



To loan or not to loan that is the question

Machine Learning 2021/2022

André Gomes - up201806224
André Nascimento - up201806461
Leonor Gomes - up201806567

Table of contents

**01. Domain
Description**

**02. Exploratory
Data Analysis**

**03. Descriptive
Problem**

**04. Predictive
Problem**

05. Conclusions

06. Annexes



01.

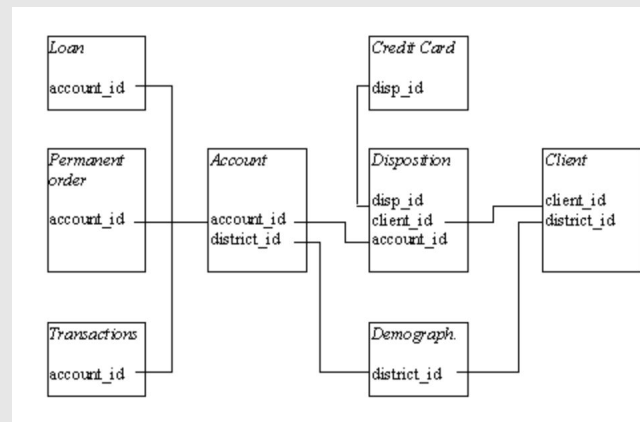
Domain Description


Banking

This case study focuses on analyzing records from a bank. These include information about the accounts, clients, payments, transactions, credit cards, demographic data and loan history.

There are:

- **4500 accounts** that can be accessed by more than one client;
- **5369 clients**;
- **682 loans**.



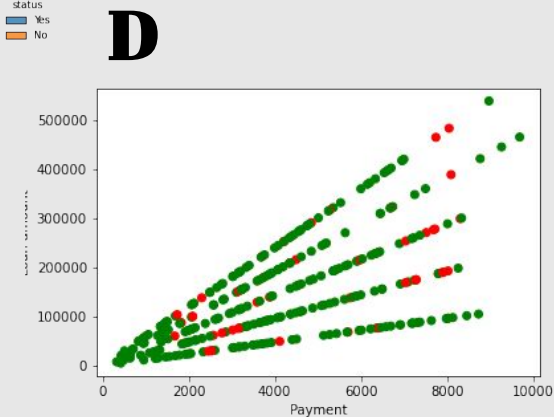
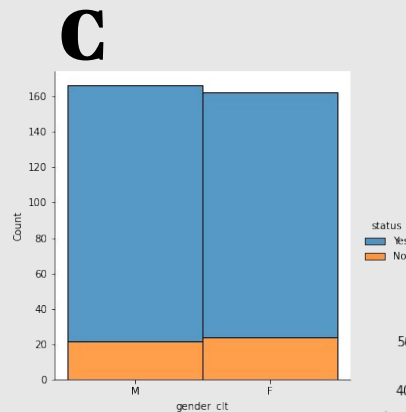
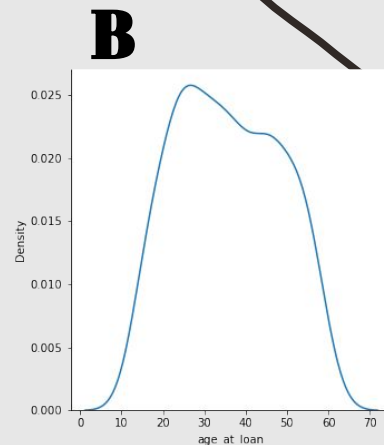
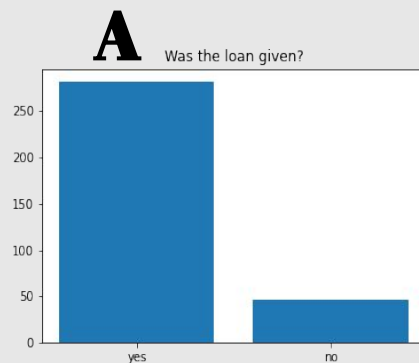


02.

Exploratory Data Analysis

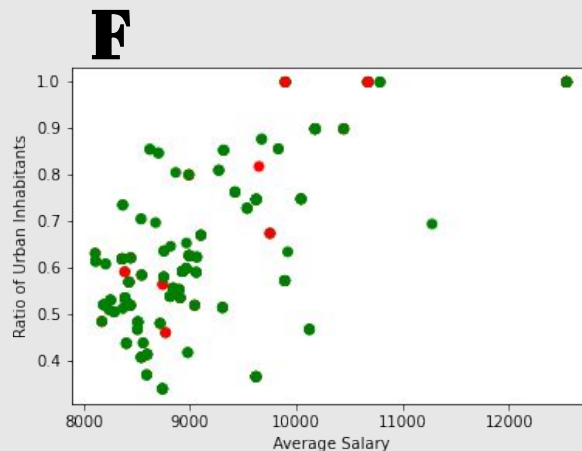
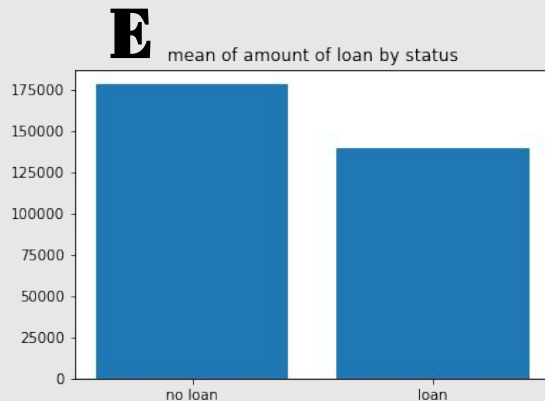
Data Analysis 1/2

- For most cases (86%), the loan was given. Therefore, this can be considered an unbalanced dataset, making it harder to predict when loans are not given. - **A**
- Most people asking for a loan are between 30-40 years at the time of the request. - **B**
- All clients that own cards had a successful loan. - **1**
- The gender is irrelevant for receiving a loan. - **C**
- The bigger the payment, the bigger the loan amount. - **D**



Data Analysis 2/2

- Mean of the amount of loan of non accepted loans was bigger than the ones accepted. - **E**
- Mean of the payments of non accepted loans was also bigger than the ones accepted. - **2**
- The most common type of card is “Classic”. - **3**
- The most common frequency of accounts is “monthly issuance”. - **4**
- Usually there is a higher average salary for higher ratio of urban inhabitants. - **F**
- The most common operation is “Credit in cash” followed by “Collection from another bank”. - **5**





03.

Descriptive Problem

The **goal** is to identify different groups of clients that represent relevant behaviours for the bank.

