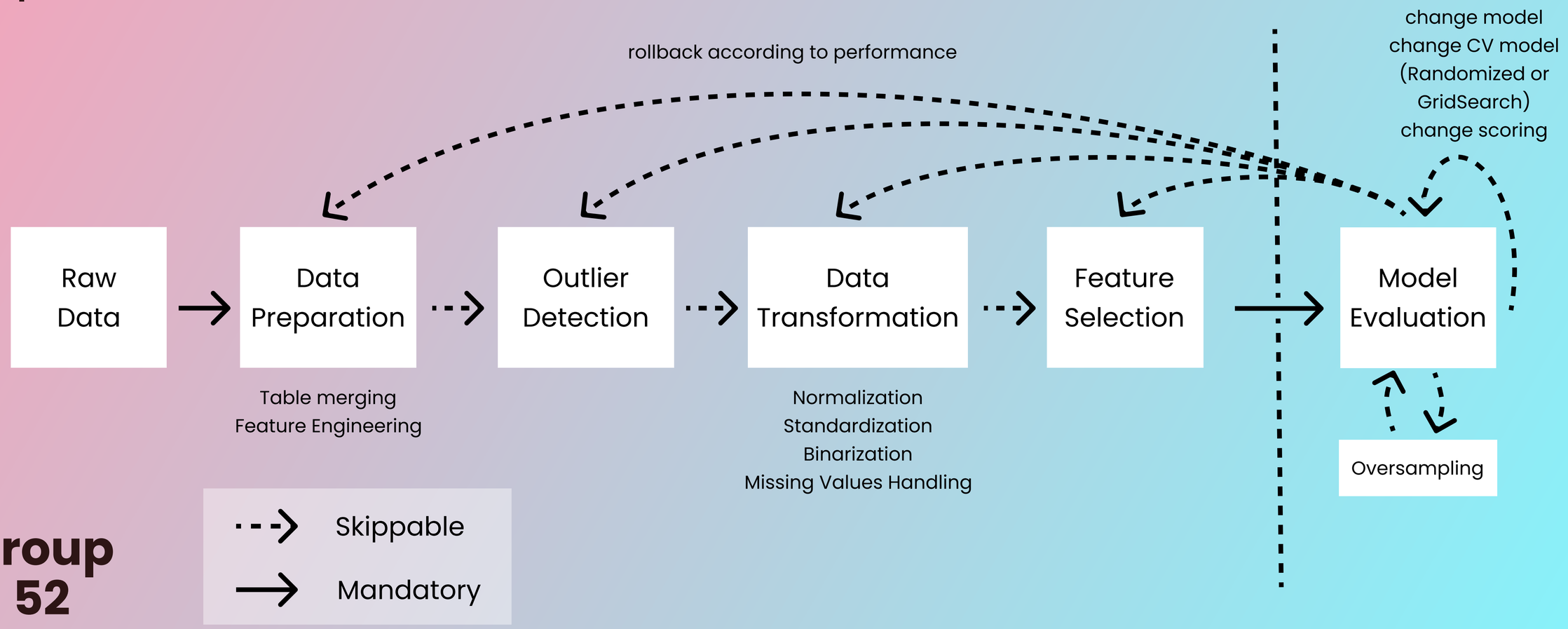
group 52

Predictive Problem

P I P E L I N E

Iterative Approach

- Re-iterate according to performance
- Ability to skip pipeline steps if they do improve performance



group 52

Predictive Problem

C R O S S V A L I D A T I O N

Predictions should take time into account

> Training data has data from the past, whilst test data has data from the future
> Cross Validation measures should take that restriction into account

Defining the folding on CV

> This restriction can be met with the splitter ***TimeSeriesSplit***

but...

TimeSeriesSplit was not as effective as it should in the training dataset, as the partitions would not be balanced since the dataset itself isn't as well - and errors would arise in some partitions as they only had predictions of one of the values of the target class; which basically means that in a period of time on which the train data was partitioned, no loans were not paid

Falling back on StratifiedKFold

> The imbalance in the dataset was more important to tackle than the previously mentioned restriction. Since StratifiedKFold balances all partitions according to the target class, so that the percentage of values of each target class value is maintained throughout partitions, the performance was better - and no errors appeared upon model training.

Predictive Problem

R E S A M P L I N G

Heavily Unbalanced Dataset

86% of the cases relate to the negative class, while 14% relate to the positive

Resampling Techniques

Two methods could be applied to the dataset in order to balance it:

Undersampling of the majority class — As expected, lowers performance significantly - the dataset is small already, so undersampling has a negative impact

Oversampling of the minority class — Best approach here, given that the majority class is relatively small too

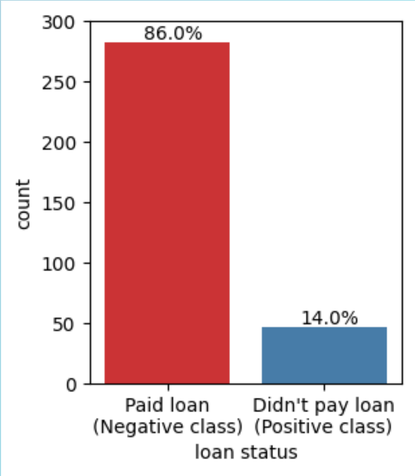
SMOTE — Synthesizes new data for the minority class using a k-neighbors algorithm

