Bank loan prediction

FEUP - AC@M.EIC

M.EIC07 (2020/21)

João António Sousa (up201806613@edu.fe.up.pt)
Diogo Rodrigues (up201806429@edu.fe.up.pt)
Rafael Ribeiro (up201806330@edu.fe.up.pt)

Domain description

Bank records from a Czech bank:

- 4500 accounts
- 202 cards
  - 177 for development
  - 25 for competition
- 5369 clients
- 77 districts with demographics
- 426885 transactions
  - 396685 for development
  - 30200 for competition
- 682 loans
  - 328 for development (labeled)
  - 354 for competition (unlabeled)

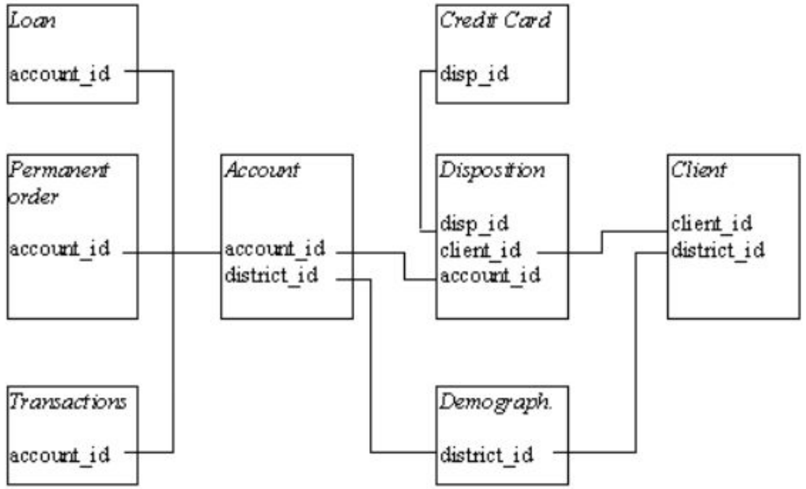


Figure 1 - Class Modeling Diagram Provided

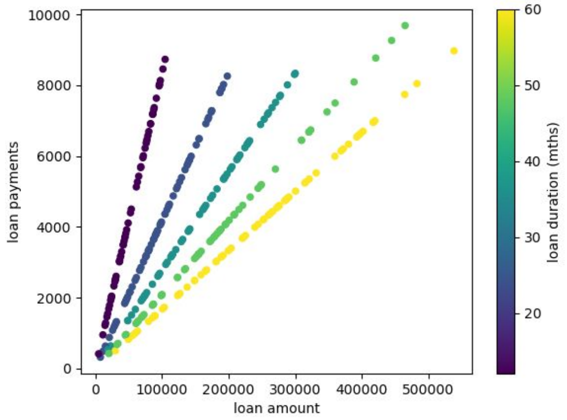
- table Permanent Order found missing

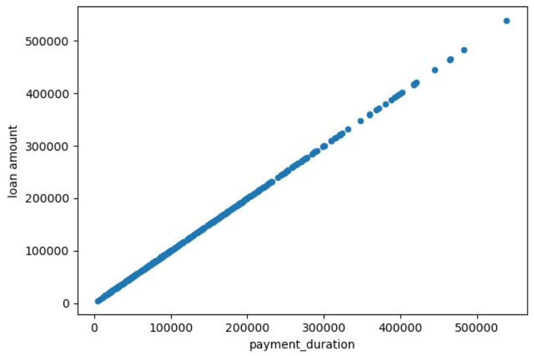
Exploratory data analysis

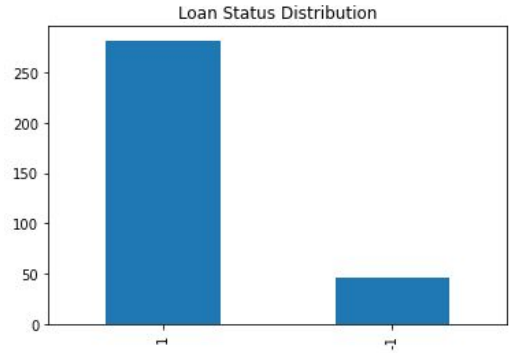
RapidMiner Studio Educational, Jupyter Notebook (Python3 & libraries, e.g. matplotlib, pandas, seaborn), Excel (beginning)

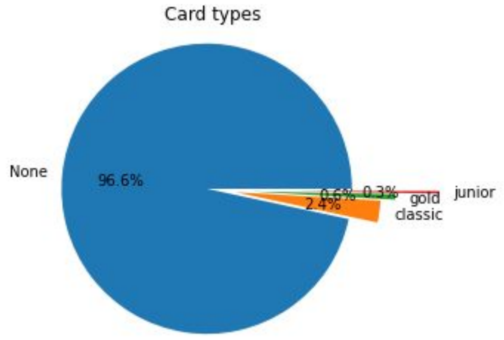
Findings:

- As expected, correlation between the loan amount, duration and payments.
- No interest rate applied by the bank (loan amount = payments * duration).
- Imbalanced loan status class: only 46 (around 14%) out of the 328 labeled loans were of the positive class (-1, i.e. not paid).
- Maximum of a single loan per account.
- The vast majority of people do not own a credit card (96.6%) but among those who do, the classic type is the most popular.







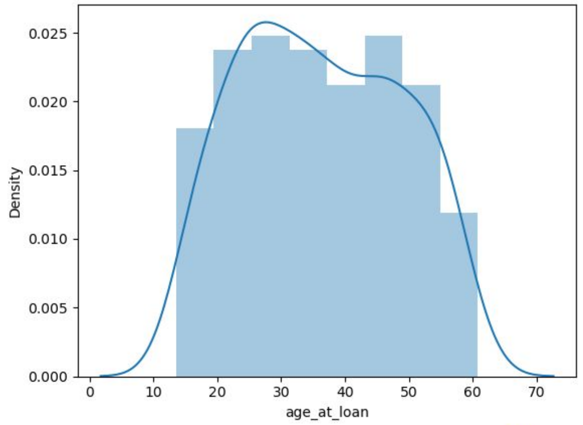


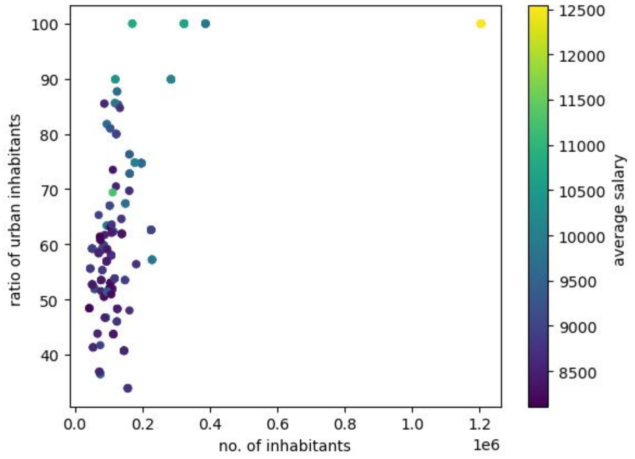
Exploratory data analysis

RapidMiner Studio Educational, Jupyter Notebook (Python3 & libraries, e.g. matplotlib, pandas, seaborn), Excel (beginning)

Findings:

- People's age distribution somewhat according to the expected: at the time of the loan most between their 20s and 40s.
- There are 22 loans (6.7%) that are associated with underaged people (age at loan under 18).
- Higher district number of inhabitants / ratio of urban inhabitants are associated with higher average salary.
- No relevant correlation found between gender of the individuals and the loan outcome.





Predictive data mining problem

Problem definition

A bank wishes to improve its service, by:

- Helping employees and reduce credit analysis time
- Improving its credit analysis
- Increasing/lowering interest for bad/good clients

Create system that:

- Reduces defaulting by 75% (TPR > 75%)
- Does not reduce credit approval below 95% (FPR < 5%)

In data mining goals:

- AUC = 0.7125 is the minimum acceptable
- AUC > 0.8500 is reasonable
- AUC > 0.9125 is good

Data Preparation

Jupyter Notebook (Python3 & libraries, e.g. pandas, numpy) and RapidMiner Studio Educational

- Extraction of birth date and gender form “birth_number” and date format conversion.
- Categorical features encoding / dummy encoding for later use in algorithms, requiring numerical data.
- Renaming of some columns to avoid confusion after dataset joins.
- Table Joins (accounts, cards, clients, disponents, districts, loans, transactions), where each row corresponds to the information available regarding a loan (associated account, client personal data - geographic and financial history), i.e. the complete table has as many rows as the number of loans.
- Replacement of missing values (1 row) of unemployment rate and number of crimes regarding the year 1995 with the median of the respective column.
- Removal of columns with very high percentage of missing values (e.g. ‘bank’ - bank of the partner).
- Experiments using oversampling (smote) in an attempt to deal with the imbalance status class.