Cluster comparison

2



X-Means

K varying between 2 and 30

1 second

10

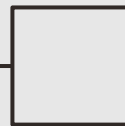


k-Means

Grid Search for value of k

5,5 seconds

10737



Agglomerative

Bottom-up strategy for
Hierarchical clustering

6:20 minutes

Number of Clusters: 2

Cluster 0

4.706

region = Prague is on average **100.00%** smaller, **no. of committed crimes '96** is on average **75.39%** smaller, **no. of committed crimes '95** is on average **72.13%** smaller

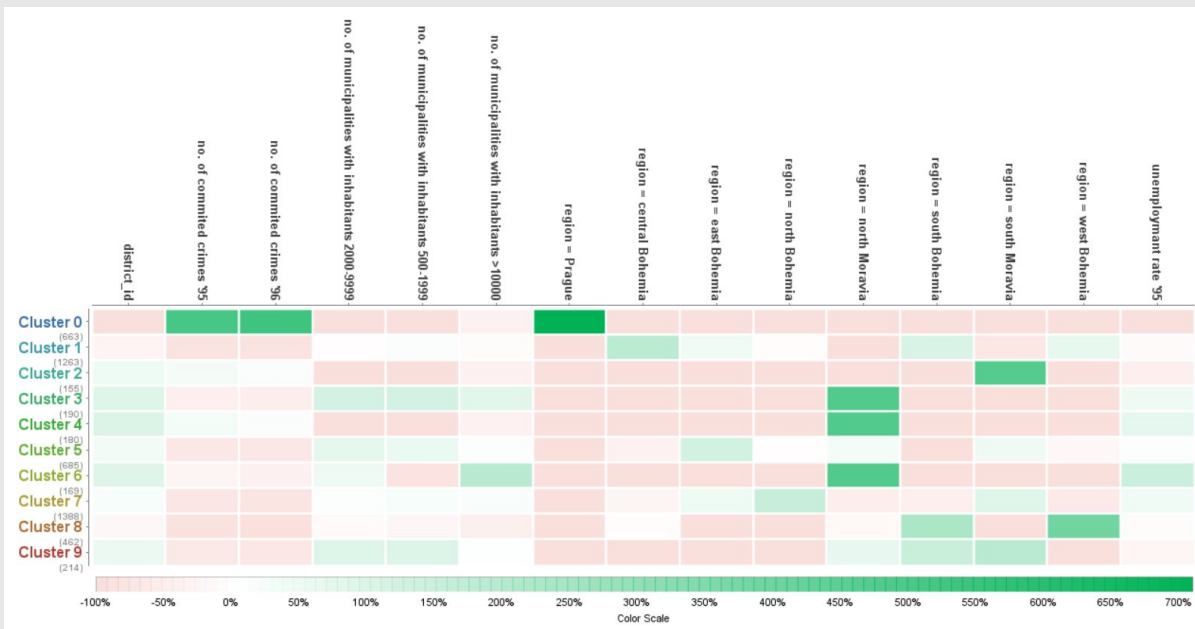
Cluster 1

663

region = Prague is on average 709.80% larger, no. of committed crimes '96 is on average 535.14% larger, no. of committed crimes '95 is on average 512.01% larger

There is a large percentage of clients that are from Prague, and the crime rate in that city is also higher than in others.

From these results, instead of having a feature that determines the region where a client resides, we could have a binomial value that represents if the client lives in Prague or not. Simplifying the data, but keeping important information for the model.





04.

Predictive Problem

The **goal** is to predict whether to give a loan to a client or not.

It's a **classification** problem where the output variable is **1** (loan successful) or **-1** (loan not successful).

Dimensions of Data Quality

Accuracy

Data was given, methods of obtention unknown, should be accurate



Timeliness

Data between 1992 and 1997. Not relevant today

Completeness

Row: More records -1 than 1

Column: k-symbol, bank, account, district
Missing table permanent order



Uniqueness

ID's validated and all unique

Consistency

Clients:5369; Cards:202;

Accounts:4500; Loans:682;

Status should be A/B/C/D but is 1/-1



Validity

Missing values and withdrawal in cash

Data Preparation

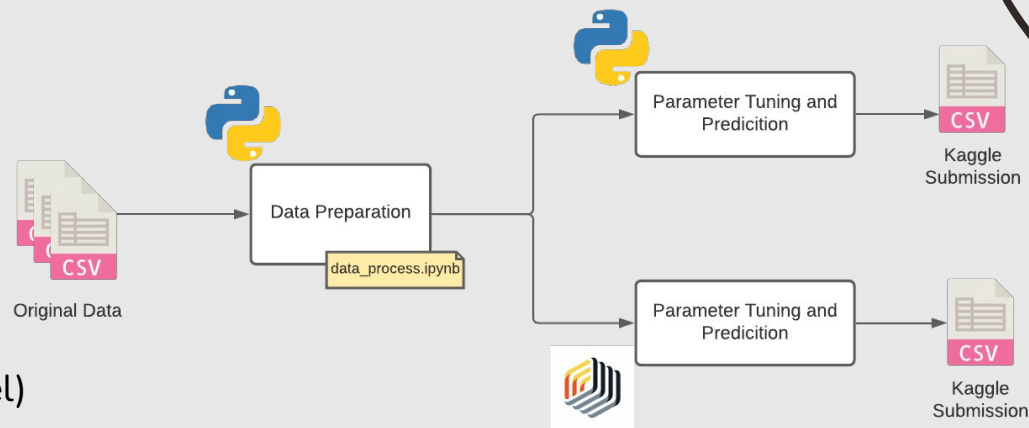
- Process each table and merge (except cards)
- Extract client's birthdate/age and gender from `birth_number`
- Conversion of dates to 'yyyy-mm-dd' format
- Only keep rows with type 'OWNER' in *disposition* table
- Replace cells with '?' in both *no. of committed crimes '95* and *unemployment rate '95* with median values
- Fill *null* values in *operation* column (*transaction* table) with 'interest credited'
- Convert type 'withdrawal in cash' to just 'withdrawal' in the transactions
- Drop ids
- Drop irrelevant features from district and transaction tables (`k_symbol`, `bank`, `account`,)
- Feature engineering (next slide)
- Drop highly correlated features

Feature Engineering

- Client's age at loan
- Account's months at loan
- Criminality and unemployment growth (from '95 and '96 values)
- Transaction data aggregation with statistical methods:
 - Operations count
 - Withdrawal and Credit count and ratios
 - Balance mean, min, abs_min, max, std, last, cov
 - Amount mean, min, abs_min, max, std, last, cov
- Months until bankrupt
- Operations per month