SMOTETomek — Applies Tomek Links after applying SMOTE, which removes ambiguous rows from the dataset

By ambiguous we mean rows such that the predicted classes are opposite but the values of their feature are closely related (neighbors)
This removal helps the model generalize better

Both provide similar performance, with a small edge for SMOTETomek

Fig.27 - Imbalanced distribution of loan status



group 52

Predictive Problem

A L G O R I T H M S

Focus on three different algorithms

RandomForest

Ensemble Learning Method
Bagging and Feature Randomness allow for robust results, not prone to overfit (good generalizer)
General purpose algorithm (although it shines in large datasets, which is not the case)

AdaBoost

Ensemble Learning Method
Its iterative approach corrects failed predictions on each iteration, increasing their weights on the next iteration
Great algorithm for unbalanced datasets due to its nature - which is the case of this dataset

StackingClassifier

Ensemble Learning Method
Combines the predictions of a list of several other models - first-level models - to train another - the meta-learner
It reduces biases, allowing for possibly better generalizations - and better AUC scores

The next slides showcase the results for these three algorithms. However, many more were tested:

DecisionTree
GaussianNB
LogisticRegression
GradientBoosting
XGB and *LGBM Boost*

Predictive Problem

R A N D O M F O R E S T

Fig.28 - ROC_AUC curve of the Best Estimator for the RandomForestClassifier

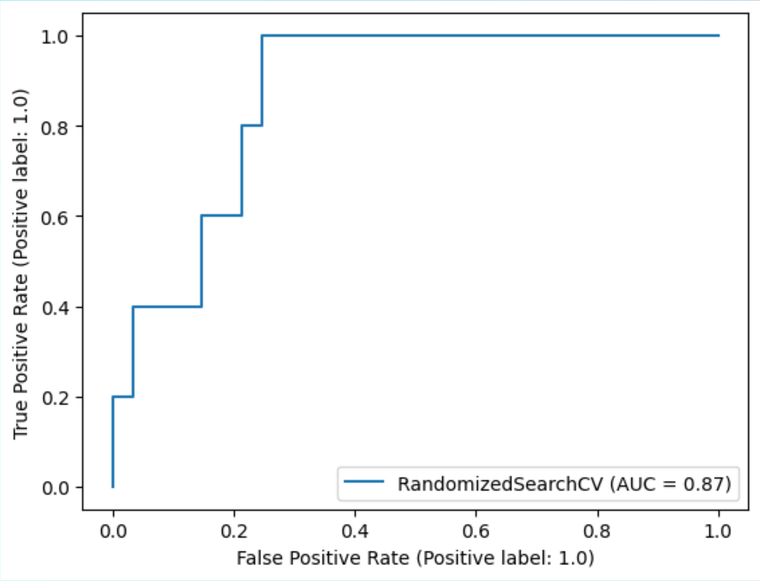


Fig.29 - Confusion Matrix of the Best Estimator of the RandomForestClassifer

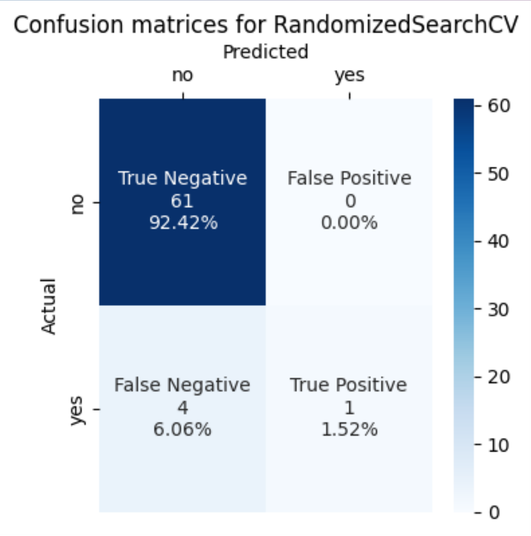
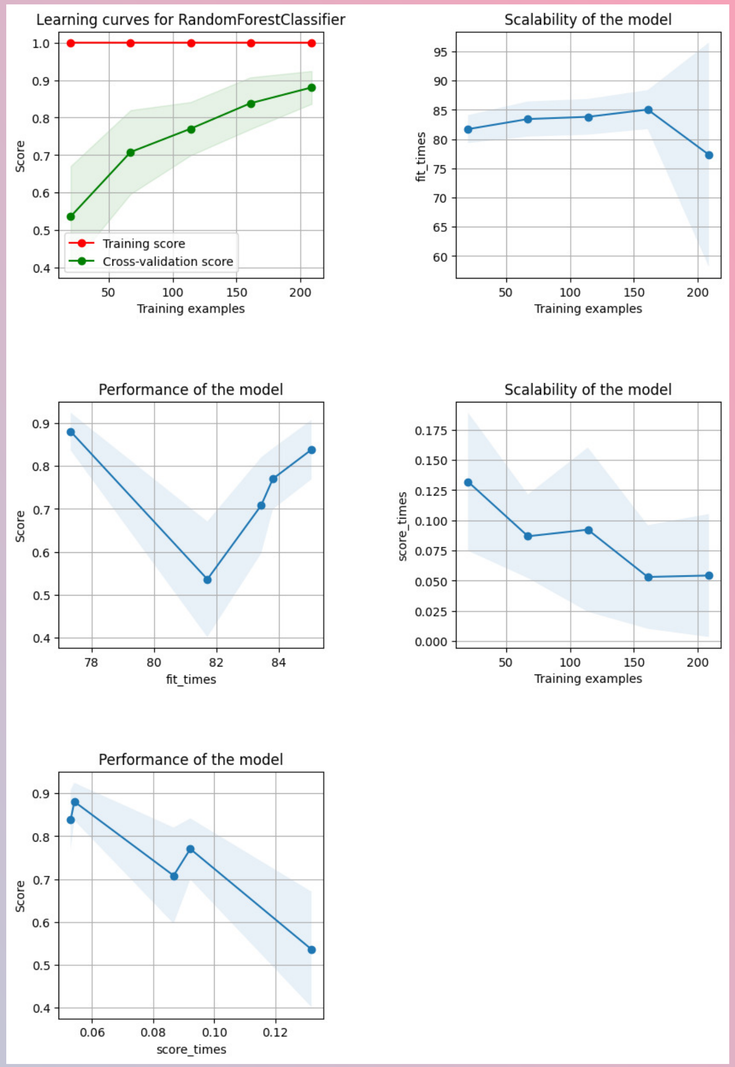


