Machine Learning @FEUP

G41
Adriano Soares (up201904873@up.pt)
Filipe Campos (up201905609@up.pt)
Francisco Cerqueira (up201905337@up.pt)
Vasco Alves (up201808031@up.pt)

Business Understanding

Domain Analysis

- The regions of the dataset (Prague, Bohemia, Moravia) indicate that **we are dealing with a Czech bank**
- The dataset contains information about records **between 1993 and 1998**, right after **the dissolution of Czechoslovakia on December 31, 1992**
- The dissolution had some **negative impact on both economies**, especially in 1993
- The currency used after the dissolution is **czech koruna (Kč)**

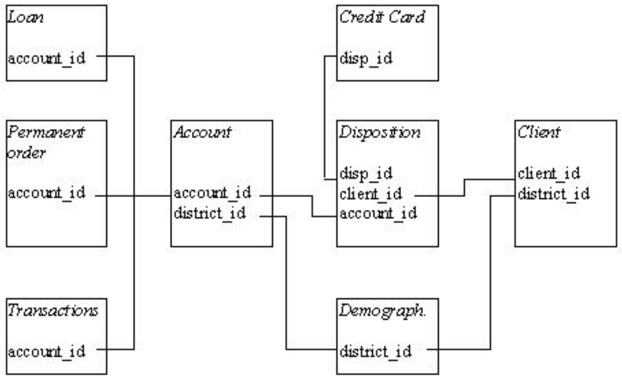


Figure 1 - Domain Model

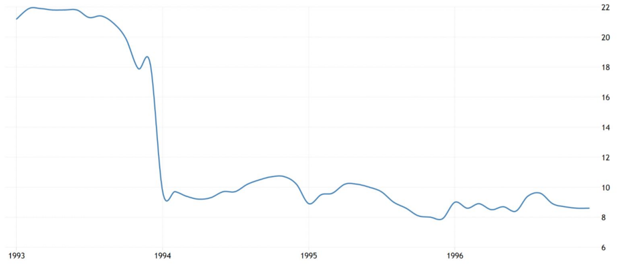


Figure 2 - Czech Republic inflation rate between 1993 and 1999

Business Goals

The bank wants to **predict whether or not a customer will bring profit**

The bank presented us with:

- A predictive problem: **based on information from their clients (transactions, loans granted, credit cards issued...) will a client be able to pay a loan in full before its duration has expired?**
- An open descriptive problem, we choose: **to determine if a client is good or bad based on available data**

Constraints:

- The model utilized for the predictive model should be explainable, due to European regulatory reasons

Assumptions:

- Clients who will be able to pay the loan will bring higher profit to the bank, these customers can be classified as good clients
- Clients who won't be able to pay the loan may bring even higher profit in the long run, we choose to define them as bad clients since there's a higher risk for the bank not to have profit at all

Data Mining Goals

Business Goals:
1. Predict whether a client will pay a loan or not
2. Determine if a client is good or bad based on available data

- The first goal can be defined as a classification problem, where the target variable is the loan status (paid or not paid). The positive class is not paid, because it is the minority class and the one we are most interested in. **To be considered successful, an roc-auc score greater than or equal to 0.85 should be achieved**
- The second goal can be defined as a clustering problem, where we group clients into two distinct clusters, good and bad clients. For this approach we can try to minimize the intra-cluster distance while also maximizing the distance between clusters and evaluate it using metrics such as the silhouette, completeness, homogeneity and v-measure scores. **We consider successful a V-measure score greater than or equal to 0.6**

Data Understanding

Exploratory Data Analysis (1/7)

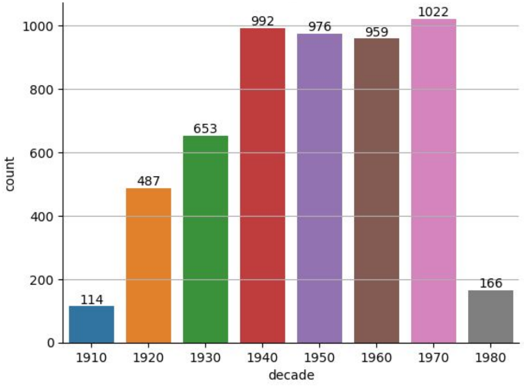


Figure 3 - Number of clients per decade of birth
(the majority of the clients were **born between 1940 and 1970**, ages from 23 (minimum) to 57 (maximum) years old)

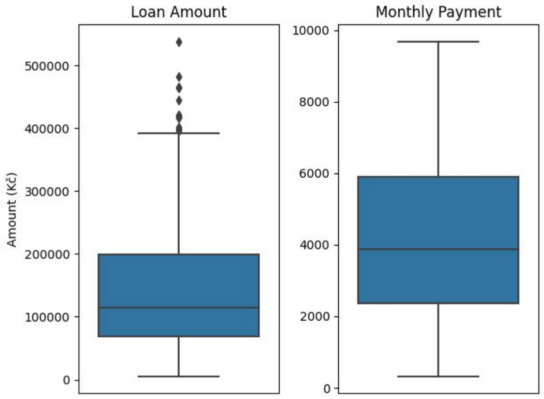


Figure 4 - Loan amount and monthly payment values
(there are some outliers when it comes to loan amount, they are expected as **loan amounts of that magnitude are scarcer**)

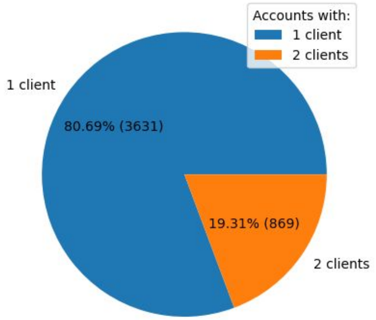


Figure 5 - Amount of clients per account
(accounts only have **one or no disponent**)