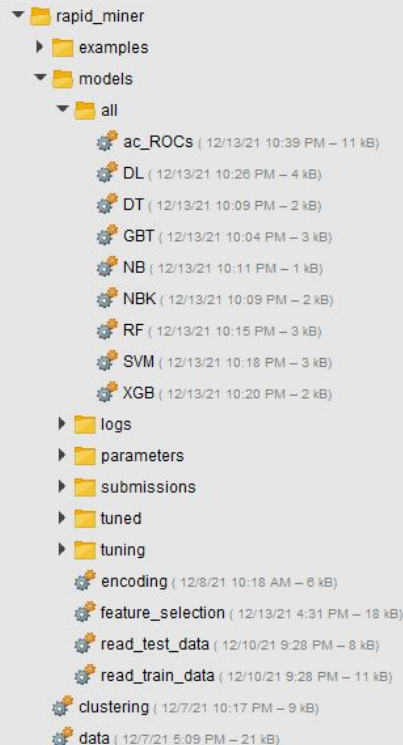
Experimental Setup - Pipeline

- Data normalization
- Parameter Tuning - **GridSearch** with **CV**
- Oversampling (**SMOTE**)
- Feature Selection (**MRMR** and **RFE**)
- Predict on different Models:
 - Decision Trees
 - Gradient Boosted Trees
 - Random Forest
 - XGBoost
 - SVM
 - Naive Bayes (simple and kernel)
 - Deep Learning
- Submit best predictions to Kaggle

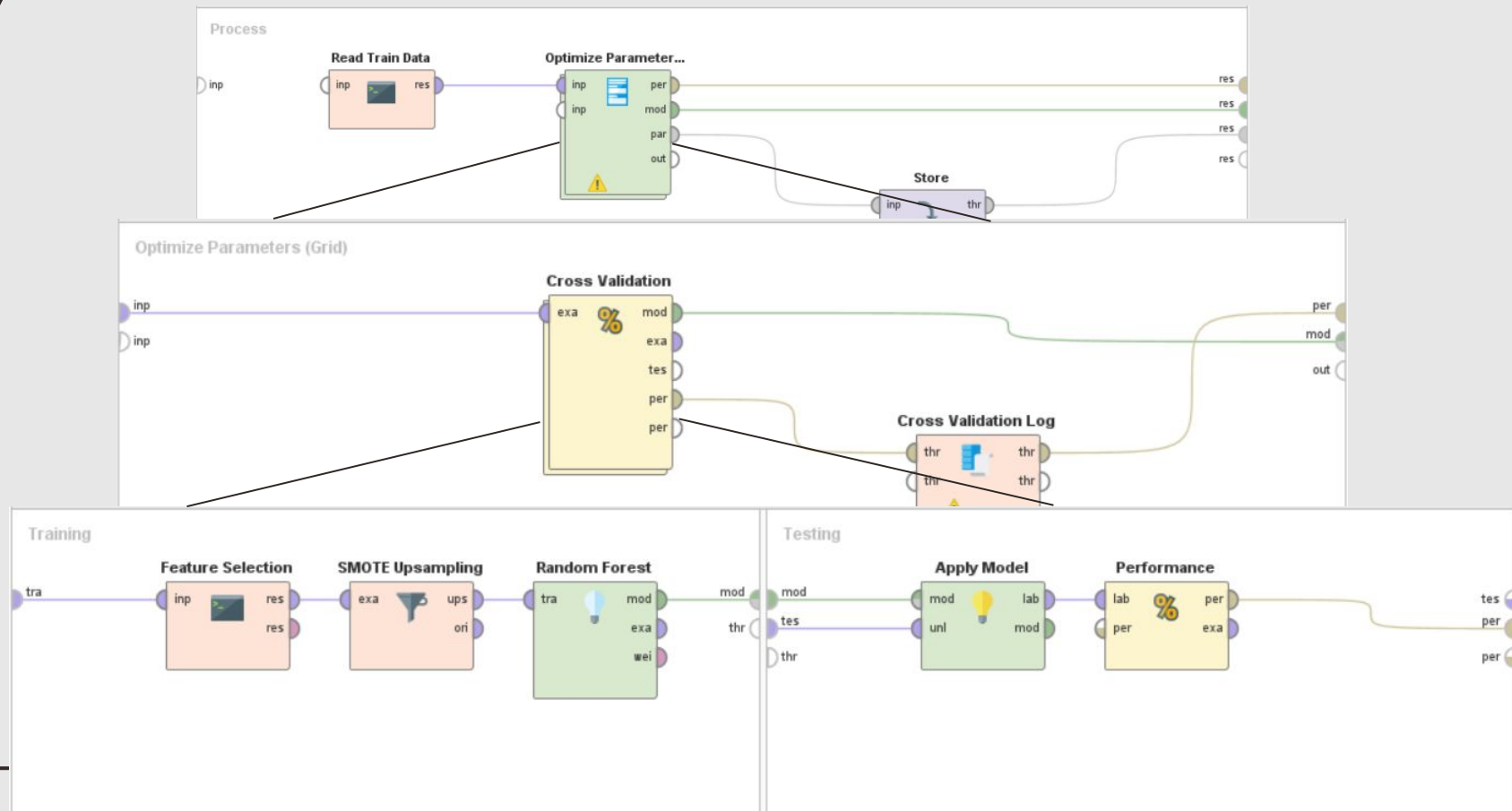


Experimental Setup - RapidMiner

- Clean data is received from the Python Notebook
- Normalizing data with **Z-Transformation**
- Encoding data, when required by the Models, using **Dummy Coding**
- Log procedures are present to save parameters and results from the Hyperparameter Tuning
- Models tested are on the image