Fig.30 - Learning Curve, Scalability and Performance in relation to score, score_times and fit_times under training of the RandomForestClassifier



Chosen Hyperparameters after GridSearchCV:

```
{
  'n_estimators': 500,
  'learning_rate': 0.01
}
```

group 52

Predictive Problem

A D A B O O S T

Fig.31 - ROC_AUC curve of the Best Estimator for the AdaBoostClassifier

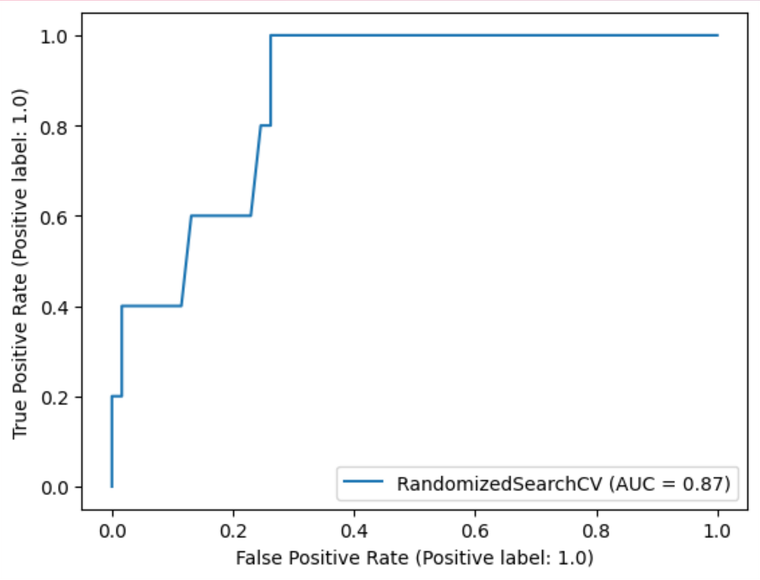


Fig.32 - Confusion Matrix of the Best Estimator of the AdaBoostClassifier

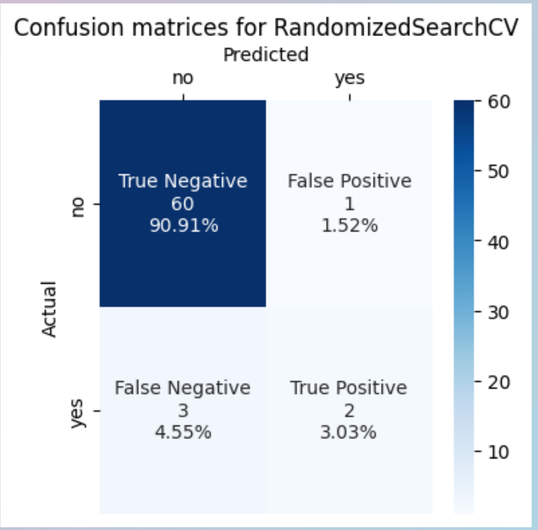
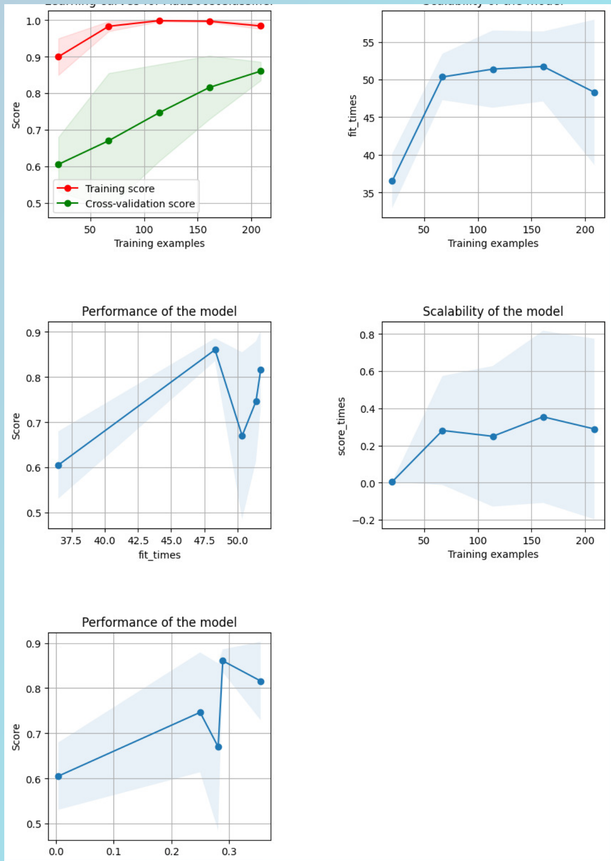