Data Preparation

Jupyter Notebook (Python3 & libraries, e.g. pandas, numpy) and RapidMiner Studio Educational

Feature Engineering : 3 main ‘groups’

- Statistic analysis of variables based (for example, sum, minimum, maximum and median transaction amounts of operations made by each ‘loan-associated’ account)
- Features ‘relative’ to others (ratios)
- Binomial features (answer ‘yes’ or ‘no’ questions)

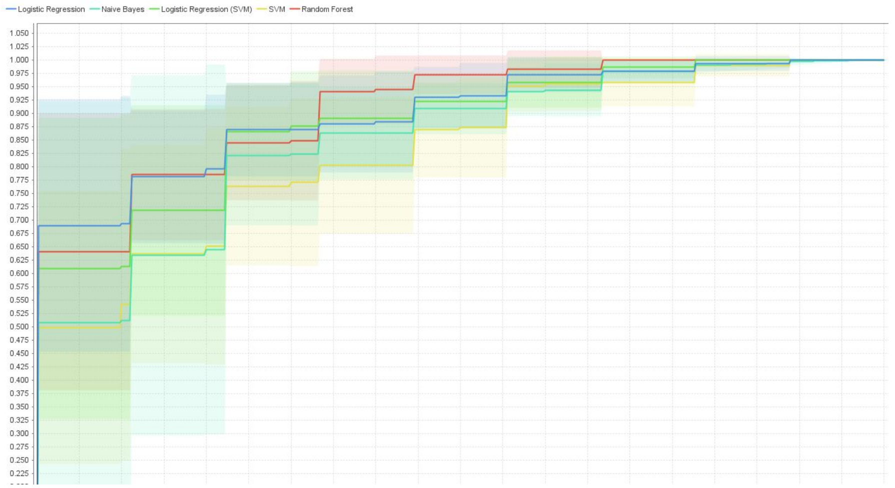
Feature Selection

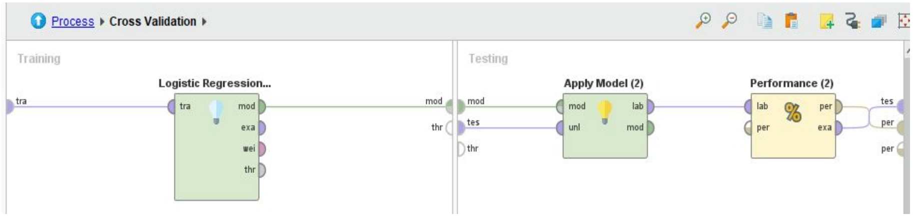
- Manual selection at first. Evolution to filter-based (and some experiments with wrapper-based) method.

Experimental setup

RapidMiner Studio Educational (mainly) and Jupyter Notebook (limited experiments. Python3 & libraries, e.g. sklearn, numpy)

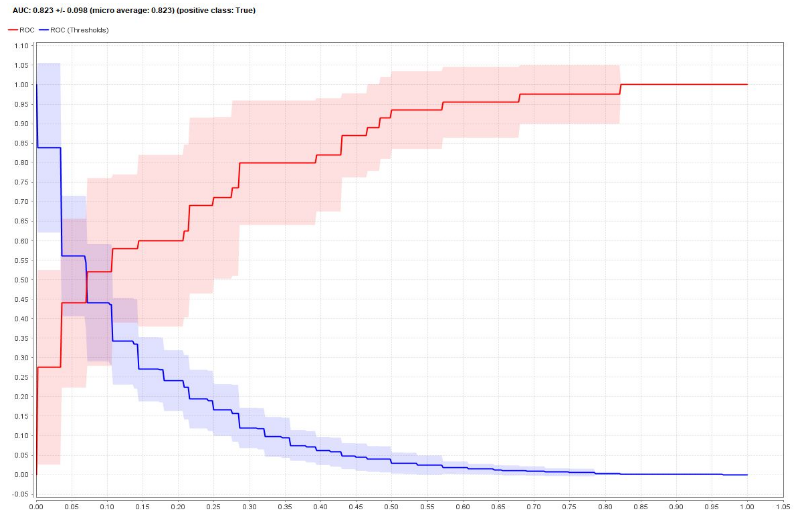
- Import (preprocessed) data
- Algorithm Comparison (“Compare ROCs” operator)
  - Logistic Regression
  - Random Forest
  - SVM
  - ...
- Hyperparameter tuning (manual and ‘Optimize Parameters’ operator)
- Model Training using Cross Validation (5-6 folds)
- Main performance measure main criterion: AUC (others also taken into account)
- Generate predictions.

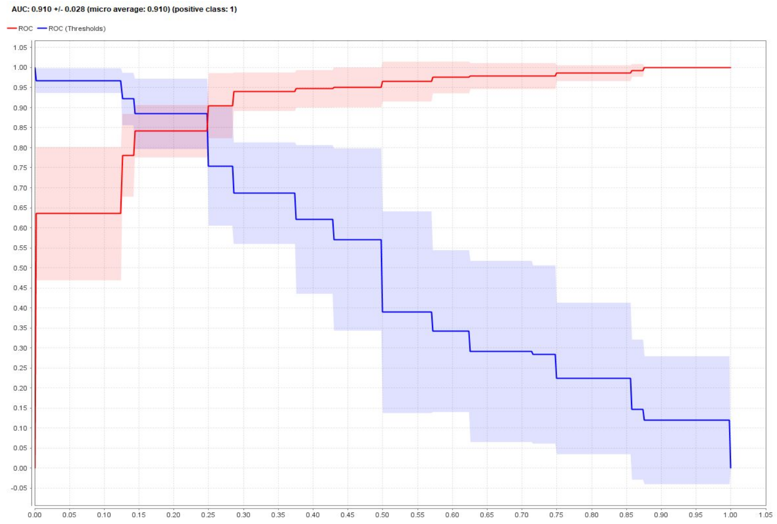




Results

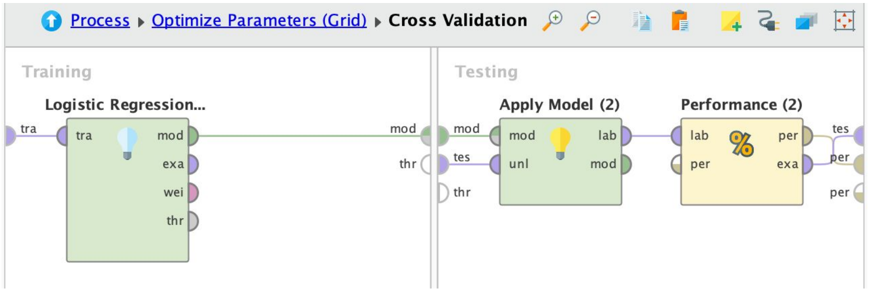
- After the data preparation overhaul (feature generation/selection) there was an improvement in the results (comparison between the images on the right).
- AUC scores in RapidMiner in par with kaggle scores.
- The limited number of experiments with smote showed inconsistency regarding the ‘translation’ of kaggle scores to RapidMiner (lower score value in kaggle than rapidminer AUC).

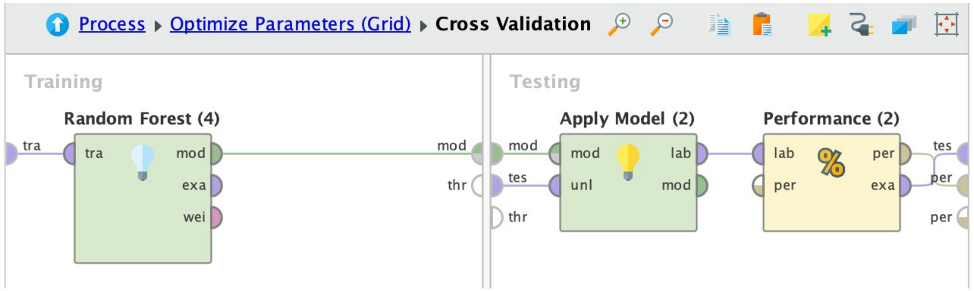




Results

- Focus in two of them: Logistic Regression and Random Forest that ended up translating into our best scoring submissions:
    - Logistic Regression: similar score both in public and private leaderboards. Best submission.
    - Random Forest: public and private scores with a higher discrepancy.