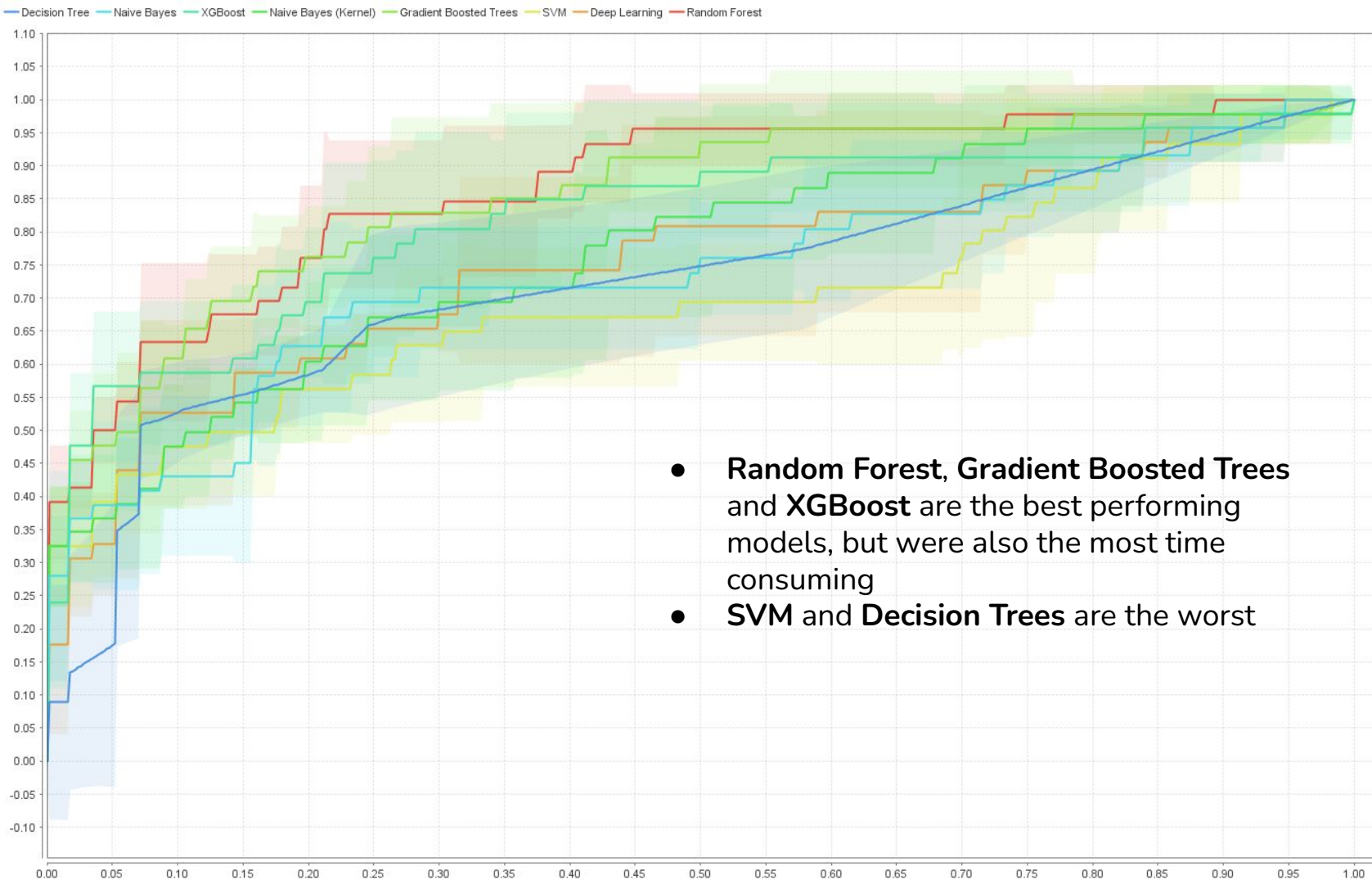
Experimental Setup - RapidMiner Example



Results

- Main metric used to evaluate the results was **AUC** (Area Under the Curve) since it is the scoring method used in Kaggle, but **accuracy** and **f-measure** were also taken into consideration.
- Locally getting consistent score of **87%** AUC with **GBT**, **RF** and **XGB**.
- Performing better on the public leaderboard. Some submissions had an increase of 0.10, concern about overfit.
- Upsampling and Feature Selection increased slightly the score.
- Feature Selection improved drastically tuning time.

Classifier	AUC	f-measure	Execution Time
RF	0.861	51.71%	28:43
RF with SMOTE	0.868	52.29%	38:00
RF with SMOTE and MRMR	0.872	53.41%	13:27



Results - Kaggle

- RF and XGB submissions selected for the final score
- Kaggle's public score is a lot higher than the local in some cases
- Local AUC score consistent with Kaggle's private score

XGB.csv 11 days ago by André Nascimento -----accuracy: 87.80% +/- 1.57% ****AUC: 0.851 +/- 0.108 -----precision: 57.71% +/- 8.84% ----recall: 56.67% +/- 14.91% ---- f_measure: 55.98% +/- 7.05%	0.86399	0.84218	<input checked="" type="checkbox"/>
RF.csv 8 days ago by G20 -----accuracy: 87.80% +/- 2.43% *****AUC: 0.872 +/- 0.098 -----precision: 59.33% +/- 14.18% ----recall: 50.22% +/- 10.84% ----f_measure: 53.41% +/- 8.68%	0.88230	0.96008	<input checked="" type="checkbox"/>
XGB.csv 13 days ago by André Nascimento -----accuracy: 89.01% +/- 3.70% *****AUC: 0.853 +/- 0.101 -----precision: 69.11% +/- 22.94% ----recall: 52.44% +/- 15.28% ----f_measure: 57.13% +/- 11.21%	0.91975	0.96090	<input type="checkbox"/>



05.

Conclusions

Conclusions and future work



Data

Methods for data
exploration



Progress

Make the Pipeline, but
after it follow the correct
steps



Methods

Don't use methods
blindly, know first what
they do

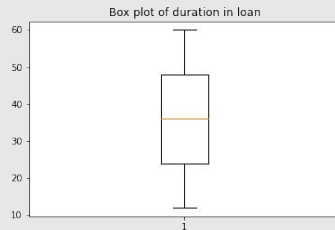
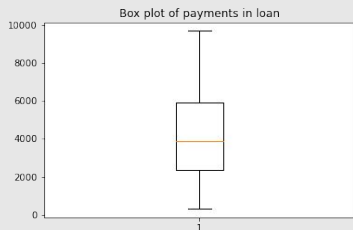


06.

Annexes

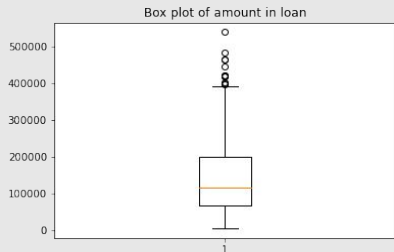
Box plots

Box plots are great ways to detect outliers in the data. They were used to analyse in which parameters there could be more outliers.