Fig.33 - Learning Curve, Scalability and Performance in relation to score, score_times and fit_times under training of the AdaBoostClassifier



Chosen Hyperparameters after GridSearchCV:

```
{
  'n_estimators': 150,
  'min_samples_split': 2,
  'min_samples_leaf': 3,
  'max_features': 'sqrt',
  'max_depth': 11,
  'criterion': 'entropy',
  'bootstrap': True
}
```

group 52

Predictive Problem

S T A C K I N G C L A S S I F I E R

Fig.34 - ROC_AUC curve of the Best Estimator for the StackingClassifier

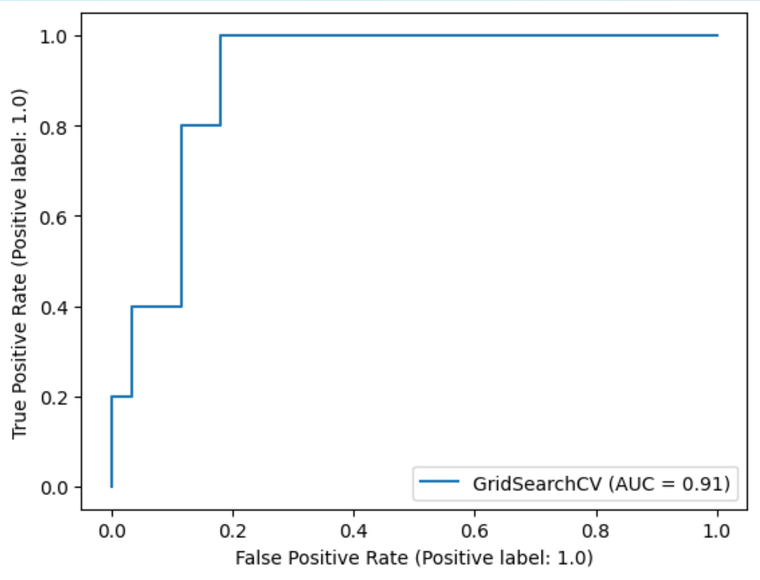


Fig.35 - Confusion Matrix of the Best Estimator of the StackingClassifier

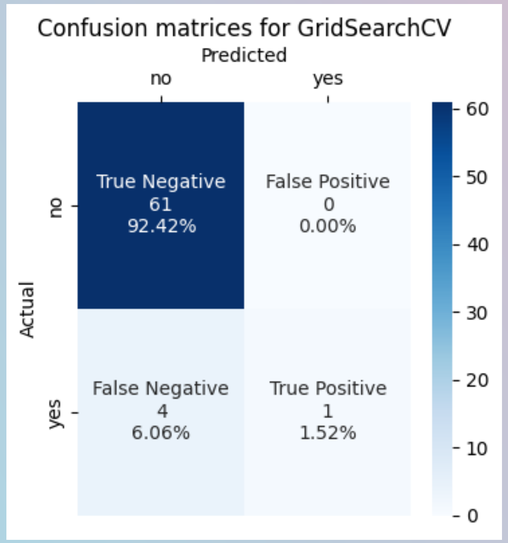
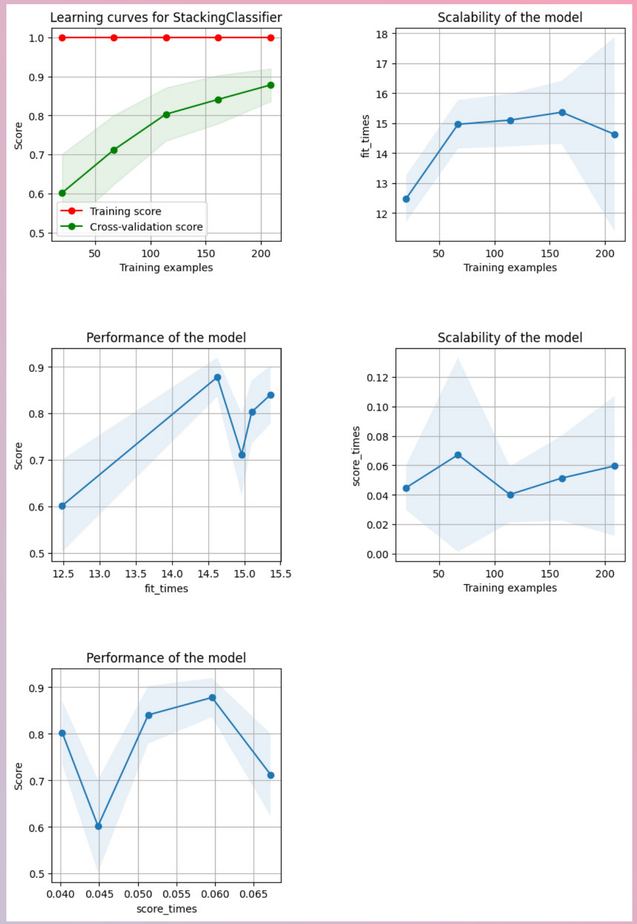