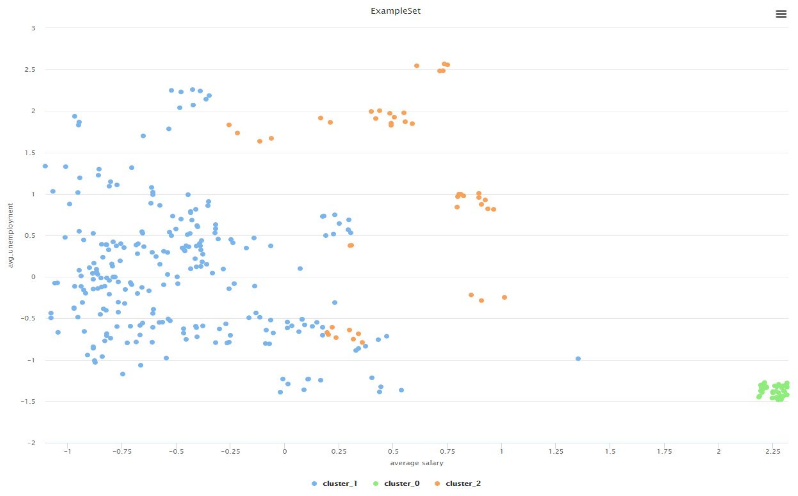


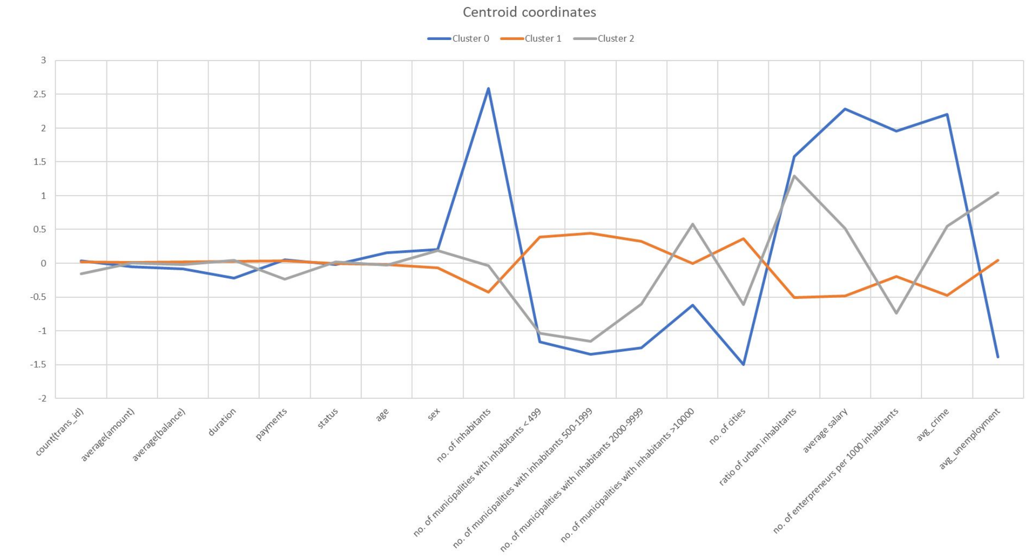


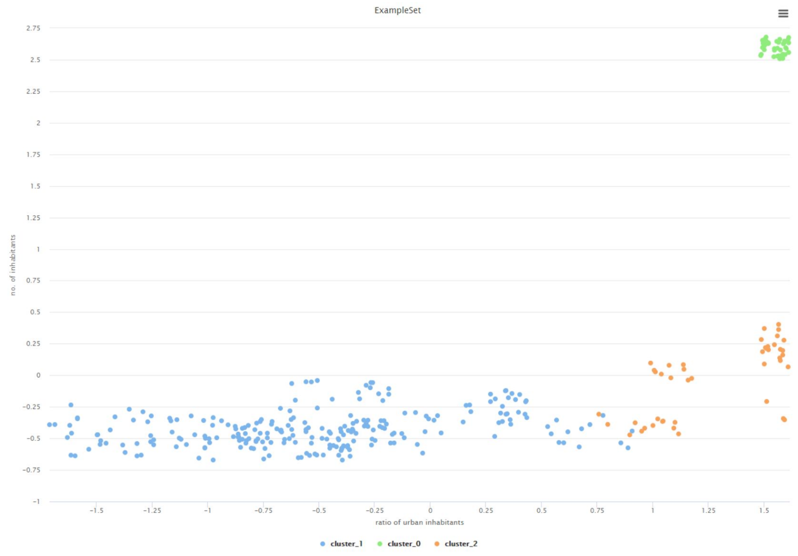
Descriptive Task

RapidMiner Studio Educational (mainly), Excel

- Clustering proved effective at k values of 2 and 3
- Most attempts confirmed our exploratory data analysis findings







Conclusions, Limitations and Future Work

Conclusions

- Feature Generation and Feature Selection (at least in this ‘data context’) have a far greater impact in the results than the hyperparameter tuning of an algorithm or than, in some cases, in its choice.
- Defined Data Mining goals achieved (AUC > 0.9125).

Limitations

- Similar quantity of available data for training (328) comparing to the amount of data to predict (354).
- High number of columns after table joins (before even thinking about feature generation)
- Time-gated and time-limited submissions.

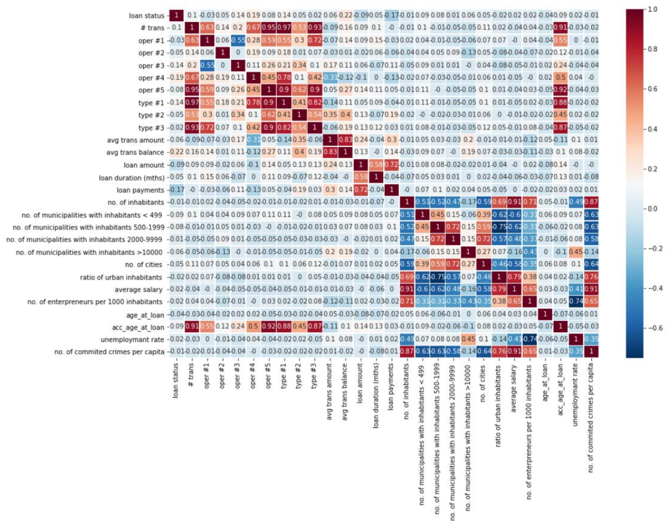
Future Work

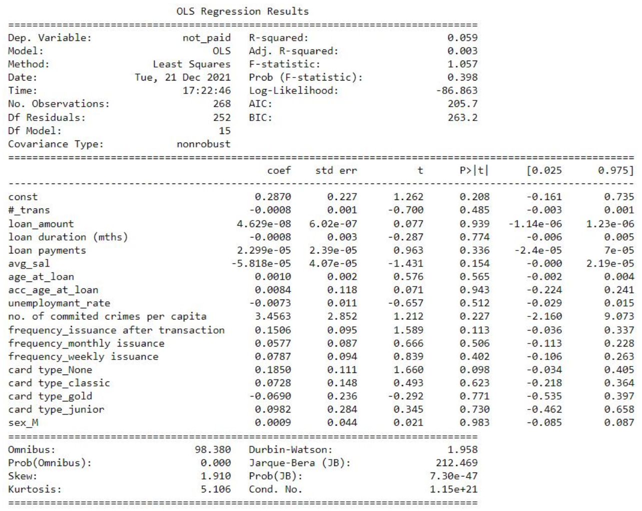
- Further in-depth testing regarding up and undersampling, filter and wrapper methods and explore other algorithms.

Annexes

Exploratory data analysis (DU)

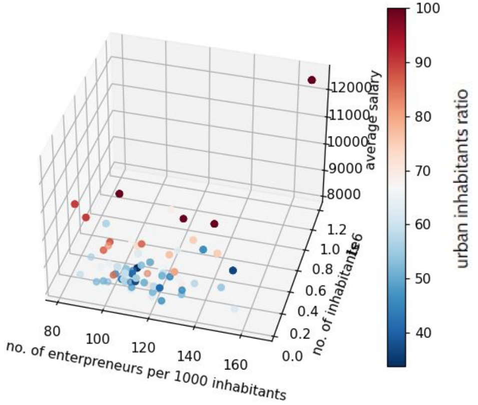
- At first, a very basic analysis of each individual column was performed to get an overview of the datasets provided (see Data Exploration notebook). Then a more in-depth understanding was developed (see Complete notebook).
- At a variable-level, basic metrics calculated (depending on the type of the variable): frequency, mean, median, minimum, maximum, standard deviation, (variance,) skewness, kurtosis and interquartile range to infer about the distributions.
- Regarding multivariable analysis, correlation matrices (both pearson and spearman yielded similar results) with p-value calculations, for some combinations of attributes basic stats (e.g, mean) applied to each value of a categorical value, and regression, by applying OLS (which indicated that Linear Regression would not be a good choice to explore later).

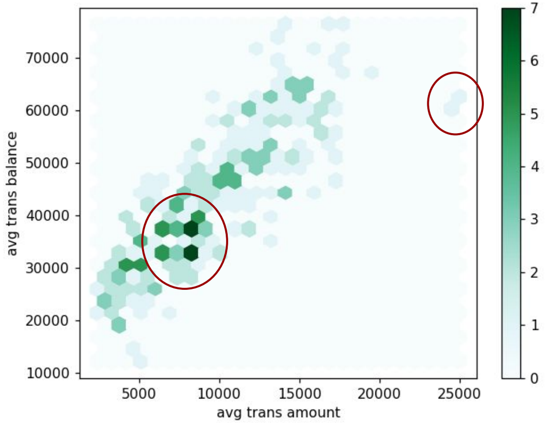



```
                            OLS Regression Results
==============================================================================
Dep. Variable:               not_paid   R-squared:                       0.059
Model:                            OLS   Adj. R-squared:                  0.003
Method:                 Least Squares   F-statistic:                     1.057
Date:                Tue, 21 Dec 2021   Prob (F-statistic):              0.398
Time:                        17:22:46   Log-Likelihood:                -86.863
No. Observations:                 268   AIC:                             205.7
Df Residuals:                     252   BIC:                             263.2
Df Model:                          15
Covariance Type:            nonrobust
==================================================================================================
                                       coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
const                                0.2870      0.227      1.262      0.208      -0.161       0.735
#_trans                             -0.0008      0.001     -0.700      0.485      -0.003       0.001
loan_amount                       4.629e-08   6.02e-07      0.077      0.939   -1.14e-06    1.23e-06
loan duration (mths)                -0.0008      0.003     -0.287      0.774      -0.006       0.005
loan payments                     2.299e-05   2.39e-05      0.963      0.336    -2.4e-05       7e-05
avg_sal                          -5.818e-05   4.07e-05     -1.431      0.154      -0.000    2.19e-05
age_at_loan                          0.0010      0.002      0.576      0.565      -0.002       0.004
acc_age_at_loan                      0.0084      0.118      0.071      0.943      -0.224       0.241
unemploymant_rate                   -0.0073      0.011     -0.657      0.512      -0.029       0.015
no. of commited crimes per capita    3.4563      2.852      1.212      0.227      -2.160       9.073
frequency_issuance after transaction 0.1506      0.095      1.589      0.113      -0.036       0.337
frequency_monthly issuance           0.0577      0.087      0.666      0.506      -0.113       0.228
frequency_weekly issuance            0.0787      0.094      0.839      0.402      -0.106       0.263
card type_None                       0.1850      0.111      1.660      0.098      -0.034       0.405
card type_classic                    0.0728      0.148      0.493      0.623      -0.218       0.364
card type_gold                      -0.0690      0.236     -0.292      0.771      -0.535       0.397
card type_junior                     0.0982      0.284      0.345      0.730      -0.462       0.658
sex_M                                0.0009      0.044      0.021      0.983      -0.085       0.087
==============================================================================
Omnibus:                       98.380   Durbin-Watson:                   1.958
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              212.469
Skew:                           1.910   Prob(JB):                     7.30e-47
Kurtosis:                       5.106   Cond. No.                     1.15e+21
==============================================================================
```

Exploratory data analysis (Data Understanding)

- Data Visualization was made possible mainly by some Python libraries like matplotlib and seaborn.
- Hexbin, pair (as a whole 'N-vars D'), histograms and scatter are some examples of the types of plots created (nothing over 4D to facilitate understanding).
- Throughout the notebook ('Complete notebook'), each graph has some observations under it (description and/or a possible explanation) that highlight other perspectives and/or confirmed the results of the 'mathematical' analysis.
- Examples (from left to right):
  - Higher concentration of transactions with amounts and balance in the 5-10k and 30-40k ranges, respectively. A note to the few data points away from the majority.
  - Higher average salary related with urban and highly populated areas.





