# Data Mining Case Study

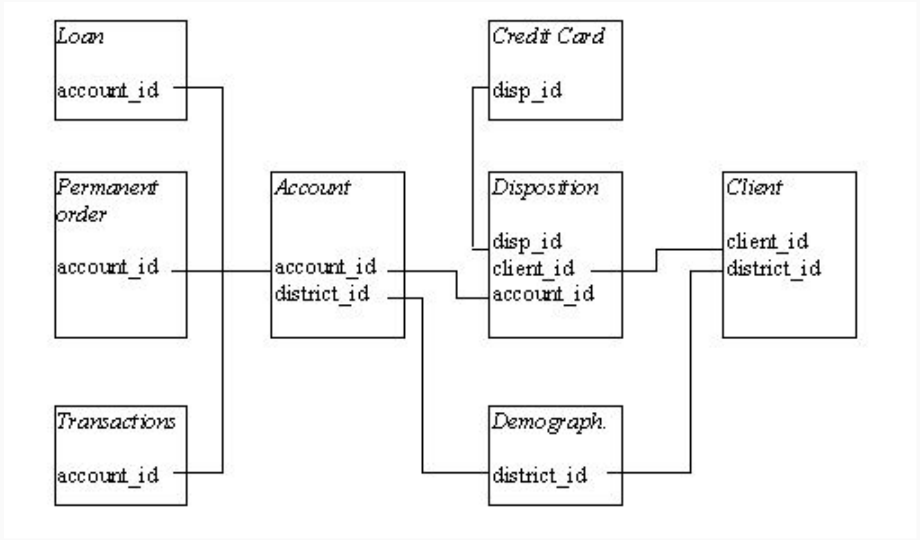
Descriptive and Predictive tasks for Bank records

Group 11
Luís Carvalho
Miguel Ramalho

Docentes: Carlos Soares, João Moreira
FEUP - ECAC
2018/2019

## Domain Description

The case study focuses on analysis of records from a Czech bank from 1996 to 1993 to 1998. The records include accounts, credit cards, clients, geographical information, transfers and loan history.

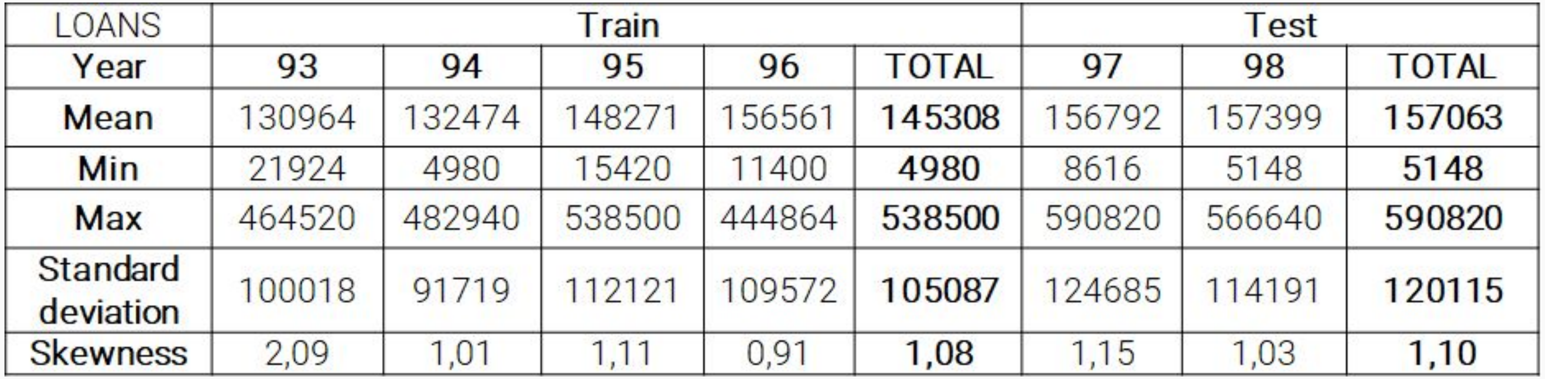


- **4500 accounts**
- **5369 clients**
- > **426.000 transfers** (include payment of *dividends* and *spread*, *withdrawals* and real *inter-account transfers*)
- **682 loans** (both labelled and unlabelled).