Fig.36 - Learning Curve, Scalability and Performance in relation to score, score_times and fit_times under training of the StackingClassifier



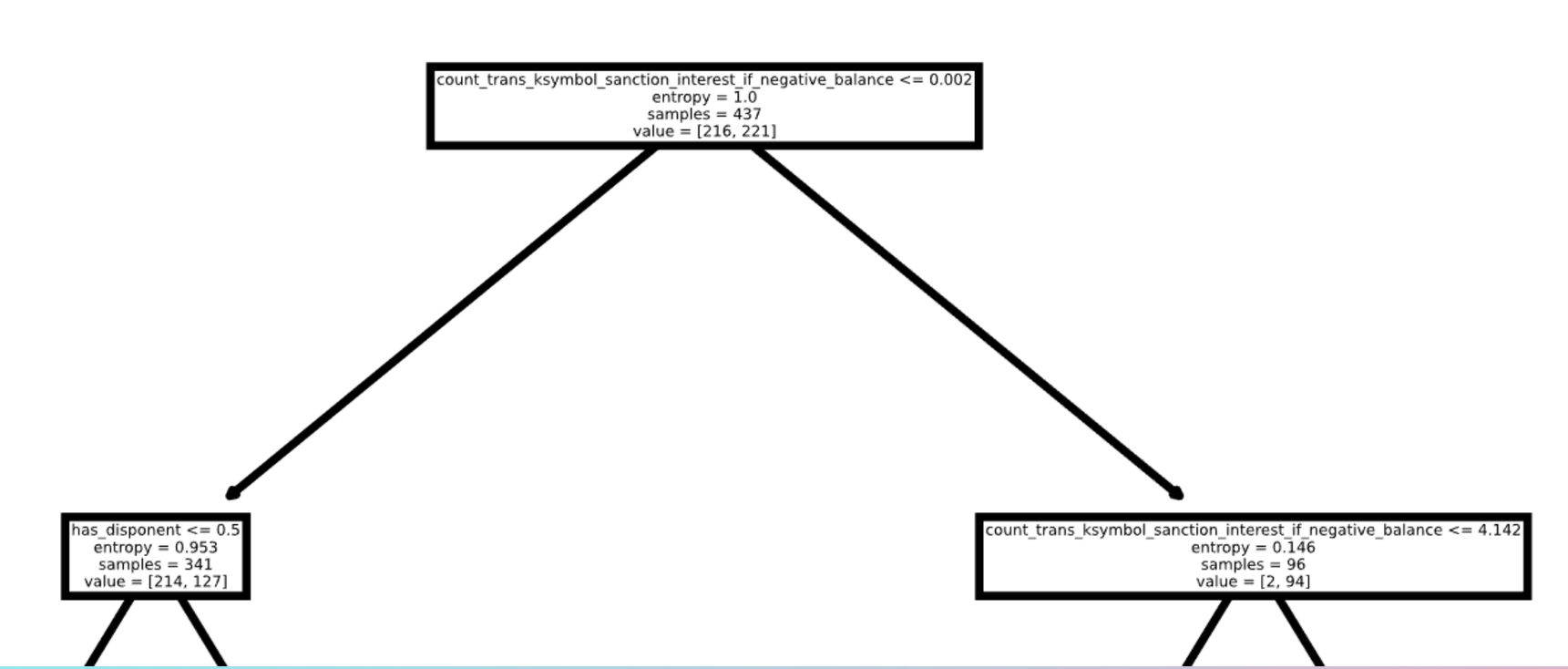
Chosen Hyperparameters after GridSearchCV:

Tuned parameters for both the RandomForest and Adaboost Classifiers

group 52

Predictive Problem

Fig.37 - Top of the Decision Tree for the best DecisionTreeClassifier after running GridSearchCV



As expected, *count_trans_ksymbol_sanction_interest_if_negative_balance* is the most impactful feature for the model

has_disponent and *mean_trans_amount_credit* also have impact on the model's final decision