Data Quality

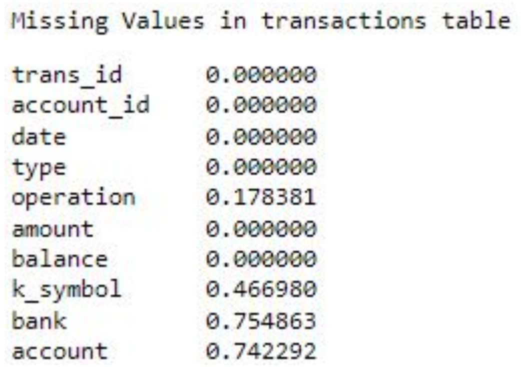
- From the Data Analysis it is possible to extract some conclusions regarding the provided data.
    - In terms of Completeness, from the set of loans point of view, which we consider the ‘elementary unit’ of the predictive problem, it is close to 100% complete. There is data available for each loan regarding the associated account and the person/people who administer it (with the exception, e.g, of some missing values in the district dataset)
    - Despite being of extreme importance in a context like the one in question (banking industry), the Accuracy of the provided data could not be asserted. It appears to illustrate valid real life situations but there was not a way we could verify it.
    - Consistency somewhat follows the same line of thought. There are no instances of entries in two different tables that unmistakably refer back to the same entity to allow for this attribute to be measured.

Data Quality (cont.)

- From the Data Analysis it is possible to extract some conclusions regarding the provided data.
    - There aren't any duplicate entries in the provided data. In the complete dataset, that is the result of joins between all the tables, there are attributes with as many unique values as the number of existing rows.
    - The Validity of the data is rather questionable: the existence of loans to certain accounts that have as owners people under the age of 17 (and in some cases no other person associated), is not legally possible (recalling the hypothesis that the [17,18] years range may refer to student loans and therefore valid).
    - The period of time (the 1990s) at which he data refers to affects its Timeliness. The loans to predict are very close in date to the ones in the training dataset, which translates into forecast confidence.
    - That wouldn't apply if we were to predict current year (2021) loans. Ideally the data should be as 'fresh' as possible. Predictions based on data that is decades old does not inspire nearly as much confidence as predictions made with data from last year and can even be considered dangerous by projecting an unlikely/wrong future scenario.

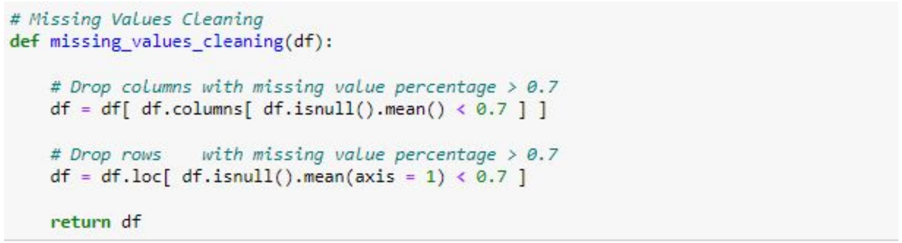
Data Preprocessing

- As mentioned in the presentation slides some missing values were found.
- In some cases (e.g. bank of the partner in the transactions table), the columns were mostly made up of nulls (missing values). In these 'extreme' cases where there weren't less than 30% of data, the column was deleted.

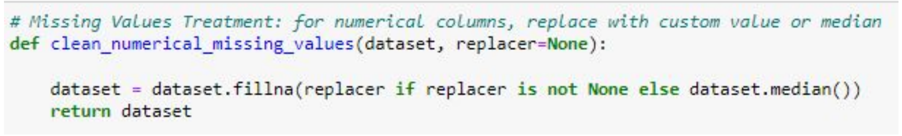


```
Missing Values in transactions table

trans_id      0.000000
account_id    0.000000
date          0.000000
type          0.000000
operation     0.178381
amount        0.000000
balance       0.000000
k_symbol      0.466980
bank          0.754863
account       0.742292
```

- The 'exception' is in the card table where the missing values were interpreted as 'no card' and replaced by '-1' (which motivated the 'has card' boolean feature afterwards).
- The general rule (applied in the district table as mentioned before), was replacing the null value by the median.



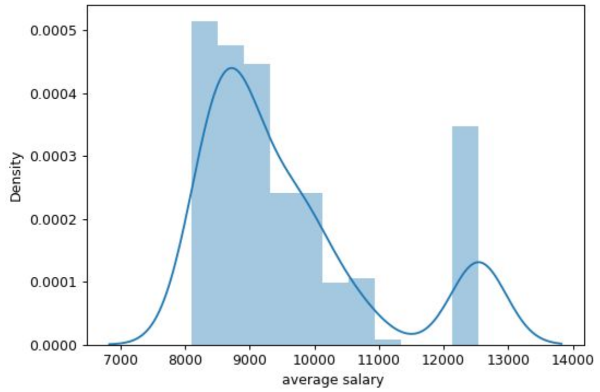
```
# Missing Values Cleaning
def missing_values_cleaning(df):

    # Drop columns with missing value percentage > 0.7
    df = df[ df.columns[ df.isnull().mean() < 0.7 ] ]

    # Drop rows    with missing value percentage > 0.7
    df = df.loc[ df.isnull().mean(axis = 1) < 0.7 ]

    return df
```

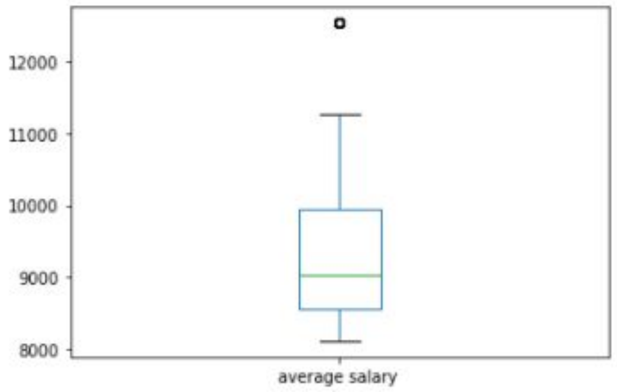


```
# Missing Values Treatment: for numerical columns, replace with custom value or median
def clean_numerical_missing_values(dataset, replacer=None):

    dataset = dataset.fillna(replacer if replacer is not None else dataset.median())
    return dataset
```

Data Preprocessing

- Over the data analysis there were some attributes of (the original) tables that appeared to have outliers. However these outliers may indicate that there is an intermediate subset of data that isn't present, therefore weren't erased.
- For example, in the case of the average salary (see images) it seems that the ones above 12000 could be considered outliers to be removed. But after a closer inspection, that 'dot' is made up of 46 entries, i.e., represent just over 14% of the data which is a considerable amount, and therefore, its removal might contribute to worse predictions.



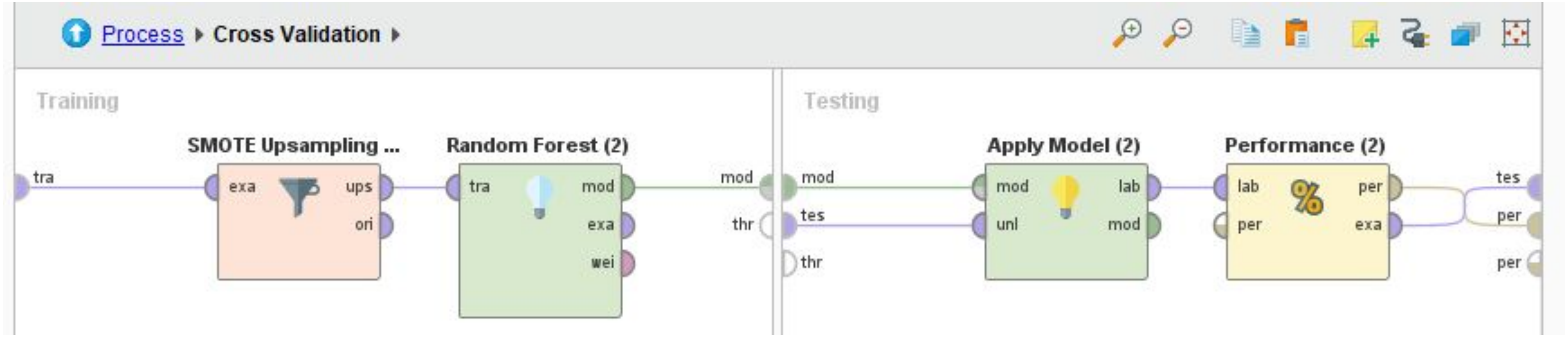


Data Preprocessing

- Of the provided features of the original isolated tables we didn’t identify any clearly redundant feature.
- After table joins, one of the features that was created to allow that aggregation was found to be redundant (number of transactions was directly connected to the number of transactions of each type, for example) and therefore was disregarded (number of transactions because the others provided at least the same degree of information).
- The data was also subject to encoding (discretization) and dummy encoding - some algorithms only allow numeric values. (Not applied in this stage, but normalization was used in the descriptive context to facilitate manual cluster attributes comparisons).
- Due to the already small size of the available data for training, we chose to use all of the data each time. We think that restricting training to even a smaller sample of data would not be fruitful in any way (46 cases of the positive class is already a very low amount).