## Exploratory Data Analysis

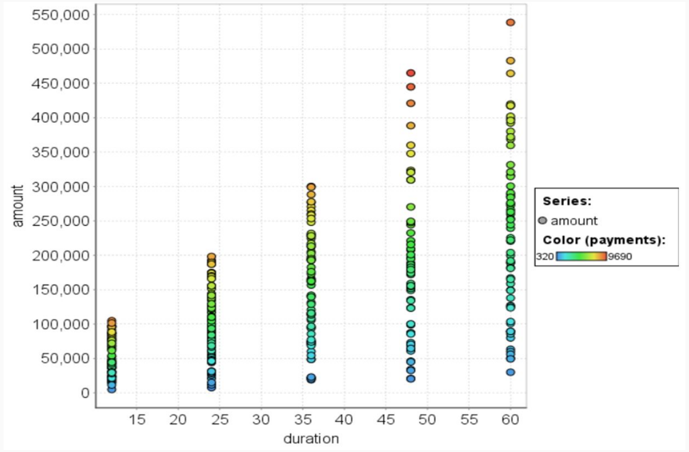
**Main Findings**

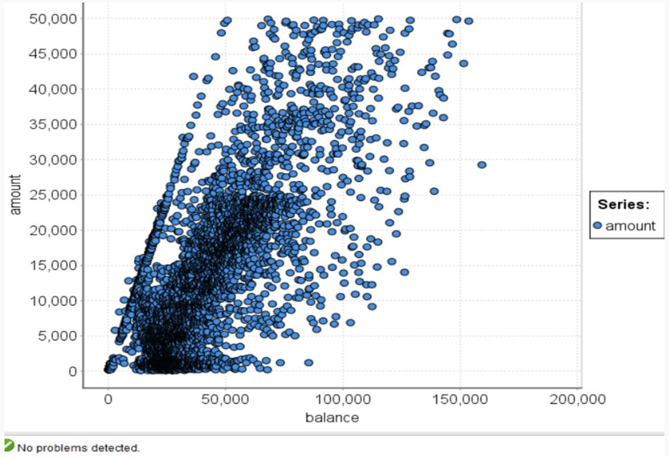
- Correlation between loan duration and amount (should be high);
- No relation between loan duration and final payment (1 vs -1);
- No correlation between gender and final payment;
- From 73 districts, only 6 have more than 100 clients. One of those districts has 663 clients;
- Transactions over 50.000$ are not allowed;
- The credit card type more used is the Classic one;
- The balance of the accounts (50%) is mainly constant (XYZ analysis);
- The age distribution shows a pattern appropriated to the business.
- The mean, STD and skewness from both train and test loan sets seem comparable.larger differences from mean of 93/94 to that of 97/98, than that of 95/96 to 97/98!!

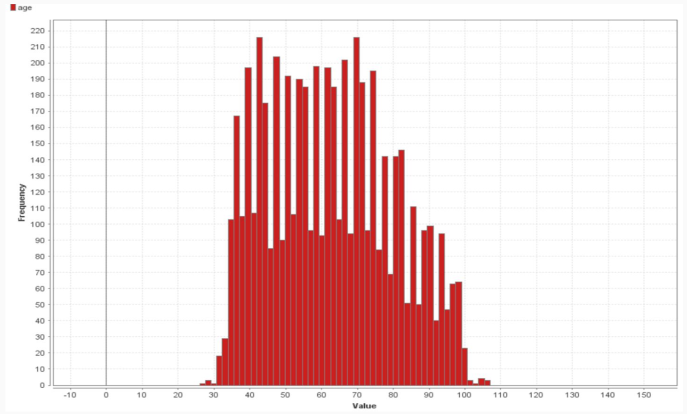


| LOANS | Train | | | | | Test | | |
|---|---|---|---|---|---|---|---|---|
| **Year** | **93** | **94** | **95** | **96** | **TOTAL** | **97** | **98** | **TOTAL** |
| **Mean** | 130964 | 132474 | 148271 | 156561 | **145308** | 156792 | 157399 | **157063** |
| **Min** | 21924 | 4980 | 15420 | 11400 | **4980** | 8616 | 5148 | **5148** |
| **Max** | 464520 | 482940 | 538500 | 444864 | **538500** | 590820 | 566640 | **590820** |
| **Standard deviation** | 100018 | 91719 | 112121 | 109572 | **105087** | 124685 | 114191 | **120115** |
| **Skewness** | 2,09 | 1,01 | 1,11 | 0,91 | **1,08** | 1,15 | 1,03 | **1,10** |