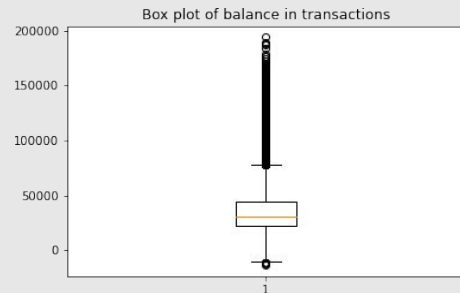
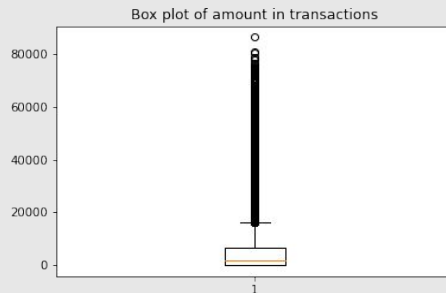


No outliers

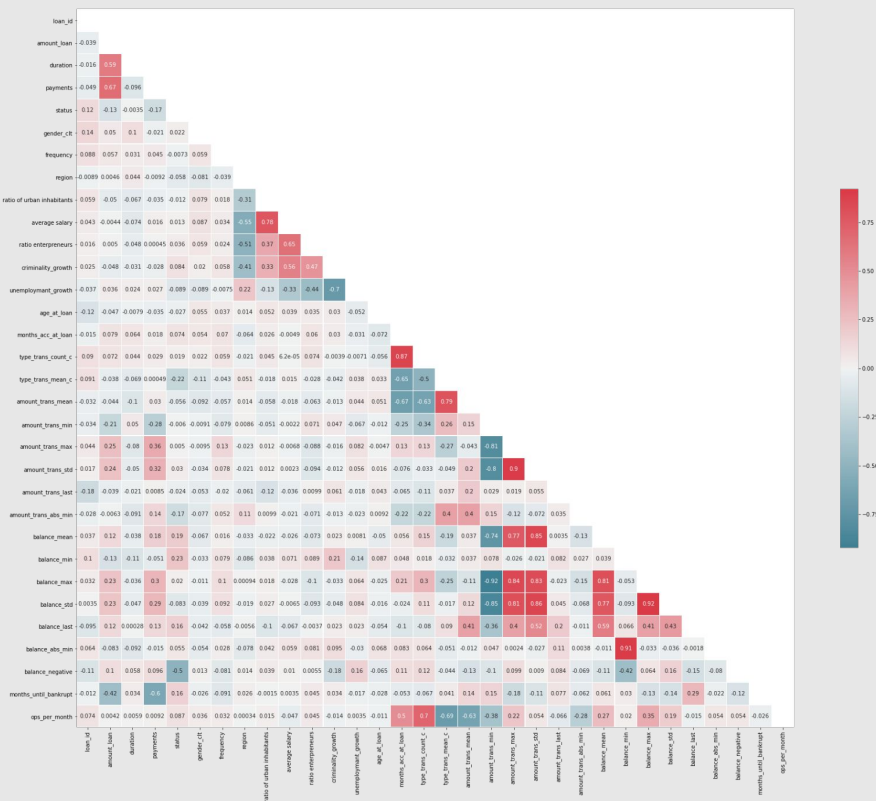
A lot of outliers



Some outliers

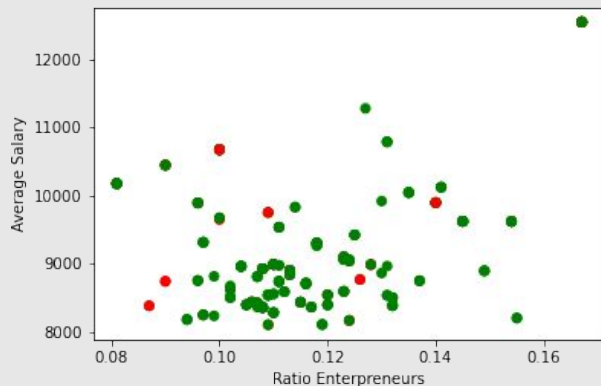


Correlation Diagram

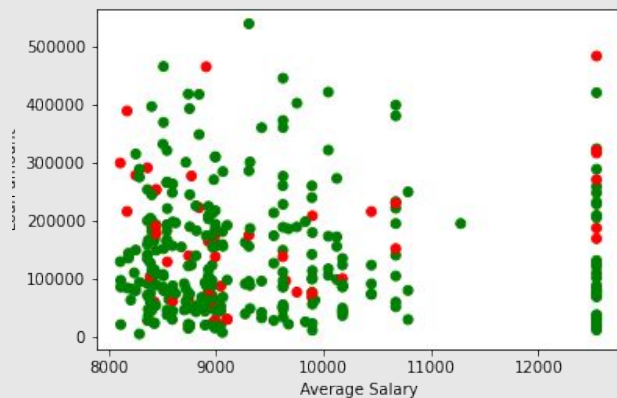


Calculating the correlation among parameters helps a lot to understand which features you can remove before applying the model. For features that have a correlation between 0.95 and 1, it doesn't really make sense to have both of them to predict the results.

3D Scatter Plots 1/2



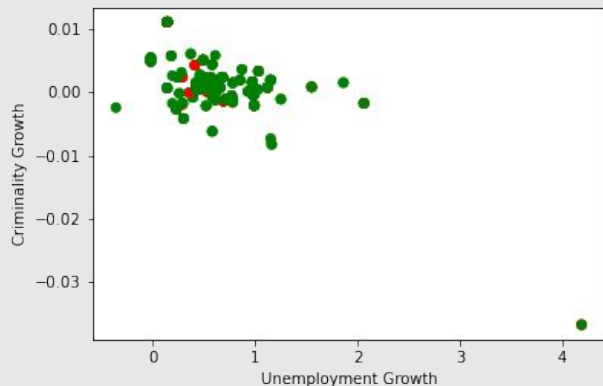
This scatter plot relates the **ratio entrepreneurs** and the **average salary**. Visually there is no strong correlation between these two.



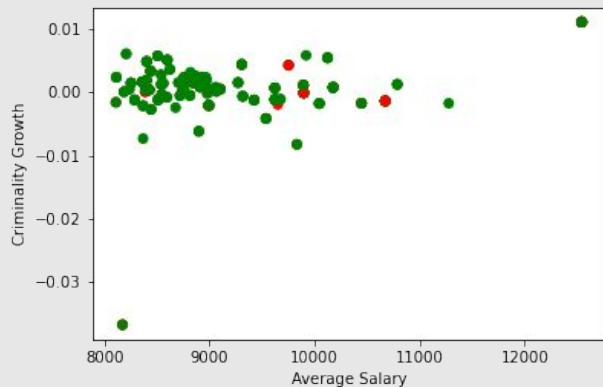
This graph puts together the **average salary** and the **loan amount**. From that, we can see that there is no direct correlation between them.

Explanation: For these graphs, if it's green, it means that the loan was given and if it's red, it means that the loan was not given.

3D Scatter Plots 2/2

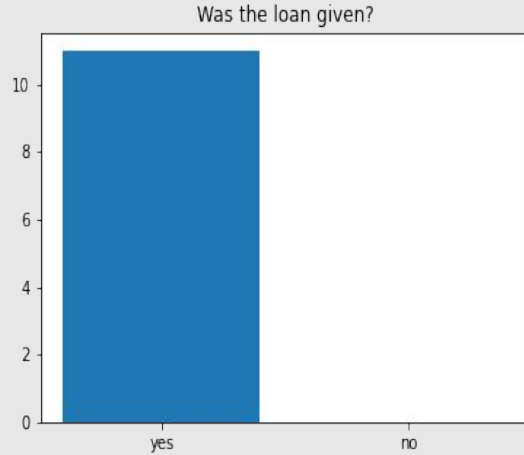


This scatter plot puts together the **unemployment growth** and the **criminality growth**. Contrary to what one might believe, visually there is no strong correlation between these two.