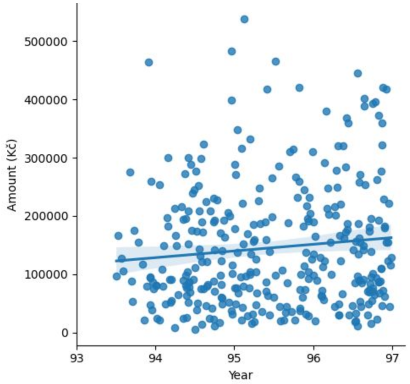


Figure 6 - Loan amount over time
(the **average loan amount has increased over the years**, probably due to the **lower loan amounts by 1993 due to the dissolution**)

Exploratory Data Analysis (2/7)

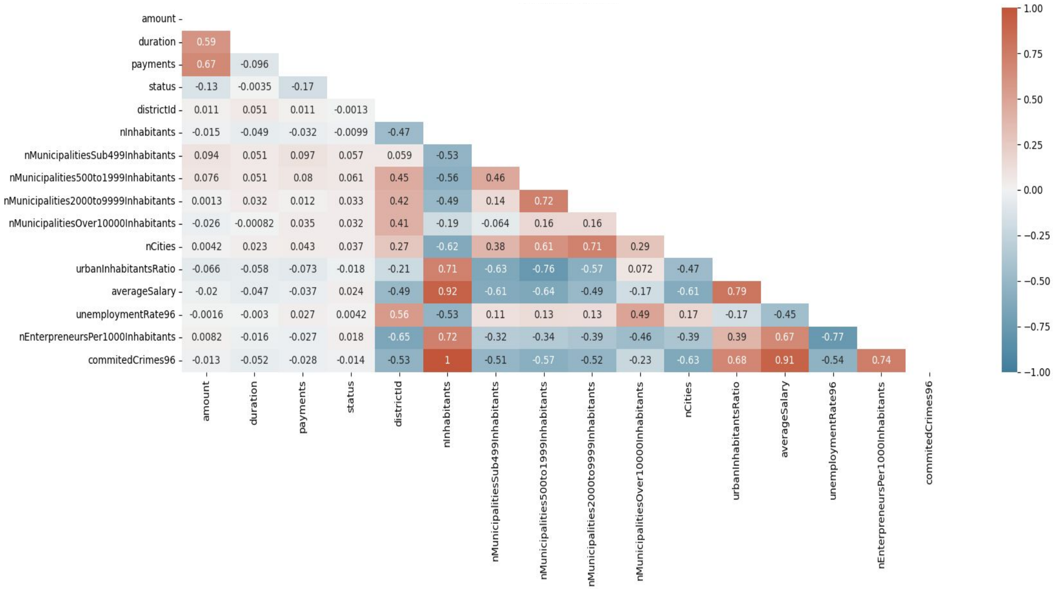


Figure 7 - District table correlation matrix
*(there's **no significant correlation between the status attribute and any of the attributes in the district table**)*

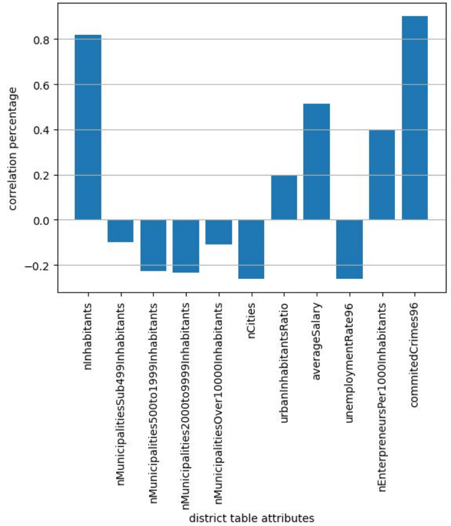
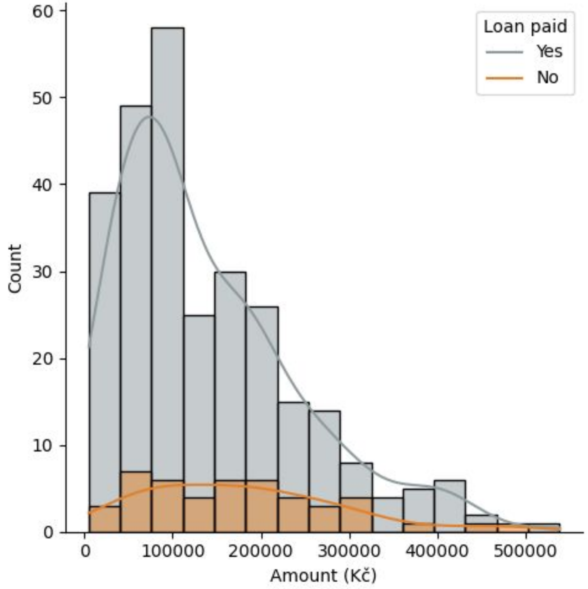
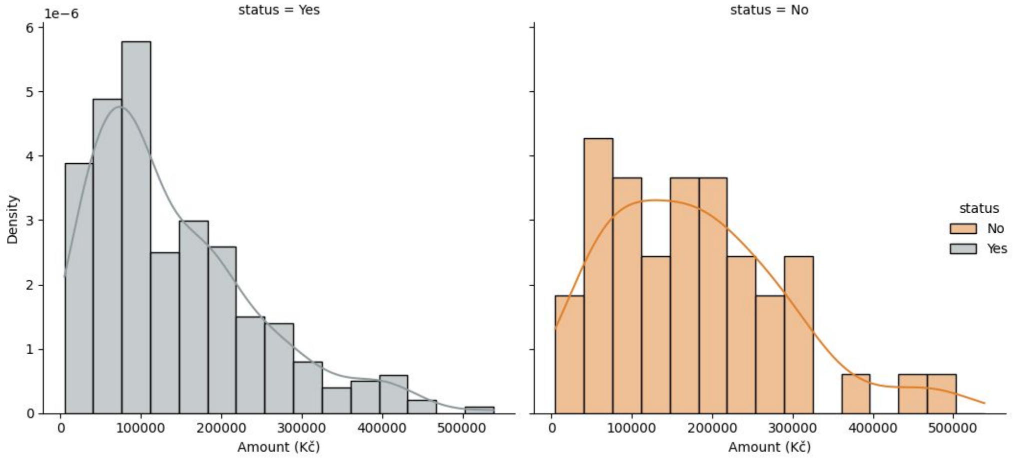