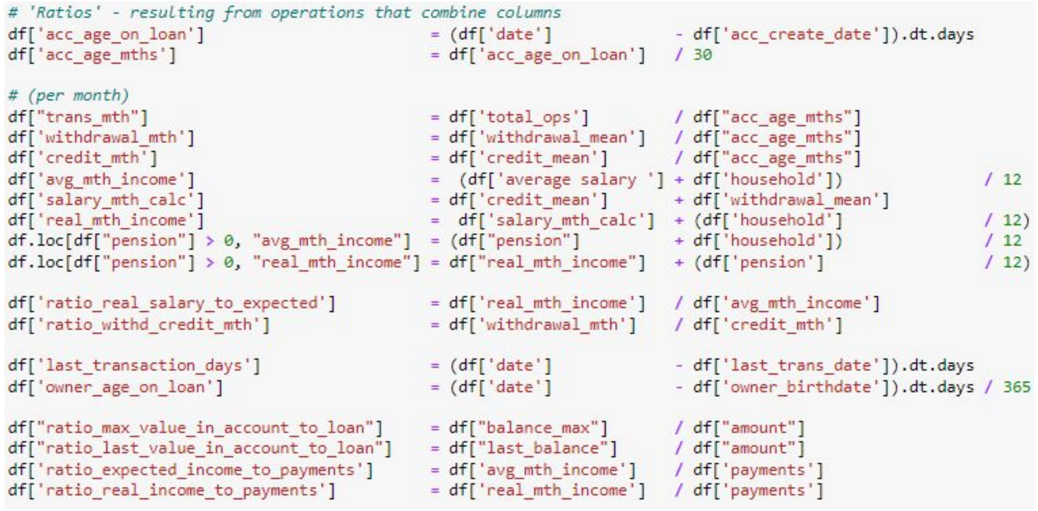
Data Preprocessing

- Imbalance in the data was detected and SMOTE was considered to deal with the situation, being applied only to the training subdivision of the data, in some of the experiments (as referred in the presentation).
- Example of SMOTE usage (in CV) in RapidMiner (available through the Operator Toolbox Extension):



Feature Engineering

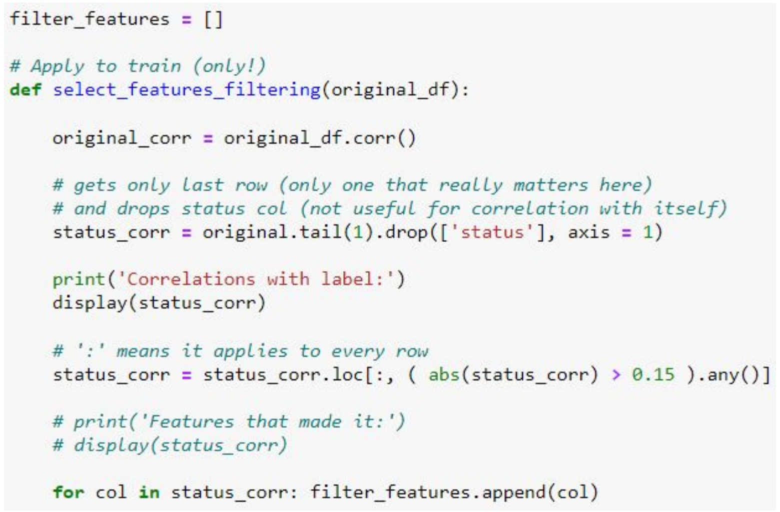
- Extensive feature creation (consult 'Pre processing revamped' Jupyter notebook for complete list). Based on basic statistics, ratios and simple 'yes or no' questions (based on possible real life contexts that would influence a person's opinion, e.g. balance having reached a negative value is assumed to be a red flag).
- Some examples of created features:
  - balance_mean
  - amount_min
  - col_another_bank
  - reached_negative_balance (1 or -1)
  - has_card (1 or -1)
  - acc_age_on_loan
  - unemployment_delta
  - log_crime_delta



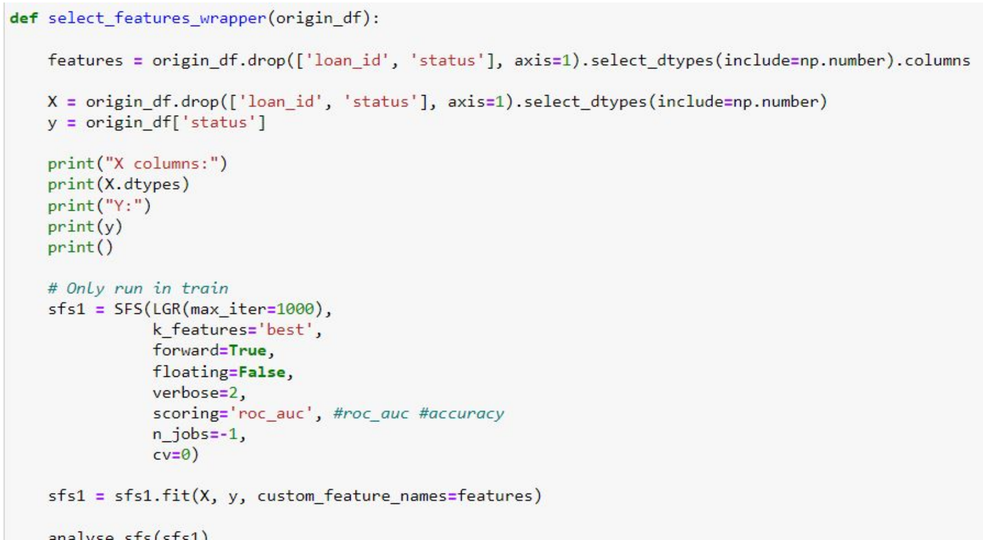
```
# 'Ratios' - resulting from operations that combine columns
df['acc_age_on_loan']                         = (df['date']              - df['acc_create_date']).dt.days
df['acc_age_mths']                            = df['acc_age_on_loan']    / 30

# (per month)
df["trans_mth"]                               = df['total_ops']          / df["acc_age_mths"]
df['withdrawal_mth']                          = df['withdrawal_mean']    / df["acc_age_mths"]
df['credit_mth']                              = df['credit_mean']        / df["acc_age_mths"]
df['avg_mth_income']                          =  (df['average salary '] + df['household'])             / 12
df['salary_mth_calc']                         = df['credit_mean']        + df['withdrawal_mean']
df['real_mth_income']                         =  df['salary_mth_calc']  + (df['household']             / 12)
df.loc[df["pension"] > 0, "avg_mth_income"]  = (df["pension"]           + df['household'])             / 12
df.loc[df["pension"] > 0, "real_mth_income"] = df["real_mth_income"]    + (df['pension']               / 12)

df['ratio_real_salary_to_expected']           = df['real_mth_income']    / df['avg_mth_income']
df['ratio_withd_credit_mth']                  = df['withdrawal_mth']     / df['credit_mth']

df['last_transaction_days']                   = (df['date']              - df['last_trans_date']).dt.days
df['owner_age_on_loan']                       = (df['date']              - df['owner_birthdate']).dt.days / 365

df["ratio_max_value_in_account_to_loan"]      = df["balance_max"]        / df["amount"]
df["ratio_last_value_in_account_to_loan"]     = df["last_balance"]       / df["amount"]
df['ratio_expected_income_to_payments']       = df['avg_mth_income']     / df['payments']
df['ratio_real_income_to_payments']           = df['real_mth_income']    / df['payments']
```

Feature Selection

- Manual and common-sense driven at the start.
- Evolved to Filter Method based on correlation threshold: feature chosen if correlation with label was higher than 0.15 (arbitrary value that seemed appropriate).
- Usage of Sequential Forward Selection (Wrapper Method) in a few experiments (due to being far more computational heavy / time consuming than the filter method because of the high number of features), taking advantage of Logistic Regression model for evaluation in order to find optimal features.



```python
filter_features = []

# Apply to train (only!)
def select_features_filtering(original_df):

    original_corr = original_df.corr()

    # gets only last row (only one that really matters here)
    # and drops status col (not useful for correlation with itself)
    status_corr = original.tail(1).drop(['status'], axis = 1)

    print('Correlations with label:')
    display(status_corr)

    # ':' means it applies to every row
    status_corr = status_corr.loc[:, ( abs(status_corr) > 0.15 ).any()]

    # print('Features that made it:')
    # display(status_corr)

    for col in status_corr: filter_features.append(col)
```