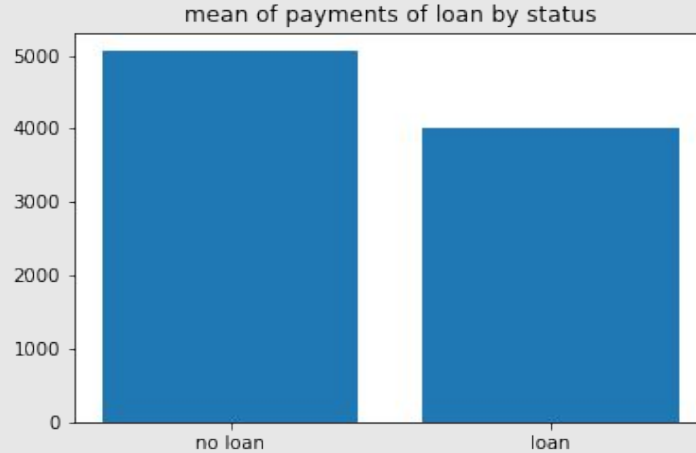


This graph shows the **average salary** and the **criminality growth**. Even though one might think they could be strongly correlated, there is no indication of that in this image.

Bar Plots 1/3



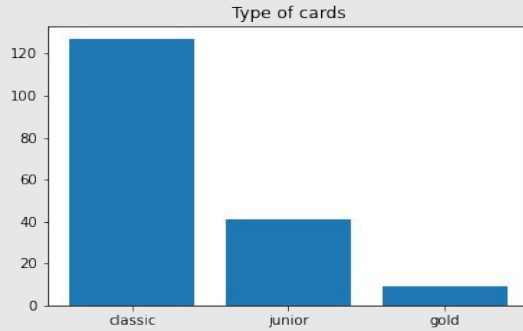
1



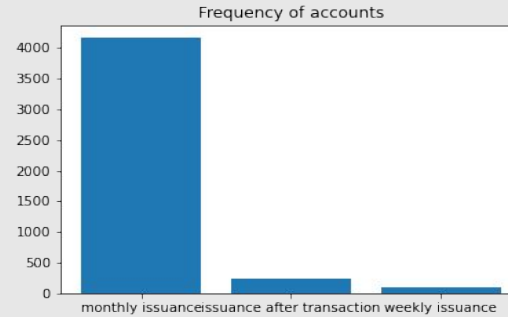
2

These graphs correspond to the rest of the conclusions in the “Exploratory Data Analysis” section

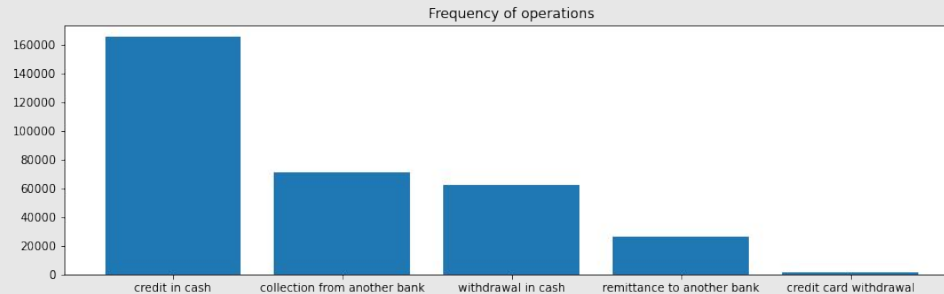
Bar Plots 2/3



3



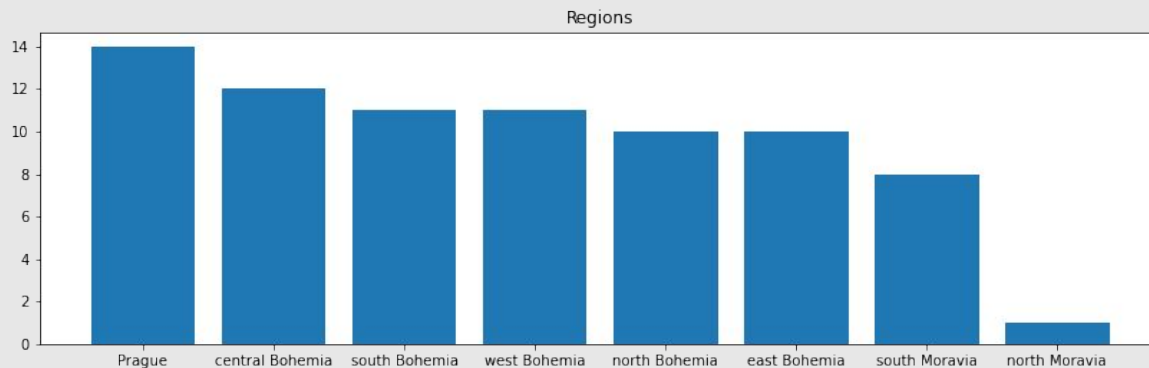
4



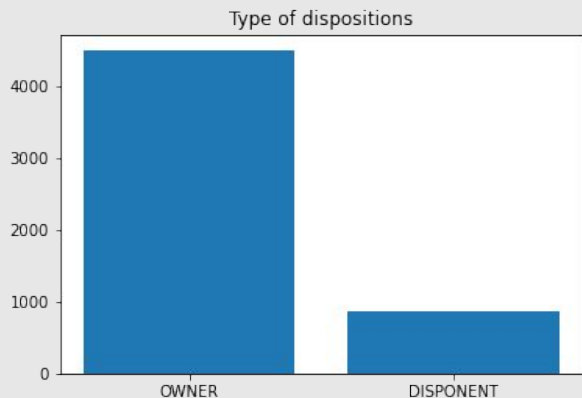
5

These graphs correspond to the rest of the conclusions in the “Exploratory Data Analysis” section