Figure 8 - Correlation between the number of accounts per district and the district table attributes
*(the **number of accounts per district is highly correlated with the number of inhabitants** and **somewhat related to the average salary**; given that the number of accounts per district is factoring all the accounts from 1993 to 1997, **we can consider the correlation with the committed crimes from 1996 as an outlier and irrelevant**)*

Exploratory Data Analysis (3/7)





Figures 9 and 10 - Loan amount distribution
(the number of paid loans and the ratio between paid/unpaid loans is higher the lower the loan amount is)

Exploratory Data Analysis (4/7)

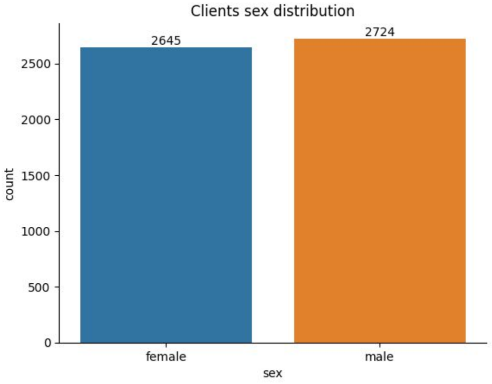


Figure 11 - Clients sex distribution
*(the data is **very balanced** when it comes to this attribute)*

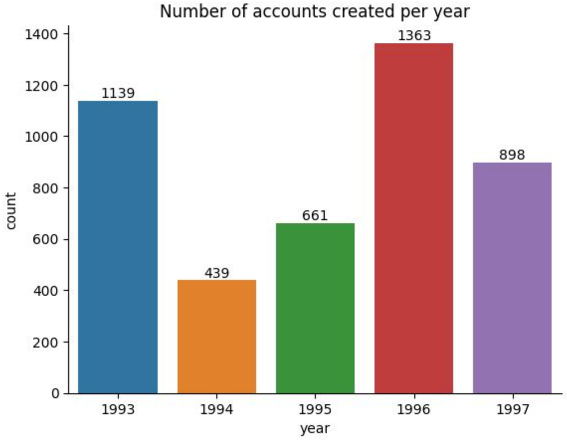


Figure 12 - Number of accounts created per year
*(there's **steep decrease** in the number of accounts created between **1993 and 1994 probably due to the dissolution**)*

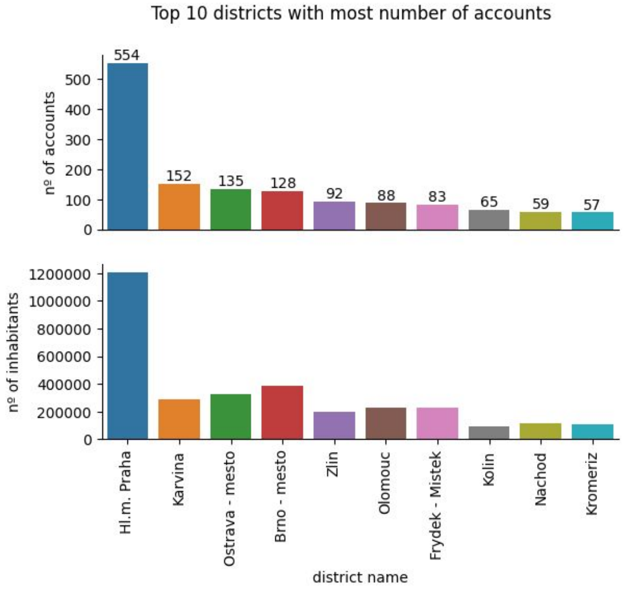


Figure 13 - Top 10 districts with most number of accounts
*(there's a **relation** between the **number of accounts per district** and the n**umber of inhabitants** of that district, as expected; Praha has the most number of inhabitants and accounts due to it being a metropolitan area)*

Exploratory Data Analysis (5/7)