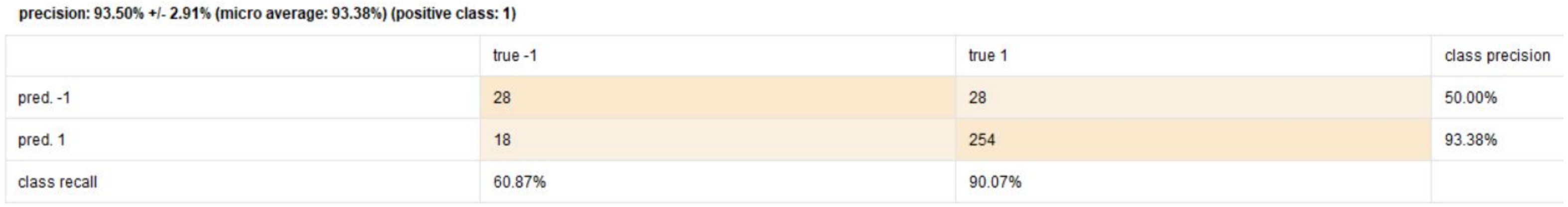
```python
def select_features_wrapper(origin_df):

    features = origin_df.drop(['loan_id', 'status'], axis=1).select_dtypes(include=np.number).columns

    X = origin_df.drop(['loan_id', 'status'], axis=1).select_dtypes(include=np.number)
    y = origin_df['status']

    print("X columns:")
    print(X.dtypes)
    print("Y:")
    print(y)
    print()

    # Only run in train
    sfs1 = SFS(LGR(max_iter=1000),
               k_features='best',
               forward=True,
               floating=False,
               verbose=2,
               scoring='roc_auc', #roc_auc #accuracy
               n_jobs=-1,
               cv=0)

    sfs1 = sfs1.fit(X, y, custom_feature_names=features)

    analyse_sfs(sfs1)
```

Predictive Task

- Instead of choosing algorithms at random or start using every single one that we came across, we decided to investigate what would be the most adequate ones for the project context to then try to improve via hyperparameter tuning, due to the rules imposed not only related to time until the end of the kaggle competition but also to the 1 submission limit per day.
- After acknowledging the Supervised Learning nature of the problem, we picked 4 main algorithms:
  - Logistic Regression
  - Random Forest (Ensemble Method)
  - Support Vector Machine
  - Naïve bayes
- RapidMiner and Python based approaches were both carried out but the former was by far the main tool that was utilized, because of its simplicity and time-efficiency.

Predictive Task - Naïve bayes

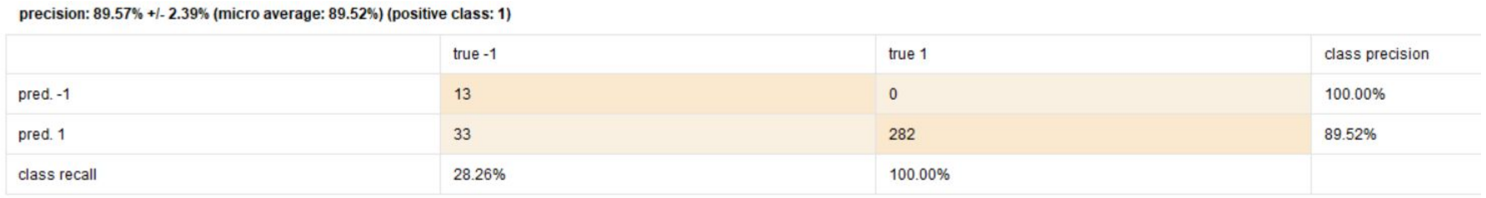
- Naïve bayes looked appropriate for early quick experiments to evaluate the data preparation process due to its speed and scalability with the high number of predictors and pretty much inexistence of parameters: within the RapidMiner operator (that uses Gaussian probability densities to model the attribute data) the only one was 'laplace correction' (which is a measure to avoid that the conditional probability is set to zero if an attribute value doesn't occur in the context of a given class).
- In the end, yielded good results locally (AUC: 0.887, Precision: 50.34%, Recall: 60.71%, f measure: 53.59%, specificity: 90.11%), but, as it assumes feature independency, its results were thought not to be in line with the score of a possible submission to kaggle.


precision: 93.50% +/- 2.91% (micro average: 93.38%) (positive class: 1)

	true -1	true 1	class precision
pred. -1	28	28	50.00%
pred. 1	18	254	93.38%
class recall	60.87%	90.07%	

Predictive Task - Support Vector Machine

- Support Vector Machine classifies the data by using a hyperplane (or set of hyperplanes) which acts like a decision boundary. The better the separation, the lower is the generalization error, i.e., the larger is the distance between the hyperplane to the nearest (training) data point of any of the classes. It is effective even with high dimensional data and has a great set of customizable parameters such as 'C' (which tells the SVM optimization how much you want to avoid misclassifying; the higher the value of C, the smaller is the chosen hyperplane margin that allows for correct classification).
- In the end, it too yielded good results locally (AUC: 0.882, Precision: 100.00%, Recall: 27.98%, f measure: 41.20%, specificity: 100.00%), but, as it has such a vast range of parameters (kernel that we didn't have enough time to test fully via hyperparameter tuning, it was left as a 'backup' model if needed.

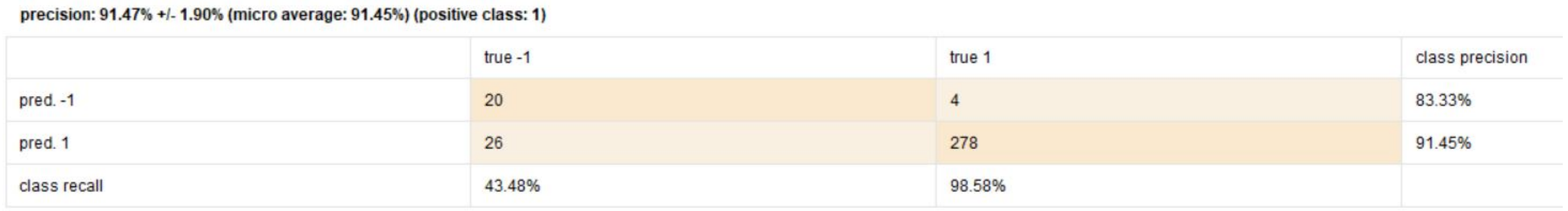


precision: 89.57% +/- 2.39% (micro average: 89.52%) (positive class: 1)

	true -1	true 1	class precision
pred. -1	13	0	100.00%
pred. 1	33	282	89.52%
class recall	28.26%	100.00%	

Predictive Task - Random Forest

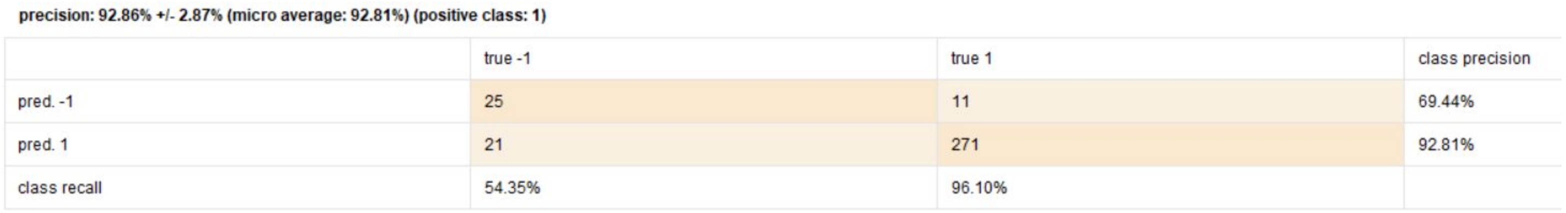
- Random Forest is an ensemble technique that builds a multitude of decision trees at training time. It also has the power to handle data with high dimensionality and if there are enough trees in the forest, overfitting is not expected to be an issue. However, it is a **black-box** model (harder to explain) and parameters such as pruning and pre-pruning, maximal depth and number of trees can also influence greatly the training time and results.
- In the end, after some tuning, yielded better results locally (AUC: 0.900, Precision: 85.56%, Recall: 47.92%, f measure: 60.80%, specificity: 98.94%) and it was the model in which our best public (second best private) leaderboard submission was based on (public: 0.97530, private: 0.93827).


precision: 91.47% +/- 1.90% (micro average: 91.45%) (positive class: 1)

	true -1	true 1	class precision
pred. -1	20	4	83.33%
pred. 1	26	278	91.45%
class recall	43.48%	98.58%	

Predictive Task - Logistic Regression

- Logistic Regression is a process of modeling the probability of a discrete outcome given an input variable. It differs from linear regression primarily by its range that is bounded between 0 and 1. $$\log \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m$$
- Based on sigmoid function, logistic regression is relatively easy to interpret and very efficient to train.
- The vast majority of parameters were tinkered with. The best scoring solver was L_BFGS - way of finding (local) minimum, making use of objective function values and gradient -, due to its characteristics of scaling better for datasets with several columns, opposing IRLSM which is better for small number of predictors.
- It was the model that represented our best private leaderboard submission (public: 0.92798, private: 0.94650) which was in line with the locally obtained performance metrics: AUC: 0.906, Precision: 75.46%, Recall: 54.46%, f measure: 60.34%, specificity: 96.12%.

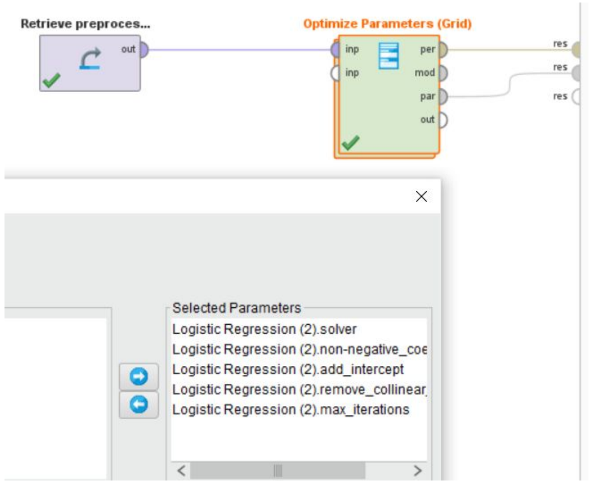

precision: 92.86% +/- 2.87% (micro average: 92.81%) (positive class: 1)

	true -1	true 1	class precision
pred. -1	25	11	69.44%
pred. 1	21	271	92.81%
class recall	54.35%	96.10%	

Predictive Task - Parameter Tuning and Performance Eval.