Descriptive Problem

Finding socio-demographic profile of account owners who made loans

Metric: *Euclidean*

Features: Social or demographic information about the owner

- owner_male
- owner_age
- owner_district_no_inhabitants
- owner_district_ratio_urban_inhabitants
- owner_district_average_salary
- owner_district_unemployment_rate_95
- owner_district_no_enterpreneurs_per_1000_inhabitants
- owner_district_no_crimes_95

- Transformations from data preparation applied
- Redundant attributes removed (eg. no_crimes_96)

Principal Component Analysis

- Explored (2 components) but ultimately discarded
- Worse results and did not help interpret the results

Tuning number of clusters

- Silhouette method
- Elbow method

Evaluation

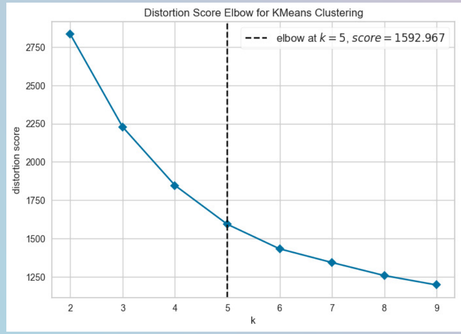
- Total average silhouette score
- Variance ratio criterion
- Davies-Bouldin score

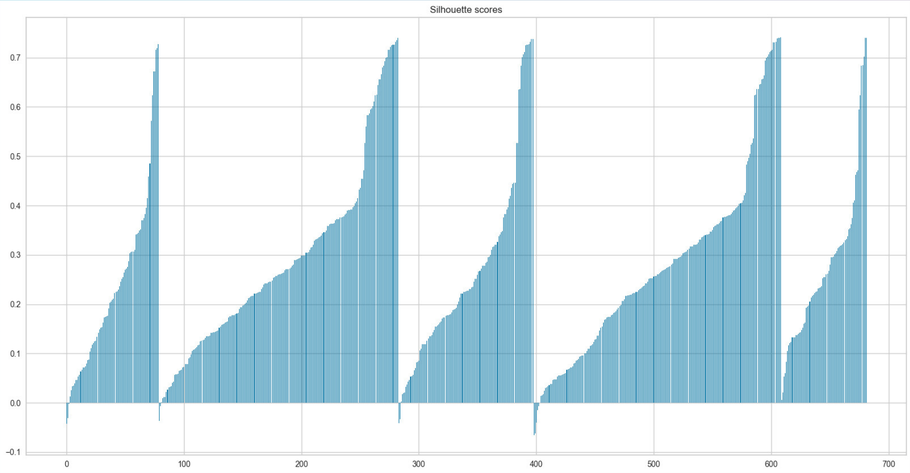
Descriptive Problem

Algorithm: *KMeans*

> Tries to partition the data so that each observation belongs to the cluster with the nearest centroid.