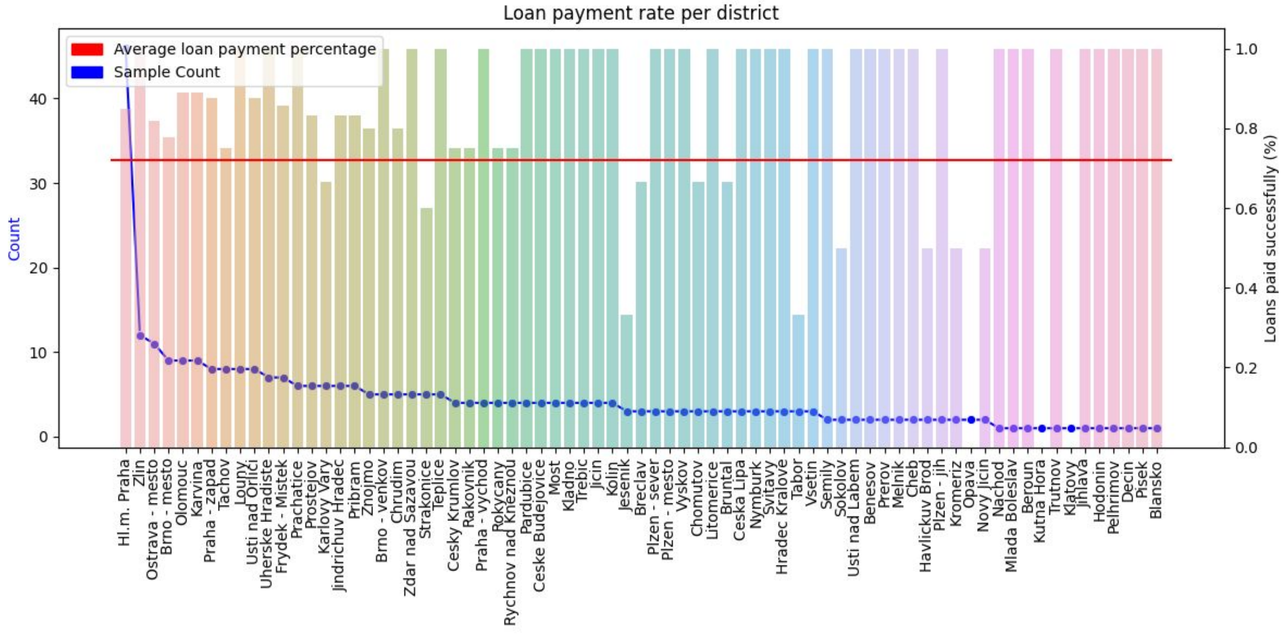


Figure 14 - Loans paid successfully per district

Exploratory Data Analysis (6/7)

	amount	balance
count	396685.000000	396685.000000
mean	299.980242	35804.792507
std	10798.494973	19692.148243
min	-86400.000000	-13588.700000
25%	-2700.000000	22424.300000
50%	-14.600000	30959.600000
75%	210.600000	44661.000000
max	74812.000000	193909.900000

*Table 1 - Different metrics regarding the transactions table, with negative amounts for withdrawals (there are **more withdrawals than credit transactions, but the average amount of credit transactions are higher**, as shown by the positive average amount; **less than 25% of all transactions were done with negative balance**)*

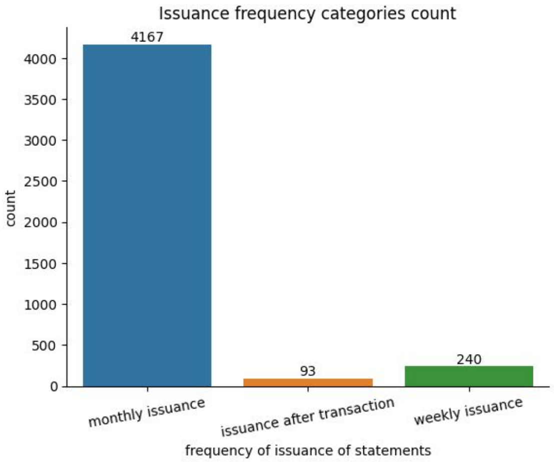


Figure 15 - Issuance frequency categories count

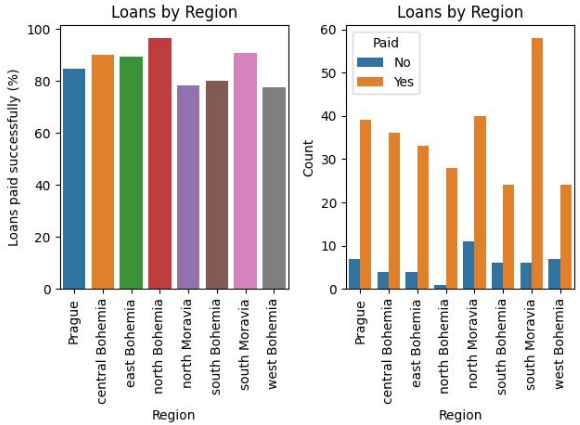


Figure 16 - Loans by region

Exploratory Data Analysis (7/7)

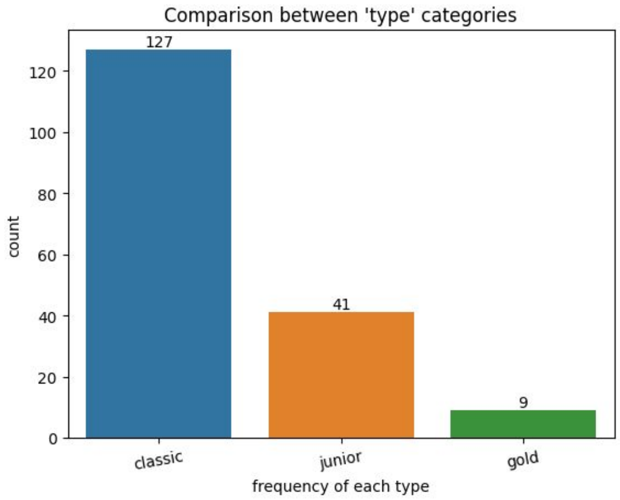


Figure 17 - Card type distribution

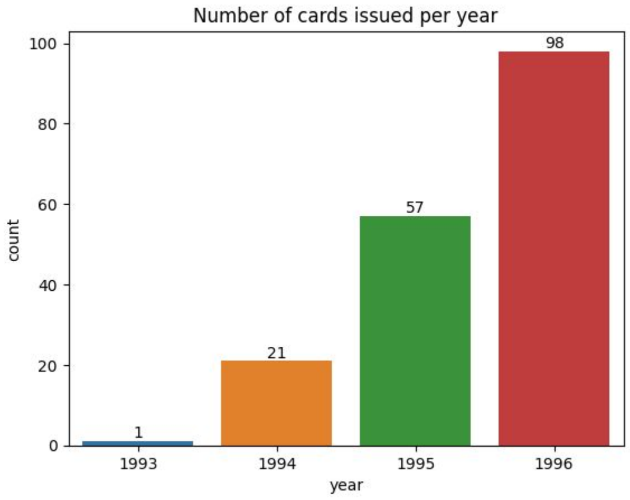


Figure 18 - Number of cards issued per year

Data Preparation

Assessments of Dimensions of Data Quality

Completeness:

- The dataset is incomplete since it has missing values, such as *crimesCommited95* and *unemploymentRate95* for Jesenik district

Accuracy:

- The dataset is accurate as we did not find any erroneous outlier

Consistency:

- The dataset is consistent, since the information matches across all tables through the use of foreign keys

Validity:

- The dataset is invalid as the loan status is not defined as ‘A’, ‘B’, ‘C’ or ‘D’ as stated in the case description

Uniqueness:

- The dataset is unique, since there are no duplicate records

Timeliness:

- The information is available when it is needed, since when we want to predict the status of a loan we have access to the information we need for it

Data Preparation (1/2)

Data Integration:

- Convert client's *birth_number* attribute to *birthday* and *sex*

Data Transformation:

- Convert dates to unix timestamps (e.g. *loanDate*)
- Apply one hot encoding to categorical features (e.g. *frequency*)
- Apply label encoding to categorical features (e.g. *districtName*)
- Apply binary encoding to binary features (e.g. *ownerSex*)
- Apply scaling ($x' = \frac{x - \bar{x}}{\sigma}$)

Data Cleaning:

- **Redundancy -** Redundant attributes were removed during the Feature Selection process
- **Missing Data -** We filled missing data from the *crimesCommited95* and *unemploymentRate95* by using the following year's data as an approximation and subtracting the average growth/decrease rate across these years
- **Outliers -** We analyzed all tables for outliers and concluded that they are not erroneous and could be kept

Train-test Split:

- We use a **80/20 train/test split** because it achieved a paid/unpaid loan ratio similar to the one present in the whole dataset without compromising the amount of available train instances (since we have a small dataset)

Data Preparation (2/2)

Sampling:

- We didn't find any use to sample for domain-specific or development purposes since the low number of instances didn't result in significant model training times
- We would start with small sample and grow to a significant one if that showed an inconvenience

Imbalanced Data:

- We applied **SMOTE** to oversample the minority class in our training sample

Feature Selection:

- We ran DVC experiments with its native grid search feature to test the combination of the following methods in our training sample:
    - **Filter** based techniques:
        - KBest
        - Variance Threshold
        - Select Percentile
    - **Wrapper** based techniques:
        - Recursive Feature Elimination
        - Forward Sequential Feature Selection
        - Backwards Sequential Feature Selection

Data Preparation - Outliers Analysis

During this important step we concluded the following:

- **balance** - a few outliers, not due to errors
- **loanAmount** - a few outliers, not due to errors
- **nInhabitants** - one outlier, Praha's district (Czech's Republic largest metropolitan area)
- **averageSalary** - one outlier, Praha's district
- **unemploymentRate96** - one outlier, Praha's district
- **commitedCrimes95/96** - one outlier, Praha's district

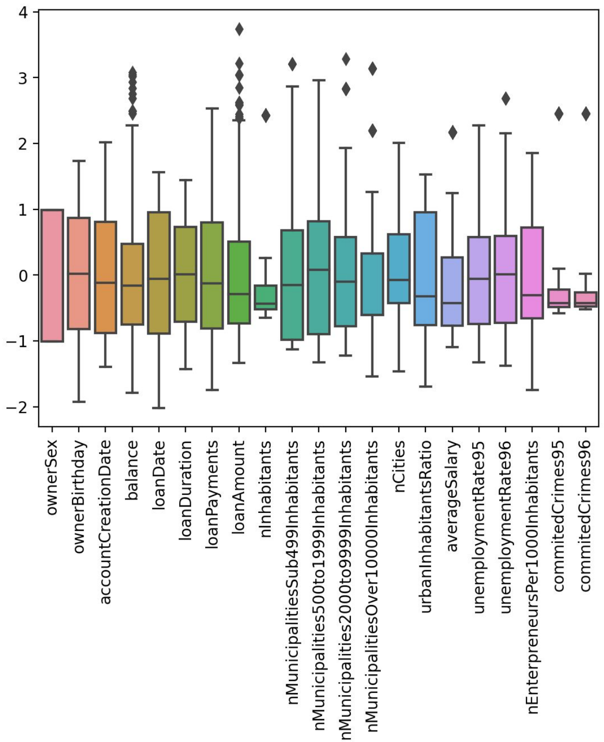


Figure 19 - Outliers analysis plot
*(**note:** the balance feature is the account balance from the last transaction made before the loan date)*

Data Preparation - Feature Engineering

From the original data we engineered the following features:

- isShared (account has one disponent)
- accountBalance (balance of the account before loan)
- medianAmount (median of transactions amount)
- sumAllTransactions (sum of transactions amount)
- insurancePaymentsCount (num. of 'insurance payment' transactions)
- insurancePaymentsAverage (average of 'insurance payment' transactions amount)
- timesIntoNegativeBalance (num. times balance transitioned into negative)
- numTransactionsNegBalance (num. of transactions done with negative balance)
- numExternalBankTransactions (num. of transactions to another bank)
- maxTransactionAmountDistance (maxCredit + |maxWithdrawal|)
- sumSactionInterest (sum of 'sanction interest if negative balance' transactions amount)
- avgSanctionInterest (average of 'sanction interest if negative balance' transactions amount)
- hasStableIncome (more than three 'collections from another bank' transactions with equal amount)
- transactionAmountIQR (interquartile distance of all transactions amount)
- ratio (ratio between loan amount and the average salary of the client's district)