Hyperparameter Tuning

- In RapidMiner this was made possible using the “Optimize Parameters (Grid)” operator (equivalent to the popular GridSearch in Python) - that simulates every combination of the value ranges specified -, inside of which there is a Cross Validation (5-fold) where the model is applied to the training subset of the input dataset.



Selected Parameters
Logistic Regression (2).solver
Logistic Regression (2).non-negative_coe
Logistic Regression (2).add_intercept
Logistic Regression (2).remove_collinear
Logistic Regression (2).max_iterations

Performance Evaluation

- The main performance metric was AUC as it was defined in the data mining goals and directly gave accurate insight to the competition scores. Nonetheless, and as mentioned in the presentation, others such as accuracy, classification error and specificity were also considered.

Predictive Task - Performance Evaluation

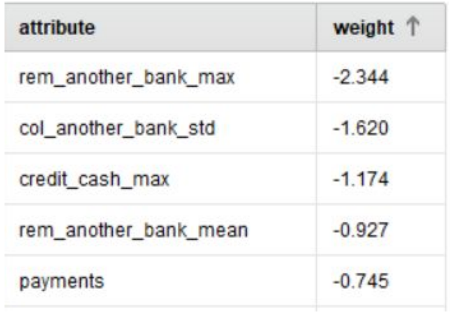
Performance Evaluation Analysis

- By analysing the performance of the (tuned) Logistic Regression we can the quality of the model. 90% of its predictions were correct and the AUC confirms the ability to successfully distinguish between the classes in most cases.
- If we look closer, we understand that the accuracy doesn't show the whole picture. Most likely due to the higher number of examples in the training set, the negative class (1) is detected more easily (specificity) than the positive class (-1) - recall metric is lower.
- In terms of reliability, these results, despite seeming convincing, should be taken with caution due to the size of the positive class sample and std deviation (degree of values' dispersion; obtained from CV).

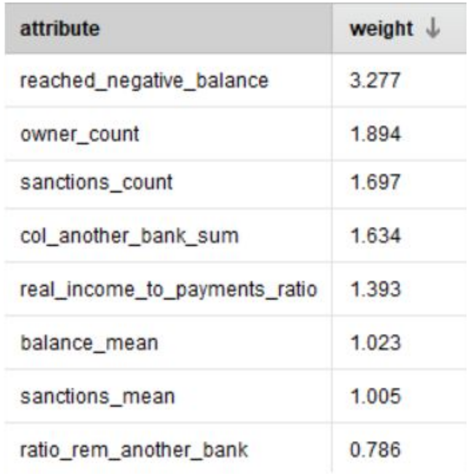
Metric	Value (CV mean)	Std Dev
Accuracy	**90.25%**	4.40%
Classification Error	9.75%	4.40%
AUC	**0.906**	0.028
Precision	75.46%	**23.99%**
Recall	**54.46%**	**18.96%**
F Measure	60.34%	17.64%
Specificity	**96.12%**	4.11%

Predictive Task - Feature Importance

- Analysing the weights that the Logistic Regression assigned to each feature (screenshots show the ones with absolute value over 0.7) we can come up with some observations. For example, an account with a higher maximum amount across his/hers transactions where the operation performed is a remittance to another bank is associated with a higher chance of paying the loan while an account that has ever reached negative balance is less likely to fulfill the 'contract'.
- These remarks fall in line with common sense and knowledge (for example, a person would rather lend money to someone that has always been on the 'green' than to someone who has already had in the past a negative account balance).
- Logistic Regression, being a **white-box** model, allows for a better understanding of the inner workings of the algorithm behaviour, more specifically the coefficients that were applied to each feature to produce the final predictions.



attribute	weight ↑
rem_another_bank_max	-2.344
col_another_bank_std	-1.620
credit_cash_max	-1.174
rem_another_bank_mean	-0.927
payments	-0.745



attribute	weight ↓
reached_negative_balance	3.277
owner_count	1.894
sanctions_count	1.697
col_another_bank_sum	1.634
real_income_to_payments_ratio	1.393
balance_mean	1.023
sanctions_mean	1.005
ratio_rem_another_bank	0.786

Descriptive Task

- The task of finding client profiles can be translated to a clustering problem (unsupervised learning problem).
- For this effect, the main algorithm we used was k-Means, but other methods were also explored such as k-Medoids, both using RapidMiner blocks.

- Our goal was to identify progressively more detailed profiles:
  1. Individual profiles - clustering based only on data from the client table.
  2. Socio-demographic profiles - including both client and district tables.
     1. Small variation adding card (and consequently account) data.
  3. Financial and Socio-demographic profiles - including data from all the previously mentioned tables plus transactions and loans information.

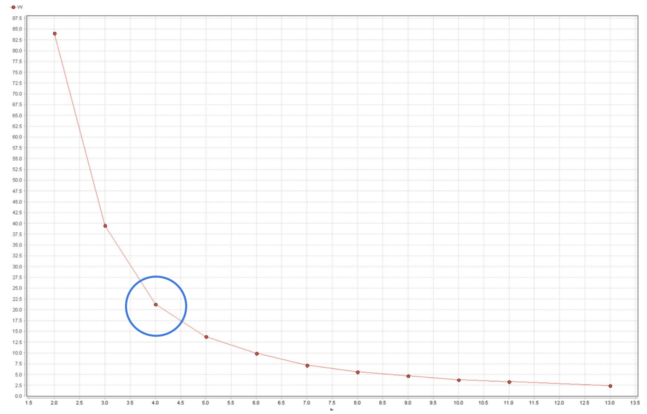
Descriptive Task - k-Means and k-Medoids

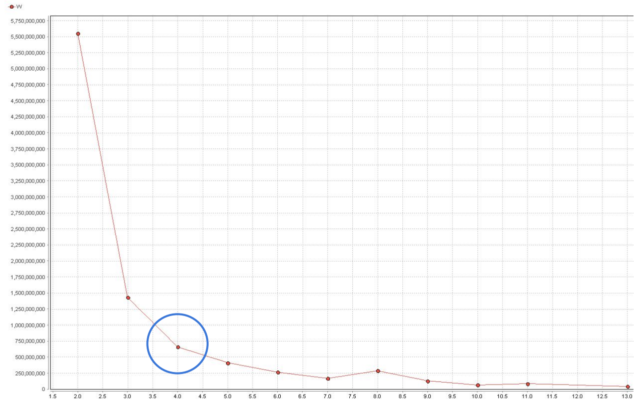
- The algorithm k-Means aims to partition the data into k clusters, where each observation corresponds to the cluster that has the nearest mean (cluster centroid). Despite the fact that the number of clusters needs to be manually chosen and that has trouble clustering data where clusters are of varying density and sizes, it assures convergence and can scale to large data set and ‘warm-starts’ the position of the centroids.

- k-Medoids (also known as Partitioning Around Medoid) is similar to k-Means in that it attempts to minimize the distance between points labeled to be in a cluster and the center of the respective cluster (medoid - defined as the one whose average dissimilarity to all other cluster elements is minimal). However, it chooses actual data points as the centers, facilitating the results understanding and it is more robust to outliers and noise than k-means.

Descriptive Task - Parameter Tuning and Performance Eval.

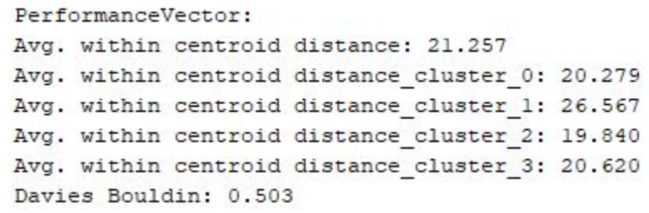
- Both algorithms mentioned previously require that the number of clusters to group up the data has to be specified *a priori*.
- Therefore, a parameter optimization operator was used in RapidMiner, varying the value of k (Elbow method).
- On the right, there are two graphs that illustrate the variation of the average cluster distance (W) with the increase of the number of clusters (k, X axis) in the k-Means algorithm, being the top one correspondent to clustering made at the individual level and the bottom one to socio-demographic clustering.
- In both cases, by applying the Elbow heuristic, the ideal k value can be considered 4 (identical results when using k-Medoids but significantly heavier computationally).