Bar Plots 3/3



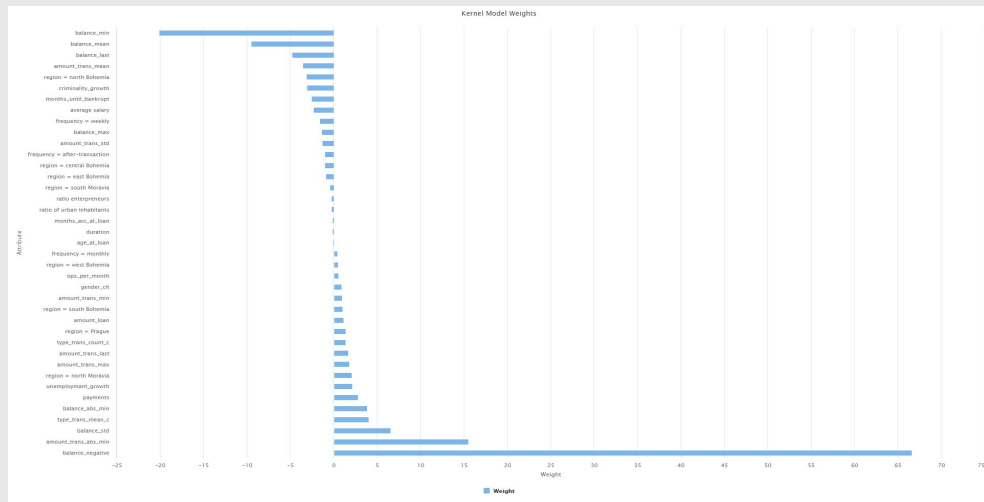
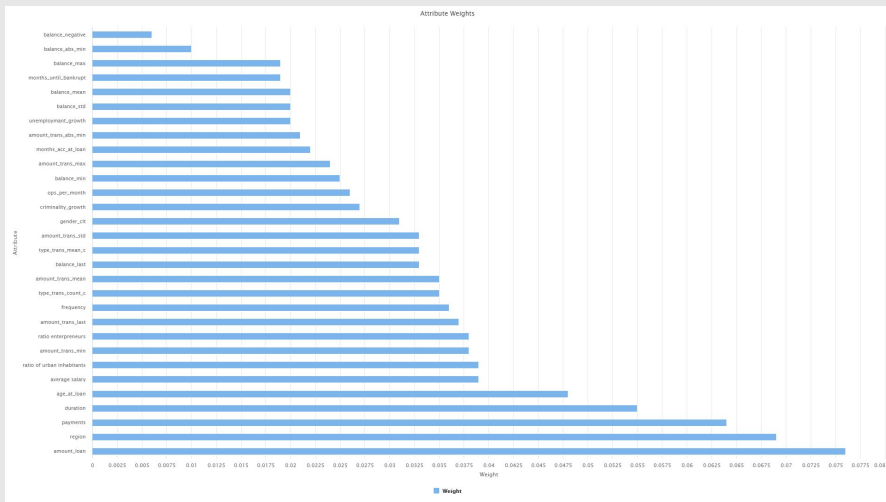
This bar plot shows the number of districts per region. We can then conclude that the region with more districts is Prague.



This graph shows the number of each type of dispositions in the dataset. We can conclude that most clients are owners.

Feature importance

Evaluating feature importance by their weights in some models



Random Forest, most important features:

- amount_loan
- Region
- Payments

SVM, most important features:

- balance_negative
- amount_trans_abs_min
- balance_std

Feature Selection

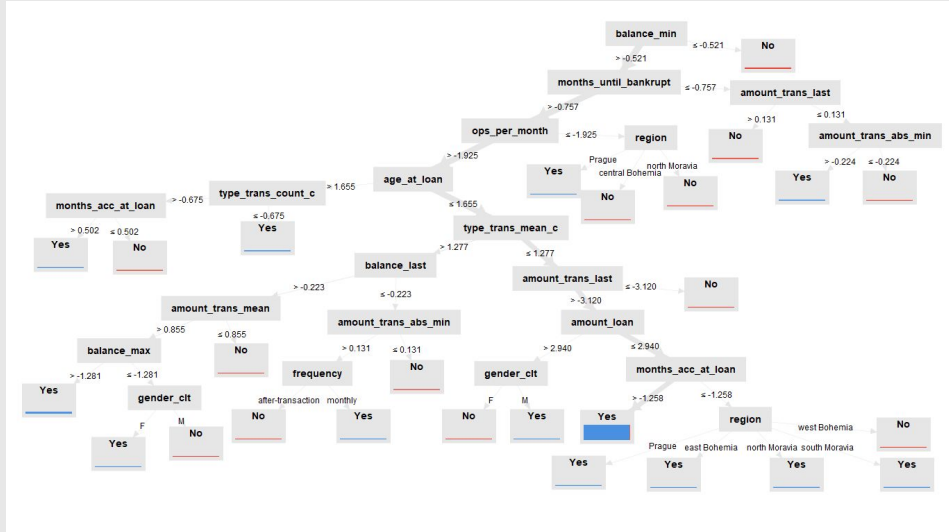
MRMR (Minimum Redundancy Maximum Relevance)

- Its basic idea is that features, that are most relevant to the label, and have the least redundancy (correlation) to previously selected features, are iteratively added.
- Selected Features: amount_loan, payments, criminality_growth, unemployment_growth, age_at_loan, type_trans_mean_c, amount_trans_abs_min, balance_mean, balance_min, balance_std, balance_last, balance_abs_min, balance_negative, months_until_bankrupt, ops_per_month

RFE (Recursive Feature Elimination)

- RFE works by searching for a subset of features by starting with all features in the training dataset and successfully removing features until the desired number remains. Technically, RFE is a wrapper-style feature selection algorithm that also uses filter-based feature selection internally
- Selected Features: amount_loan, duration, payments, average salary, type_trans_count_c, type_trans_mean_c, amount_trans_mean, amount_trans_std, amount_trans_last, balance_mean, balance_min, balance_max, balance_last, balance_negative, ops_per_month,

White-box models

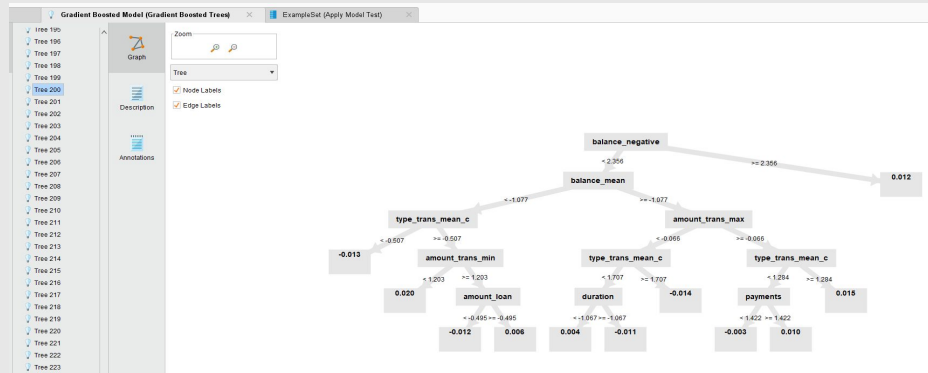


Decision Tree

In RapidMiner we have the option get a graphical view of the tree. In the image is a deep tree that provided the best DT local score. It is harder to analyze since the data is normalized, but we can see that the loan should be given when the region is Prague, and not for the other regions, making a clear separation between regions, which was already concluded in the Descriptive analysis. The main deciding feature is the client's minimum balance after a transaction, the higher the better chances of the loan being accepted. Looking at the leftmost branches we can see that the higher the months until the client goes bankrupt, number of operations per month, client's age at loan and account's month at the loan, the loan should also be accepted.