- maxWithdrawal
- maxCredit
- withdrawalCount
- cashWithdrawalCount
- creditCount
- numTransactions
- ownerSex

Experimental Setup Pipeline

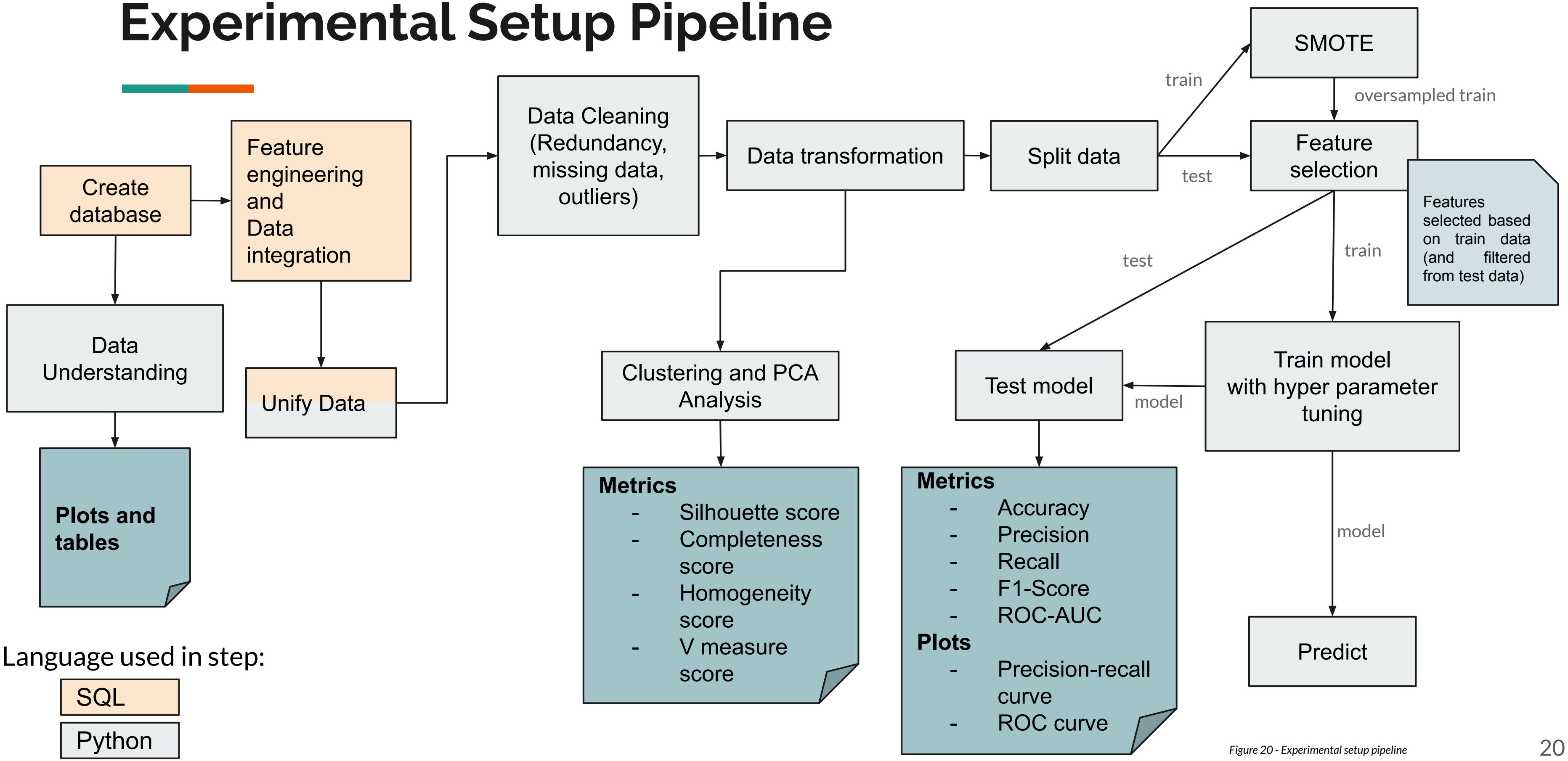


Figure 20 - Experimental setup pipeline

Descriptive Data Mining Problem

Based on the dataset, is a client a good client or a bad one?

Experimental Setup (1/2)

Algorithms Tested:

- **KMeans -** it computes K centroids and repeats until the optimal centroid is found (such that the sum of the squared distances between the data points and the centroid is as small as possible), as such, it assumes that clusters are convex shaped. Since we're working with a binary classification problem **we require two centroids**
- **DBSCAN -** it views clusters as areas of high density separated by areas of low density, as such, it can find clusters of any shape
- **MeanShift -** it's an iterative sliding-window-based algorithm that attempts to find areas of high density, similar to DBSCAN

Hyperparameter Tuning:

- **KMeans -** we experimented: altering the method for the initial centroid selection, **random or through an empirical probability distribution of the points** (KMeans++); choose **different iteration values**, 50, 150, 300 and 500; testing **a different algorithm named *elkan*** that proved to be more efficient through the use of the triangle inequality theorem
- **DBSCAN -** we experimented: **different maximum distances** between two samples for one to be considered as in the neighborhood of the other (0.1, 5 and 10, with euclidean distance metric); altering the **number of samples in a neighborhood** for a point to be considered as a core point (1 through 10); testing **different NearestNeighbour algorithms, *ball_tree*, *kd_tree* and *brute***
- **MeanShift -** given the simplicity of this algorithm it didn't require any relevant parameter tuning

Incremental improvements have been made through the hyperparameter tuning previously explained.