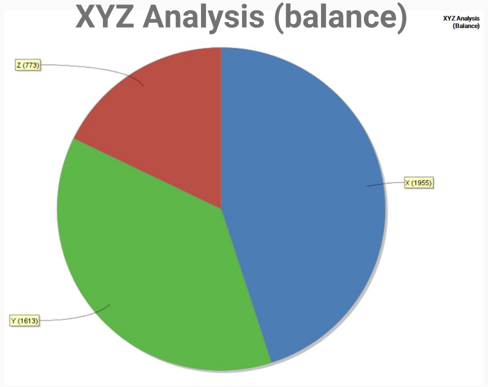
# Exploratory Data Analysis



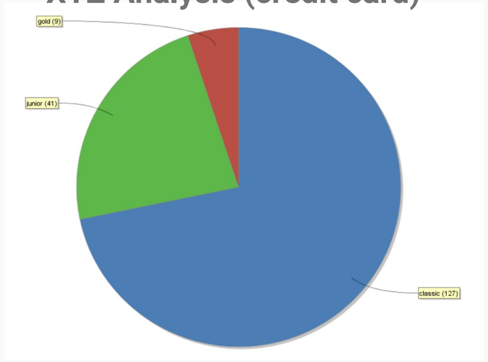




XYZ Analysis (balance)



XYZ Analysis (credit card)



## Data Preparation

**Cleaning/Formatting**

- Extract birthdate and gender from "*birth_number*"
- Conversion of dates into "*yyyy-MM-dd*" for easy import into rapidminer
- Replace districts unemployment and crime ratio for 1995 with average values (Jesenik)
- Renaming of attributes to regular names (no special chars: º, ', ...) in districts
- Nominal to numeric attributes (gender, ...), client birthdate to age, ... (*Generate Attributes*)
- Missing table *permanent order*, ignored
- Wrong info on labels for loan tables (nothing to do)
- Each client has only one account.
- 3631 accounts have 1 client, 1738 have 2 clients (5369)
- Transactions last date is previous to Loan
- No owner has more than one loan requested

**Tools used**

- Excel
- Notepad++ find and replace
- Access
- RapidMiner Studio
- R (some tests, but python...)
- Jupyter Notebooks (Python w/ libs)

**Data tasks**

- Import into access database (export as SQL + SQL queries)
- Import into RapidMiner

## Feature Engineering

Given the set of original data entries, there has been some concern with generating some new features. This proved useful for descriptive and invaluable for prediction.

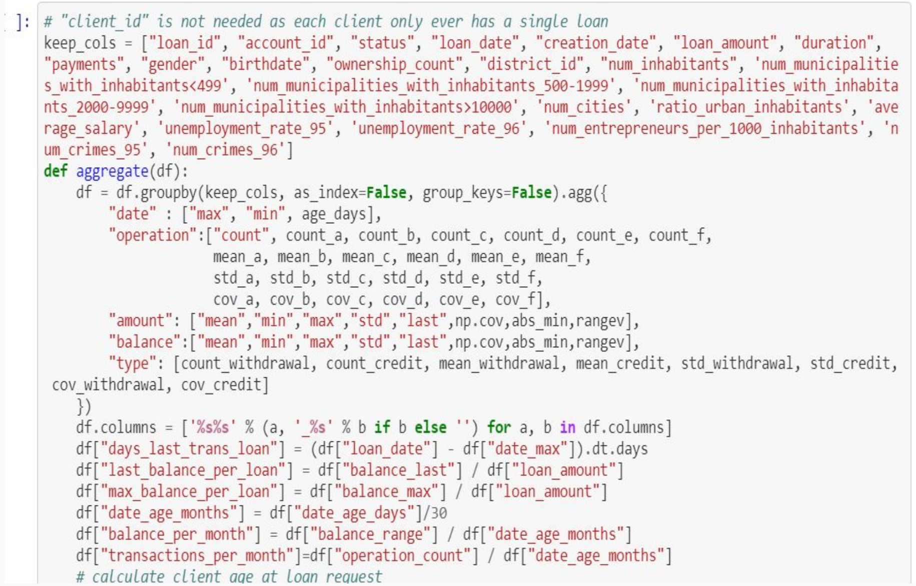
How?

- Rapid Miner
- Excel
- Jupyter

What?