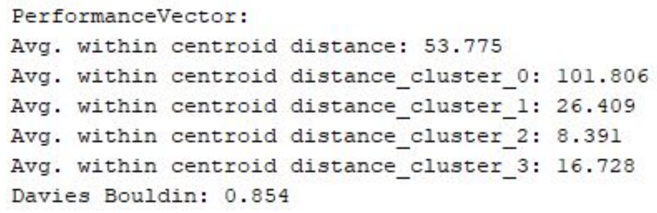
Descriptive Task - Performance Evaluation

- As evaluation metrics we used the average within centroid distance for each cluster and the Davies-Bouldin criterion which is based on a ratio between “within-cluster” and “between-cluster” distances.
- The average within centroid distance in itself didn’t provide any concrete information but allowed for comparison between parameter values (e.g. between k=3 and k=4) and between algorithms.
- The DB criterion, on the other hand, allows more objective evaluation: algorithms that produce clusters with high inter-cluster distances (low inter-cluster similarity) and low intra-cluster distances (high intra-cluster similarity) will have a low Davies-Bouldin index. The clustering algorithm that produces clusters with the smallest Davies-Bouldin index is considered the best one based on this criterion.



```
PerformanceVector:
Avg. within centroid distance: 21.257
Avg. within centroid distance_cluster_0: 20.279
Avg. within centroid distance_cluster_1: 26.567
Avg. within centroid distance_cluster_2: 19.840
Avg. within centroid distance_cluster_3: 20.620
Davies Bouldin: 0.503
```

Performance Results for k-Means (k=4), individual profile

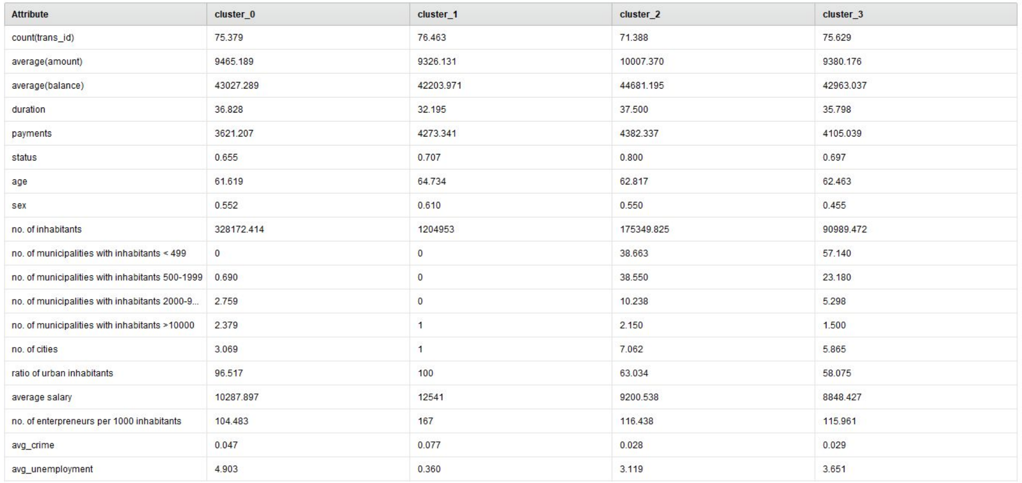


```
PerformanceVector:
Avg. within centroid distance: 53.775
Avg. within centroid distance_cluster_0: 101.806
Avg. within centroid distance_cluster_1: 26.409
Avg. within centroid distance_cluster_2: 8.391
Avg. within centroid distance_cluster_3: 16.728
Davies Bouldin: 0.854
```

Performance Results for k-Medoids(k=4), individual profile

Descriptive Task - Results

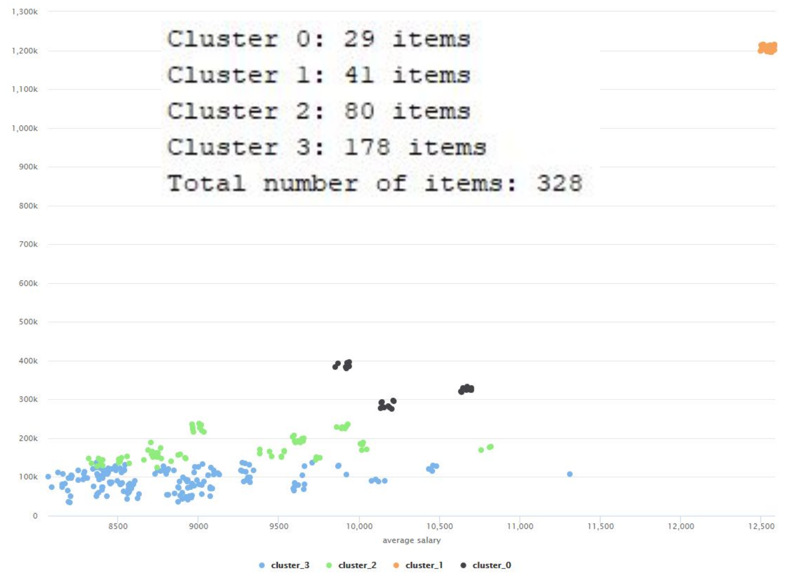
- For the most complex subtask (Financial and Socio-demographic profiles) there were obtained 4 clusters (k-Means, k value obtained via Elbow Heuristic) that confirm the conclusions obtained during the data analysis segment (e.g. densely urbanized and populated areas are related with higher salaries). Main distinguishable characteristics:
- Cluster 0 (29 items): lowest likelihood of paying back a loan and highest unemployment rate.
- Cluster 1 (41 items): highest populated area residents
- Cluster 2 (80 items): highest likelihood of paying back a loan and lowest unemployment rate.
- Cluster 3 (178 items): people that live in the less densely populated areas and with lower salaries.
- Therefore, of all groups, cluster 2 encompasses the 'best behaved' possible clients.



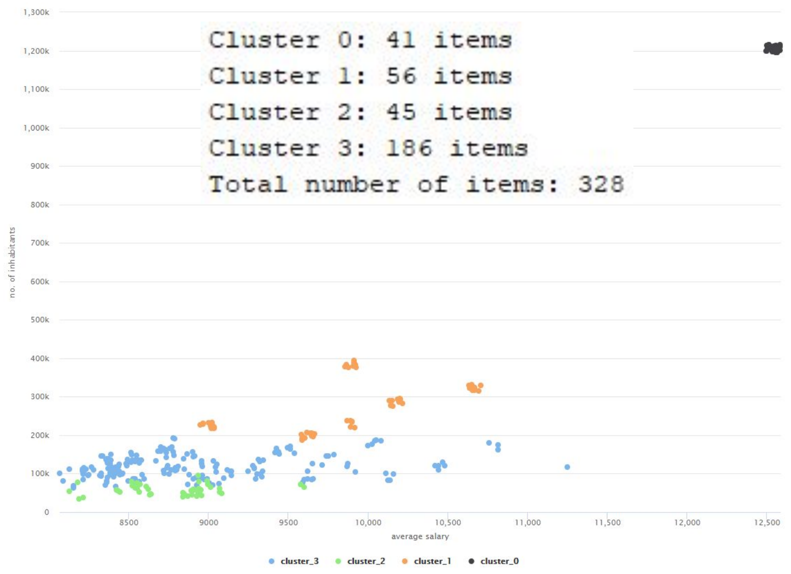
Attribute	cluster_0	cluster_1	cluster_2	cluster_3
count(trans_id)	75.379	76.463	71.388	75.629
average(amount)	9465.189	9326.131	10007.370	9380.176
average(balance)	43027.289	42203.971	44681.195	42963.037
duration	36.828	32.195	37.500	35.798
payments	3621.207	4273.341	4382.337	4105.039
status	0.655	0.707	0.800	0.697
age	61.619	64.734	62.817	62.463
sex	0.552	0.610	0.550	0.455
no. of inhabitants	328172.414	1204953	175349.825	90989.472
no. of municipalities with inhabitants < 499	0	0	38.663	57.140
no. of municipalities with inhabitants 500-1999	0.690	0	38.550	23.180
no. of municipalities with inhabitants 2000-9...	2.759	0	10.238	5.298
no. of municipalities with inhabitants >10000	2.379	1	2.150	1.500
no. of cities	3.069	1	7.062	5.865
ratio of urban inhabitants	96.517	100	63.034	58.075
average salary	10287.897	12541	9200.538	8848.427
no. of enterpreneurs per 1000 inhabitants	104.483	167	116.438	115.961
avg_crime	0.047	0.077	0.028	0.029
avg_unemployment	4.903	0.360	3.119	3.651

Descriptive Task - Results (Algorithm Comparison)

- Visual comparison of the clustering in terms of average salary and number of inhabitants of district for algorithms k-Means (left) and k-Medoids (right) for the same value of k (4). Both algorithms isolate the people that live the highest populated area, but differ in the distribution of the rest among the other 3 clusters.



```
Cluster 0: 29 items
Cluster 1: 41 items
Cluster 2: 80 items
Cluster 3: 178 items
Total number of items: 328
```



```
Cluster 0: 41 items
Cluster 1: 56 items
Cluster 2: 45 items
Cluster 3: 186 items
Total number of items: 328
```

Project Management / Tools

- The project development was somewhat based on the Cross-industry standard process for data mining (CRISP-DM)
- A dynamic ‘plan of attack’ was continuously discussed and updated over the course of the project:
    1. Focus started in Business Understanding, where the business/data mining goals were loosely described.
    2. Exploratory Data Analysis.
    3. Some Data Preparation still with little feature engineering and manual selection.
    4. Early model training results didn’t match our expectations and after hyperparameter tuning we took a step back and started improving the previous stages.
    5. Business and Data Mining goals became clear.
    6. Data Preparation revamping - highlighting the generation of many features and feature selection
    7. Models started to give the desired results and extra model comparison, evaluation and hyperparameter tuning improved the outcomes of the submissions.
    8. Lastly, the descriptive problem became the focus and clustering was applied to find types of clients.

Tools and Individual Factor

- Tools utilized:
  - Tasks Management / Distribution: Issues on Project Board (Github).
  - Collaboration: Discord, Git and Google Colab.
  - RapidMiner, Python (Jupyter notebooks and libraries - pandas, matplotlib, seaborn, numpy, sklearn -, mainly DU and DP) and Excel (DU).
- Note: for extra details and/or implementations, consult Jupyter notebooks and RapidMiner files (experiment3 and experiment5 correspond to the 2 picked submissions for the Kaggle competition).
- All elements of the group put in the effort and contributed in an equal manner for the development of the project.

Name and Number	Contribution [0-1]
Diogo Rodrigues (up201806429)	1
João Sousa (up201806613)	1
Rafael Ribeiro (up201806330)	1