Experimental Setup (2/2)

Performance Measures:

- **Silhouette Score** - it evaluates how clearly separated are the clusters without any knowledge of the dataset. We didn't use it since we can obtain very well separated clusters that don't represent good and bad clients correctly.
- **Completeness** - it evaluates if all good and bad clients are grouped in two distinct clusters. We're working with two classes, therefore, ideally, we want to separate all clients into two distinct groups/clusters, but that may not be possible due to the nature of our dataset, for instance, we can have two clusters that only represent good clients. As such, we didn't rely much on this performance measure but agree that it could be used if we haven't found a better alternative.
- **Homogeneity** - it evaluates if all clusters contain only good or bad clients. Reiterating, our objective is to divide the dataset into two (or close to two) clusters if possible, we could have high homogeneity with a overfitted model with a large amount of clusters but that wouldn't show any relevant results. As such, we didn't rely on it exclusively.
- **V-measure** - this measure is the harmonic mean between completeness and homogeneity scores. Our objective is to not only separate good and bad clients into the least amount of clusters but to separate them accurately, in other terms, we want to achieve a balance between completeness and homogeneity scores. As such, we choose this metric as the main measure to evaluate our results.

NOTE: We use the loan status information as an heuristic to determine whether a client is good or bad.

Results (1/2)

- **DBSCAN** proved to be more performant according to our target metric (v-measure score), by identifying bad clients as outliers.

- On the other hand, **KMeans** achieved the highest silhouette score since it managed to cleanly separate the dataset into two clusters, but classifies many clients incorrectly (because good and bad client overlap), resulting in a lower v-measure score.

Model	silhouette score	completeness score	homogeneity score	v-measure score
KMeans	**0.85127**	0.44860	0.13912	0.21237
DBSCAN	0.81340	**0.49933**	**0.20542**	**0.29110**
MeanShift	0.55671	0.20816	0.19766	0.20277

Table 2 - Model Performance Comparison

Results (2/2)

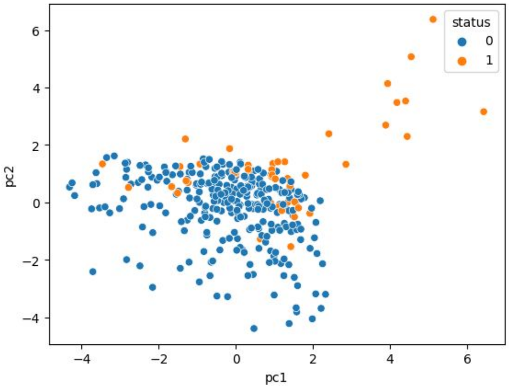


Figure 21 - PCA based on the original data

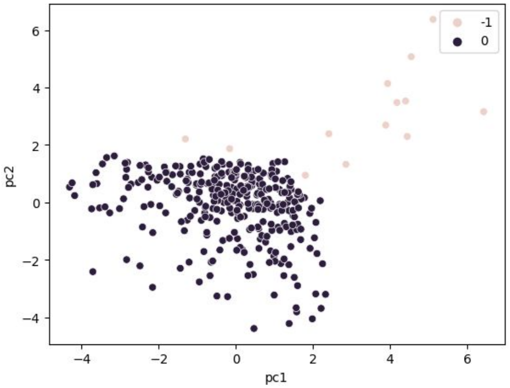


Figure 22 - DBSCAN Clustering

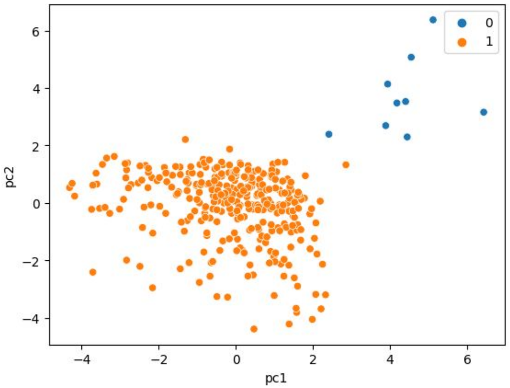


Figure 23 - KMeans Clustering

- To visualize the distribution of our data, and subsequently the predictions made by our clustering algorithms, we had to project it into a two dimensional space by performing Principal Component Analysis (PCA), as we can see in Figure 20
- We can perceive that the bad clients (with status=1) heavily overlap with the remaining customers making it difficult to discern which class an instance belongs to. Therefore, it was not surprising that our algorithms performed relatively poorly, finding only some of the bad clients, as demonstrated by Figures 21 and 22

Predictive Data Mining Problem

Will the client pay the loan or not?

Experimental Setup (1/2)

Hyperparameter Tuning:

- We used grid search with the ROC-AUC metric as the scoring function and stratified k-fold (5 splits) on the test split.

Models Trained:

- **Decision Tree** classifier
  - It is an simple and understandable model, based on binary data splits. It served as our baseline classifier during the duration of the project due to its simplicity and resistance to data anomalies.

- **K-Nearest Neighbors** classifier
  - It is an simple classifier that determines the class of an instance based on its nearest neighbors.
  - Due to its definition this classifier is extremely sensitive to the SMOTE technique, which led to completely opposite predictions based on whether or not the technique was applied.

- **Gaussian Naive Bayes**
  - This method allowed us to apply the Naive Bayes model in our data set, which is mostly made up of numerical attributes which the typical implementation does not support, by assuming a normal distribution of all the features.
  - This model is not ideal because it assumes the features are completely independent from each other, which we cannot guarantee, even given our feature selection methods.

Experimental Setup (2/2)

Models trained:

- **Random Forest** ensemble algorithm
  - It improves upon the original Decision Tree (DT) by combining a multitude of different DT classifiers.

- **Support Vector Machine** classifier
  - The SVMs are binary linear classifiers, that can be extended to non-linear classification problems by using the kernel trick. We tested different kernels to find which bias better fit our data, these included the standard linear kernel, a polynomial kernel (3rd degree), sigmoid kernel and radial basis function kernel.