- Calculation of client age from birthdate
- How many loans some client has
- How many clients are in the loan request account
- How many transactions a client has made
- Integrate geographical information for each client
- Complete list in the **Jupyter Notebook.**

In total, more than **80 features** were generated.



```
]: # "client_id" is not needed as each client only ever has a single loan
keep_cols = ["loan_id", "account_id", "status", "loan_date", "creation_date", "loan_amount", "duration",
"payments", "gender", "birthdate", "ownership_count", "district_id", "num_inhabitants", 'num_municipalitie
s_with_inhabitants<499', 'num_municipalities_with_inhabitants_500-1999', 'num_municipalities_with_inhabita
nts_2000-9999', 'num_municipalities_with_inhabitants>10000', 'num_cities', 'ratio_urban_inhabitants', 'ave
rage_salary', 'unemployment_rate_95', 'unemployment_rate_96', 'num_entrepreneurs_per_1000_inhabitants', 'n
um_crimes_95', 'num_crimes_96']
def aggregate(df):
    df = df.groupby(keep_cols, as_index=False, group_keys=False).agg({
        "date" : ["max", "min", age_days],
        "operation":["count", count_a, count_b, count_c, count_d, count_e, count_f,
                     mean_a, mean_b, mean_c, mean_d, mean_e, mean_f,
                     std_a, std_b, std_c, std_d, std_e, std_f,
                     cov_a, cov_b, cov_c, cov_d, cov_e, cov_f],
        "amount": ["mean","min","max","std","last",np.cov,abs_min,rangev],
        "balance":["mean","min","max","std","last",np.cov,abs_min,rangev],
        "type": [count_withdrawal, count_credit, mean_withdrawal, mean_credit, std_withdrawal, std_credit,
 cov_withdrawal, cov_credit]
    })
    df.columns = ['%s%s' % (a, '_%s' % b if b else '') for a, b in df.columns]
    df["days_last_trans_loan"] = (df["loan_date"] - df["date_max"]).dt.days
    df["last_balance_per_loan"] = df["balance_last"] / df["loan_amount"]
    df["max_balance_per_loan"] = df["balance_max"] / df["loan_amount"]
    df["date_age_months"] = df["date_age_days"]/30
    df["balance_per_month"] = df["balance_range"] / df["date_age_months"]
    df["transactions_per_month"]=df["operation_count"] / df["date_age_months"]
    # calculate client age at loan request
```

## Descriptive Task (DT)

**Goal:** Separate customers into groups that map relevante behaviours.

Clustering operations performed (each includes several iterations by adding or removing attributes):

1. Simplest client clustering (age and gender);
2. Clients and numerical district information;
3. Clients and transaction history (what types, amount and balance);
4. Clients, transactions and demographic information;
5. Clients and loans history;
6. Clients and balance history.

# DT - Experimental Setup

Clustering the different groups was an iterative process, where the main concerns were:

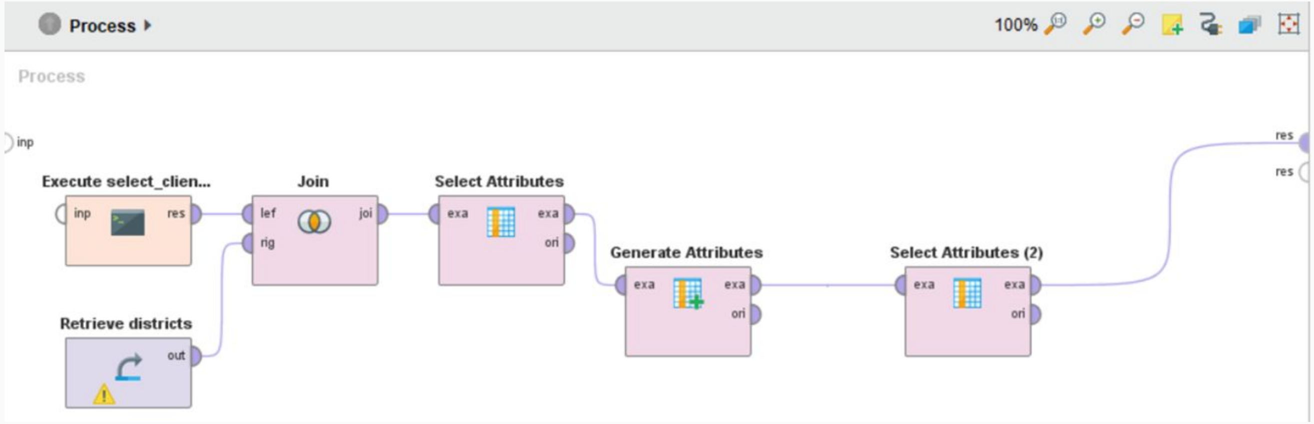
- Cluster size
- Variation of attributes between groups
- Meaning of the group in the business context

The iterative process for the clustering task was composed by:

- Varying the number of groups (k)
- Adding/removing attributes
- Creating new attributes