Figs.38 and 39 - KMeans Elbow





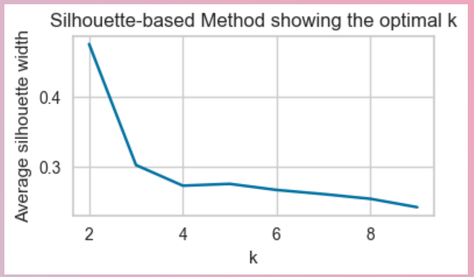
Elbow method - 5 clusters

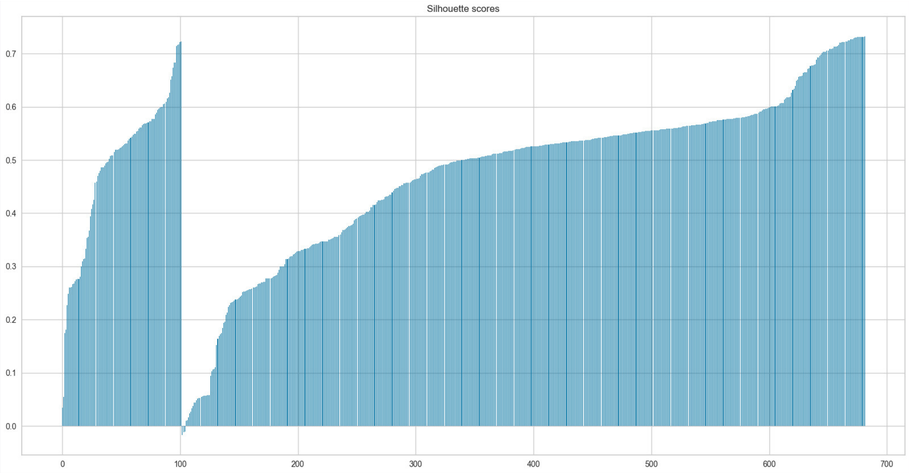
Total average silhouette score: 0.275

Variance Ratio Criterion: 343.701

Davies-Bouldin score: 1.294

Figs.40 and 41 - KMeans Silhouette





Silhouette method - 2 clusters

Total average silhouette score: 0.476

Variance Ratio Criterion: 478.700

Davies-Bouldin score: 0.734

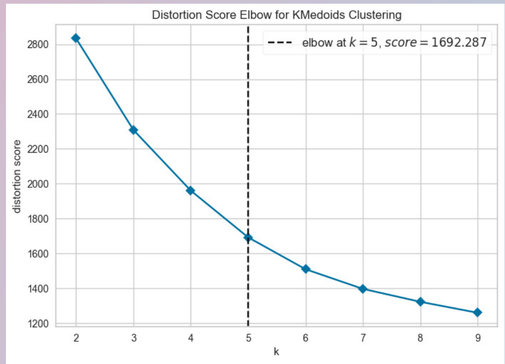
Descriptive Problem

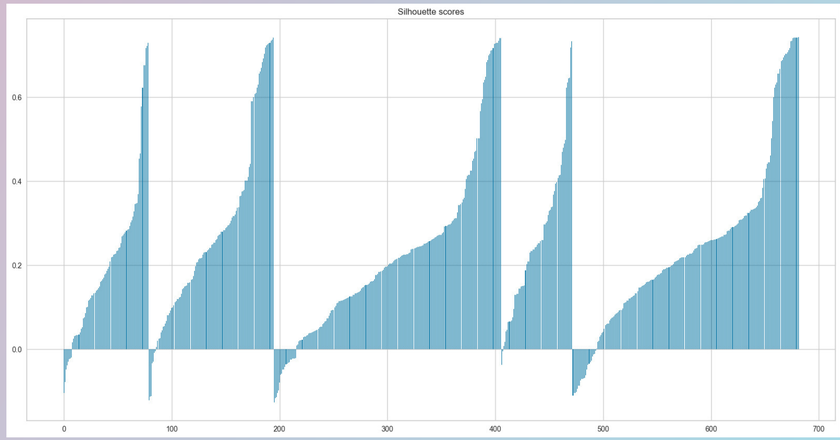
Algorithm: *KMedoids* (PAM)

Similar to *KMeans*, but instead of using the centroid of each cluster, this algorithm uses the medoid (actual data point whose average dissimilarity to all others is minimal).

This method is more robust to outliers.

Figs.42 and 43 - KMedoids Elbow





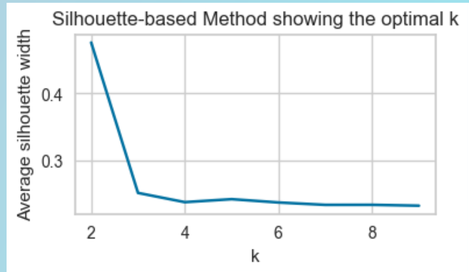
Elbow method: 5 clusters

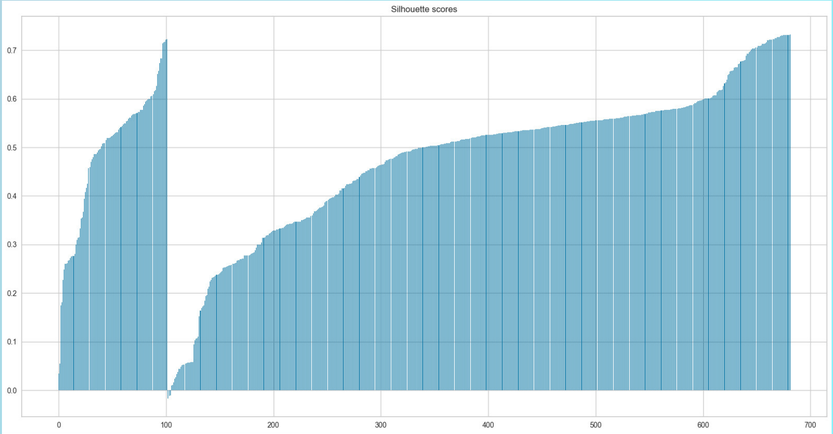
Total average silhouette score: 0.241

Variance Ratio Criterion: 313.596

Davies-Bouldin score: 1.448

Figs.44 and 45 - KMedoids Silhouette





Silhouette method: 2 clusters

Total average silhouette score: 0.476

Variance Ratio Criterion: 478.700

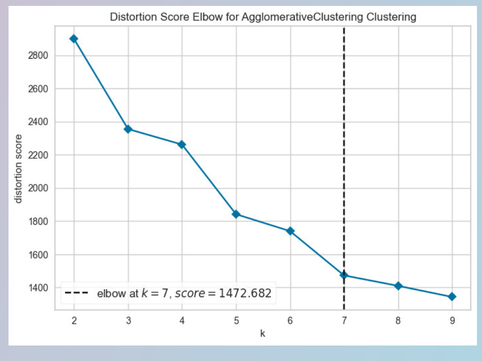
Davies-Bouldin score: 0.734

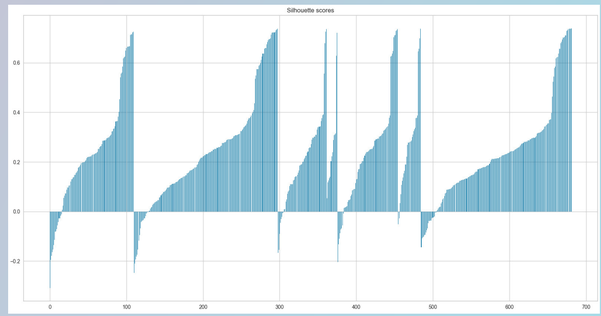
Descriptive Problem

Algorithm: *Agglomerative Clustering (average-link)*

> Starts with a cluster for each observation and on each iteration merges two clusters based on the average distance between their elements.