GBT and RF

In RapidMiner, all of the trees are created and saved as their graphical view



Hyperparameter Tuning Results

Decision Trees

Execution time: 1:46

-----accuracy: 80.16% +/- 4.79%
*****AUC: 0.768 +/- 0.125
-----precision: 37.77% +/- 11.23%
-----recall: 58.89% +/- 20.64%
-----f_measure: 45.13% +/- 11.87%

ConfusionMatrix:

True:	Yes	No
Yes:	236	19
No:	46	27

Parameters:

Decision Tree.criterion = information_gain
Decision Tree.maximal_depth = 10
Decision Tree.apply_prepruning = true
Decision Tree.confidence = 0.3503

SVM

Execution time: 27:24

-----accuracy: 81.39% +/- 3.52%
*****AUC: 0.810 +/- 0.016
-----precision: 41.60% +/- 6.33%
-----recall: 59.78% +/- 28.38%
-----f_measure: 43.90% +/- 15.40%

ConfusionMatrix:

True:	Yes	No
Yes:	240	19
No:	42	27

Parameters:

SVM.kernel_type = radial
SVM.C = 251.1886431509581
SVM.kernel_gamma = 0.001
SVM.kernel_degree =
0.0039810717055349725

Deep Learning

Execution time: 17:42

-----accuracy: 63.90% +/- 28.89%
*****AUC: 0.786 +/- 0.096
-----precision: 30.42% +/- 16.34%
-----recall: 75.56% +/- 32.77%
-----f_measure: 40.43% +/- 17.75%

ConfusionMatrix:

True:	Yes	No
Yes:	175	11
No:	107	35

Parameters:

Deep Learning.activation = Tanh
Deep Learning.rho = 0.6400000000000001
Deep Learning.epsilon = 0.4009600000000001

DT model is easy to understand and great for visual analysis, proved useful in the creation of the pipeline, but it tends to overfit.

We were expecting a better score from SVM since it is closely related to logistic regression, and can be used to predict the probabilities as well based on the distance to the hyperplane.

Deep Learning was implemented just to evaluate the results, it was already expected to not have a good performance due to the small dataset.

Hyperparameter Tuning Results

RF

Execution time: 15:03

-----accuracy: 87.80% +/- 3.04%
*****AUC: 0.843 +/- 0.070
-----precision: 66.00% +/- 24.66%
-----recall: 41.33% +/- 9.24%
-----f_measure: 48.90% +/- 8.77%

ConfusionMatrix:

True:	Yes	No
Yes:	269	27
No:	13	19

Parameters:

Random Forest.criterion = information_gain
Random Forest.number_of_trees = 178
Random Forest.maximal_depth = 14
Random Forest.apply_prepruning = true
Random Forest.apply_pruning = true
Random Forest.minimal_leaf_size = 1
Random Forest.minimal_size_for_split = 2

GBT

Execution time: 14:11

-----accuracy: 86.58% +/- 1.74%
*****AUC: 0.865 +/- 0.042
-----precision: 51.56% +/- 5.70%
-----recall: 67.11% +/- 17.94%
-----f_measure: 57.65% +/- 8.81%

ConfusionMatrix:

True:	Yes	No
Yes:	253	15
No:	29	31

Parameters:

Gradient Boosted Trees.number_of_trees = 57
Gradient Boosted Trees.learning_rate = 0.0423
Gradient Boosted Trees.number_of_bins = 41

XGBoost

Execution time: 11:09

-----accuracy: 86.90% +/- 3.12%
*****AUC: 0.864 +/- 0.075
-----precision: 53.80% +/- 8.96%
-----recall: 54.89% +/- 18.65%
-----f_measure: 53.11% +/- 12.24%

ConfusionMatrix:

True:	Yes	No
Yes:	260	21
No:	22	25

Parameters:

XGBoost.rounds = 30
XGBoost.booster = DART
XGBoost.top_k = 15
XGBoost.feature_selector = cyclic
XGBoost.updater = coord_descent
XGBoost.learning_rate = 1.0
XGBoost.normalize_type = tree
XGBoost.sample_type = weighted

Ensemble methods achieved the best scores and are not susceptible to overfit.

GBT is a Learning Algorithm, so the smaller the dataset the less efficient it will be, but still it had a similar performance to the other ensemble methods.

XGBoost already applies feature selection with several selectors to tune.

PM Tools

To manage the project content, collaborate and communicate we used:

- GitHub, Discord, Notion

To develop the project we used:

- Conda, Jupyter Notebook, Rapid Miner

Inside Python we used the following packages:

- Pandas, Numpy, Seaborn, scikit-learn, matplotlib, imblearn, xgboost

For the methodology, we separated the stages of the projects into different notebooks (data analysis, data processing, prediction) and the data was shared between notebooks via a pickle file.

We made a plan at the beginning of the project: complete the pipeline, then make improvements. We followed this plan and it worked well, for the exception of the descriptive problem.



Conclusion

With the project finished, we can now look back and draw some conclusions from the work done and methods used. In the first weeks we focused on building the pipeline for the project composed of Jupyter Notebooks, which worked well for the intended purpose. We did not follow along the moodle criteria from the start, because the information was too much and too variate, and so, after having the pipeline done, we focused on feature engineering and data cleaning. The optimal route here would have been to analyse the data together and draw some conclusions from data understanding and the descriptive problem, so we would have a better overall idea of which features were important. Even taking into account the previous statement, we followed the CRISP-DM by always returning back to previous stages and improving them based on recent information. For the modeling stage, we split our pipeline into two: our clean data was sent to another notebook and to various RapidMiner Processes, this gave us great insights into various models and algorithms, which, together with other students inputs on the matter, gave us the ability to pinpoint the models that were relevant to our case study.

By the end of the project, we did not get the best possible score, but still, it was overall a positive result. If we were to remake this project, we would be more confident on our methodologies and would most likely be able to come through with better results.

Useful References

[How to predict Loan Eligibility using Machine Learning Models](#)

[SMOTE for Imbalanced Classification with Python](#)

[The right way of using SMOTE with Cross-validation](#)

[Why NOT to select features before splitting your data](#)

[Feature Selection Part 2: Using the Feature Selection Extension](#)

[Visualizing distributions of data](#)

[Correlogram in Python Graph Gallery](#)

[A Comprehensive Guide to Data Analysis using Pandas](#)