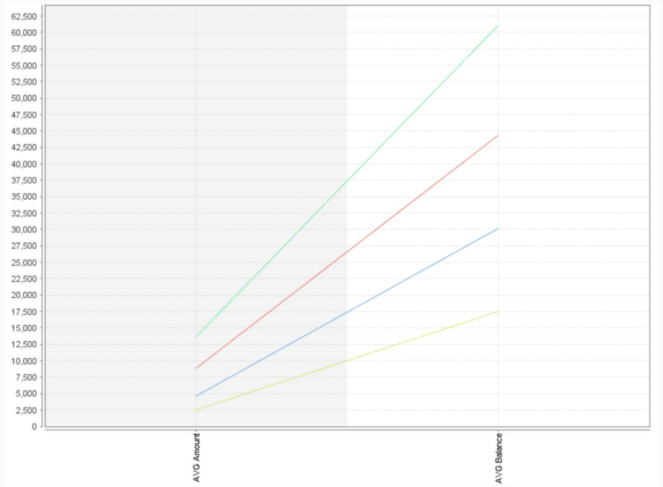
Algorithms used:

- K-means
- K-medoids
- DBScan



# DT - Results

## Clients and Transactions (AVG amount and AVG balance)



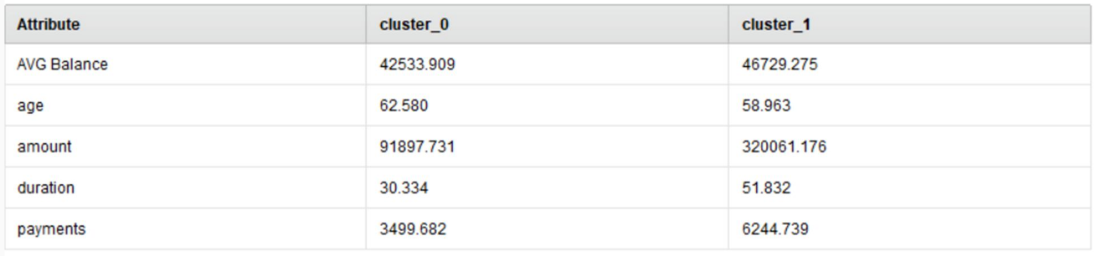
The purpose of this clustering was to segment the groups according to their economic power.

The results of this clustering show a rational behaviour with the average transaction amount is expected to be higher for the richer groups.

## Clients and Loans

The correlation found in the exploratory analysis is confirmed by the two biggest groups . The higher the amount of the loan the higher the duration.

The cluster with the highest AVG balance has higher payments.



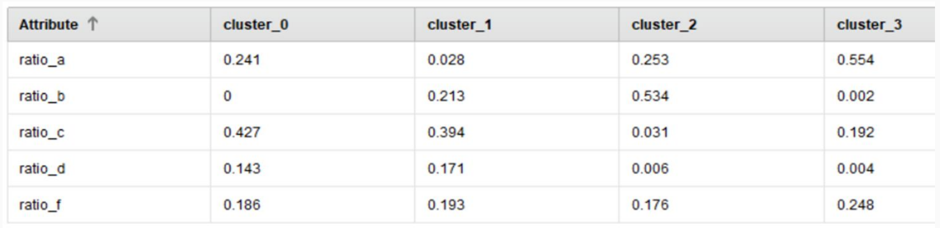
| Attribute | cluster_0 | cluster_1 |
|---|---|---|
| AVG Balance | 42533.909 | 46729.275 |
| age | 62.580 | 58.963 |
| amount | 91897.731 | 320061.176 |
| duration | 30.334 | 51.832 |
| payments | 3499.682 | 6244.739 |

## DT - Results (continued)

### Clients and Transactions (ratio of transactions (e excluded))

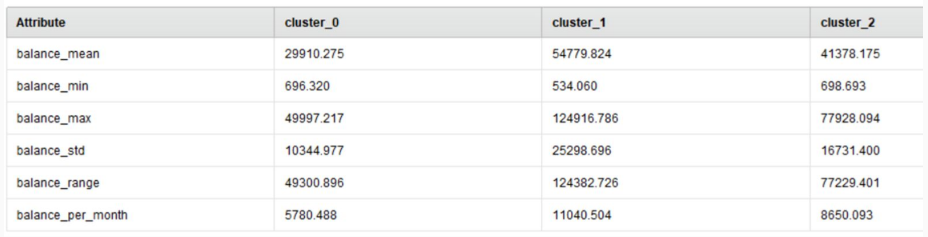
The ratios presented in this cluster are related with the type of the transactions.