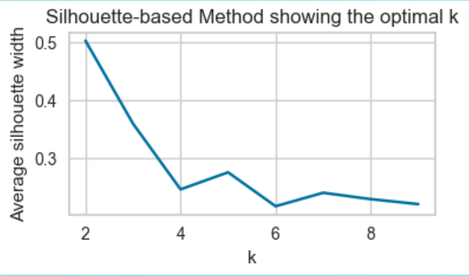
Figs.46 and 47 - Agglomerative Clustering Elbow

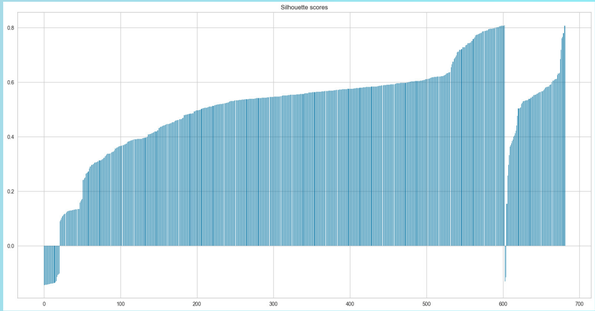




Elbow method: 7 clusters
Total average silhouette score: 0.240
Variance Ratio Criterion: 256.306
Davies-Bouldin score: 1.286

Figs.37 and 38 - Agglomerative Clustering Slihouette





Silhouette method: 2 clusters
Total average silhouette score: 0.503
Variance Ratio Criterion: 453.141
Davies-Bouldin score: 0.586

Descriptive Problem

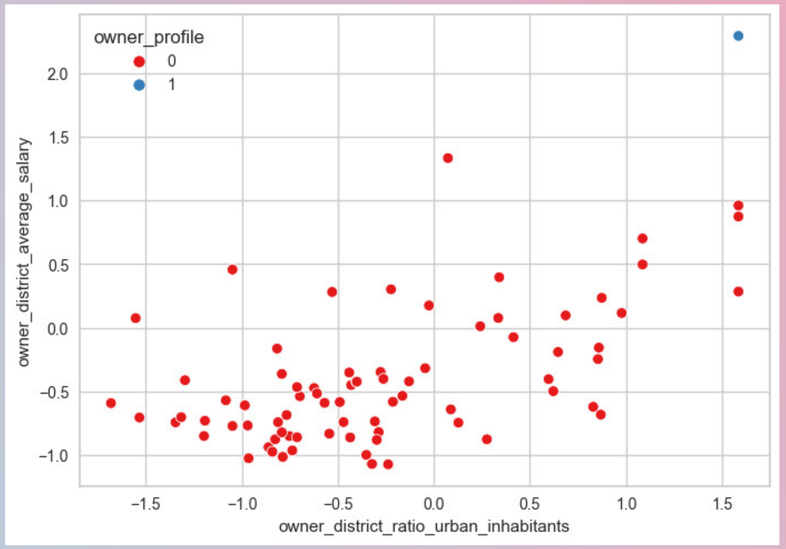
Comparison of Results

> Better scores when using the silhouette method, which led to less clusters
> Best *total average silhouette score*: **Agglomerative Clustering** (2 clusters)
> Best *Variance Ratio Criterion*: **KMeans** and **KMedoids** (2 clusters); both had the same result
> Best *Davies-Bouldin score*: **Agglomerative Clustering** (2 clusters)
> Based on these metrics, we conclude the best method overall was **Agglomerative Clustering** (2 clusters)

Analysis of Results

> Best model had samples with negative silhouette scores, suggesting that those samples do not belong to their cluster. The total average silhoutte score of around 0.503 is not high enough to definitely conclude that the clusters are clearly distinguished from each other.
>
> The profiles obtained from the descriptive problem did not improve the predictive model when used as a feature.

Fig.50 - Clusters in number of inhabitants as a function of the average salary (normalized)



Cluster 1 isolates owners who live in places with a large amount of population and urban inhabitants, high salaries and low unemployment.

Project Management

Plan

Based on CRISP-DM methodology
On each step we would also improve previous steps

1. Business understanding and testing a simple model on the data
2. Data understanding with plots and statistical methods
3. Data integration, cleaning and transformation
4. Feature engineering
5. Modeling with multiple algorithms and evaluation of the results
6. Clustering for the descriptive problem

Tools

Collaboration tools: Github and Discord
Python

- scikit-learn
- numpy
- pandas
- missingno
- matplotlib
- seaborn
- yellowbrick

Contributions [0-1]

João Baltazar (1)
Nuno Costa (1)
Pedro Gonçalo Correia (1)