The clusters 0 and 1 are the biggest ones (>1000 clients). If we use a smaller k they would be together because they behave similarly for the transactions of type C,D,F. However they are really different on the distribution of transactions of type A,B. The smaller groups almost does not have transaction of C type (the most important for the clusters 0,1)



| Attribute ↑ | cluster_0 | cluster_1 | cluster_2 | cluster_3 |
|---|---|---|---|---|
| ratio_a | 0.241 | 0.028 | 0.253 | 0.554 |
| ratio_b | 0 | 0.213 | 0.534 | 0.002 |
| ratio_c | 0.427 | 0.394 | 0.031 | 0.192 |
| ratio_d | 0.143 | 0.171 | 0.006 | 0.004 |
| ratio_f | 0.186 | 0.193 | 0.176 | 0.248 |

### Balance Analysis



| Attribute | cluster_0 | cluster_1 | cluster_2 |
|---|---|---|---|
| balance_mean | 29910.275 | 54779.824 | 41378.175 |
| balance_min | 696.320 | 534.060 | 698.693 |
| balance_max | 49997.217 | 124916.786 | 77928.094 |
| balance_std | 10344.977 | 25298.696 | 16731.400 |
| balance_range | 49300.896 | 124382.726 | 77229.401 |
| balance_per_month | 5780.488 | 11040.504 | 8650.093 |

There is possible 3 groups with similar sizes regarding the balance.
Each group has a different average balance, with a difference about $10K. It is also possible to understand that if the average balance goes higher, the range has also becomes wider.

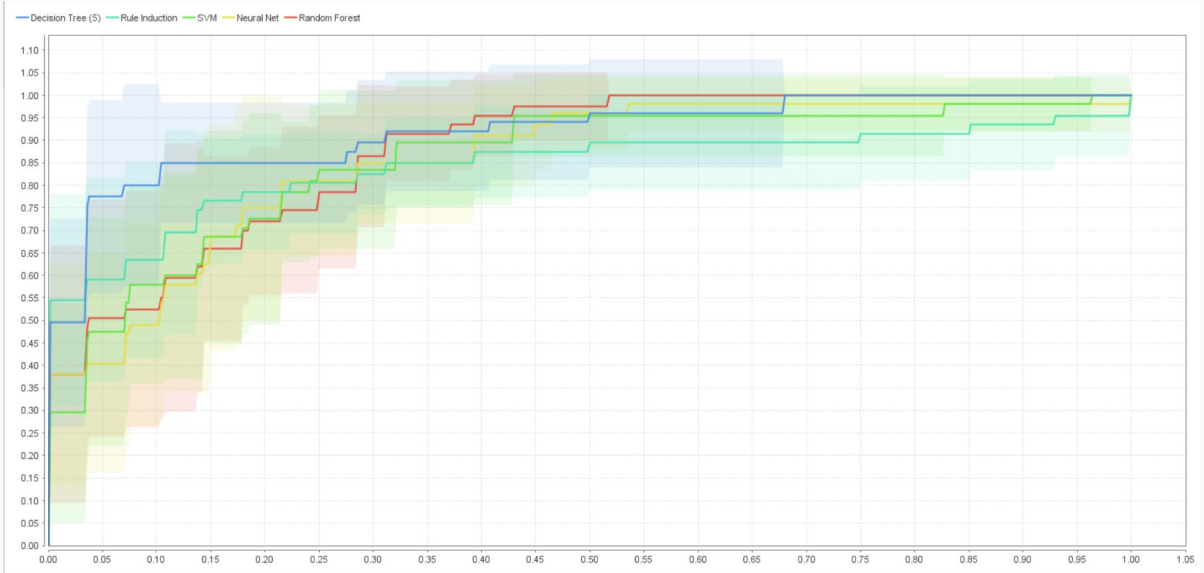
## Predictive Task (PT)

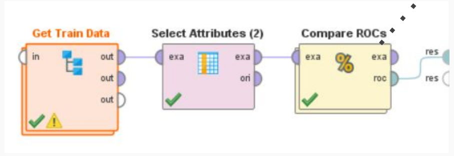
Given a set of past loans (paid or unpaid) and client information (geographical, behavioural, ...) train a ML model that can predict whether a new loan will be paid by the client, so as to answer **the question**:

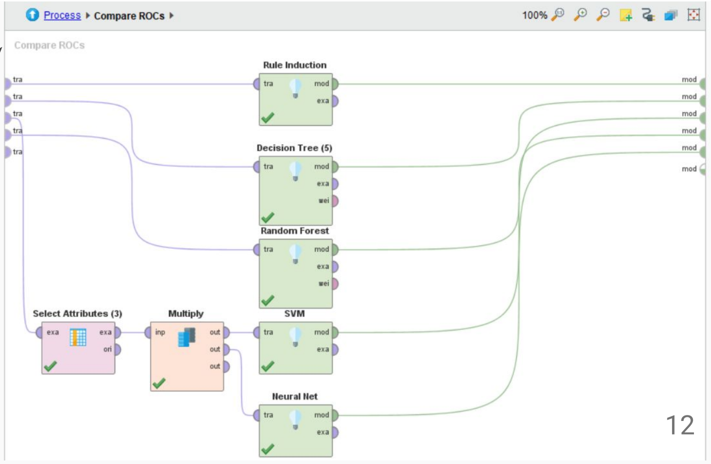
**To loan or not to loan?**

## PT - Algorithm Comparison

- Which Algorithms would be more promising?
- Neural Networks seem to have poor (relative) performance
- These and more algorithms were tested, as this was not performed at the initial stage







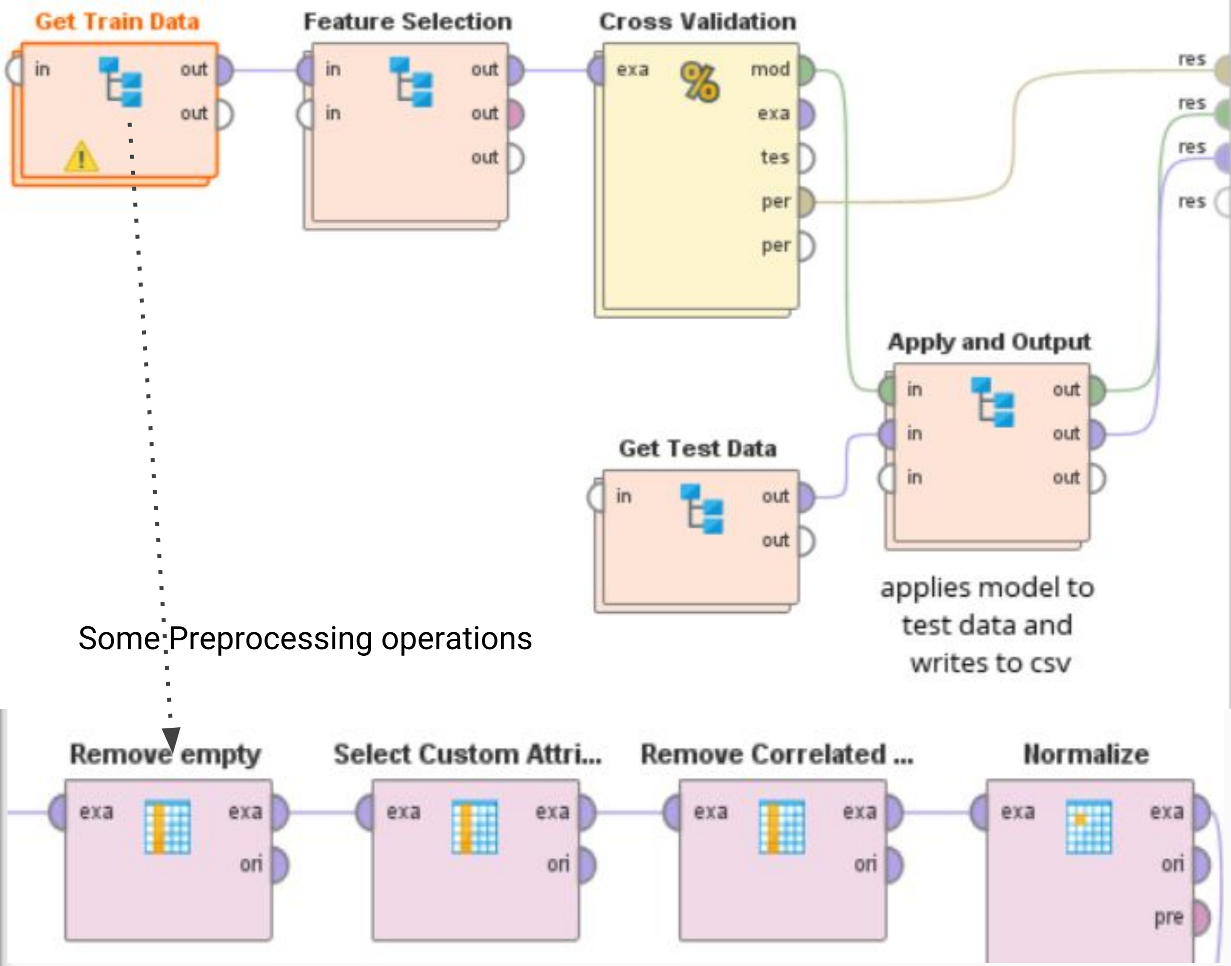
# Experimental Setup

**Setup**

- (Initially) Deterministic 80/20 Split
- (From then on) Cross Validation
- Feature selection (where needed)
- Binary Performance (Acc, F1, ...)
- *"prediction_%{process_start}.csv"*

**Algorithms**

- Decision tree
- Random Tree/Forest
- SVMs (negative examples replicated)
- Naive Bayes
- Neural Networks (but very few data points for many features)
- Vote Module for Rapidminer
- ...

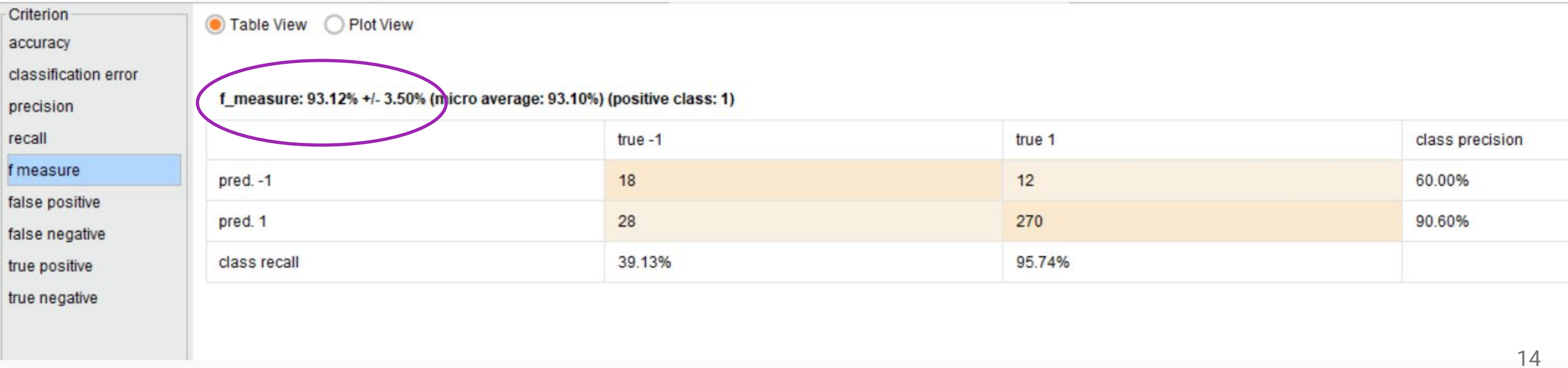


applies model to test data and writes to csv

Some Preprocessing operations

# PT - Results

- New features led to a consistent F1-score of about **0.95** for **multiple algorithms**!
- Thus, Feature Engineering is considered the **most import part** of the process
- Some concerns over overfitting kaggle public leaderboard data and having a bad result on private leaderboard
- Other results: Ada Boost (0.74), Naïve Bayes (0.79), other Ensembles (~0.80), SVM [libSVM](0.82), Deep Learning (0.90), Vote with various algorithms (0.91).
- Kaggle results tend to be consistent with predicted F1-score (when they are not, it means **overfitting**)



**f_measure: 93.12% +/- 3.50% (micro average: 93.10%) (positive class: 1)**

| | true -1 | true 1 | class precision |
|---|---|---|---|
| pred. -1 | 18 | 12 | 60.00% |
| pred. 1 | 28 | 270 | 90.60% |
| class recall | 39.13% | 95.74% | |

## Conclusions and Future Work

- Feature selection is a crucial step
- Some algorithms could not be used effectively due to the small amount of data, eg: Deep Learning
- There is cause for concern over the “age” of data, as human behaviour evolves
- Tree-based models are very interesting due to embedded feature selection abilities, but tend to overfit if not properly configured
- Other algorithms require more manual configuration and *typically* led to worse results

- Looking forward to learning more about some algorithms so they can be used properly (ensembles, SVM kernels, Naive Bayes w/ kernel, ...)
- Now that 20 years have passed, employ
- Further explore the impact of SMOTE Unsampling
-

## Some Useful References

- Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (book)
- Consumer credit-risk models via machine-learning algorithms (paper)
- Business data mining - a machine learning perspective (paper)
- RapidMiner Studio Documentation (tool docs)
- Pandas Documentation (tool docs)
- Kaggle Tricks and Tips (blog post)
- Best Practices for Feature Engineering (blog post)

# Made with ❤ at FEUP

Group 11

Luís Carvalho
Miguel Ramalho

ECAC - 2018/2019

“Coming up with features is difficult, time-consuming and requires expert knowledge. *‘Applied machine learning’* is basically feature engineering.”

Andrew Ng