- **AdaBoost** ensemble algorithm
  - As the name implies is an adaptive boosting ensemble algorithm in which a sequence of models are trained sequentially and each model focuses on the mistakes made by the previous models. As the base model we decided to use Decision Trees, in order to achieve a more direct comparison to the Random Forest classifier.

Results (1/2)

Model	precision	recall	f1-score	auc	fit time (s)	predict time (s)
Decision Tree	0.3	0.6	0.4	0.7197	0.0039	0.0011
KNeighbors	0.2222	**0.8**	0.3478	0.8852	0.0021	0.0033
Gaussian Naive Bayes	0.25	0.25	0.2	0.8197	0.0022	0.0011
Random Forest	0.5	0.6	**0.5455**	**0.8951**	0.1929	0.0189
SVM (RBF kernel)	0.3	0.6	0.4	0.8262	0.1086	0.0020
AdaBoost	**1**	0.2	0.3333	0.8525	0.1898	0.0103

Table 3 - Model Performance Comparison

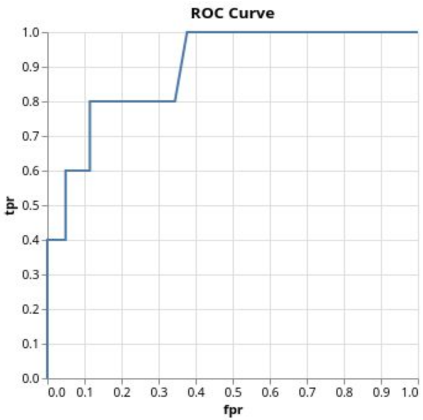


Figure 24 - ROC curve obtained from Random Forest classifier

Results (2/2)

We compared AUC score of our best model (Random Forest) to all the other models that were trained using an 5 x 2 cross validated t-test.

Null-hypothesis: Both classifiers are statistically equal.

At the significance level a=0.2 (with 5 degrees of freedom) we can reject the null hypothesis for the Decision Tree and SVM models, because the t-values are outside the confidence interval at that threshold.

To distinguish between the remaining models, an empirical evaluation method was used, by submitting the predictions of each model and analysing their public scores in the Kaggle competition.

Model	t-value
Decision Tree	3.20811
KNeighbors	1.10522
Gaussian Naive Bayes	1.46529
SVM	1.98199
AdaBoost	0.22254

Table 4 - Comparison of models against Random Forest classifier using a 5 x 2 cross validation t-test

Explainable Decision Tree (1/2)

Decision Tree obtained with a maximum depth of 5 and 8 leaf nodes. This model is simple, explainable but its performance is considerably lower than the one demonstrated by the more complex models

precision	recall	f1-score	auc
0.4	0.4	0.4	0.7295

Table 5 - Explainable Decision Tree performance

		Predicted: Loan not paid	Predicted: Loan paid
Real	Loan not paid	2	3
Real	Loan paid	3	58

Table 6 - Explainable Decision Tree confusion matrix

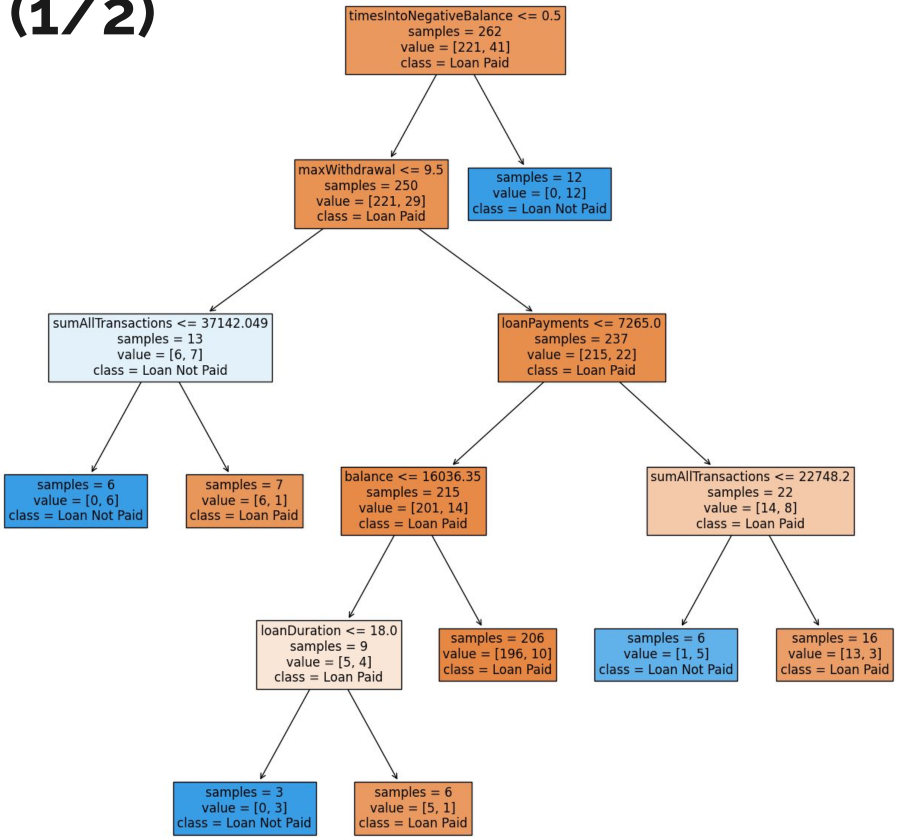


Figure 25 - Explainable Decision Tree

Explainable Decision Tree (2/2)

The decisions made by our explainable model are easily interpreted.

For instance, the clients that go into debt (whose balance went below zero) are very likely to default their loans. Other big factors that contribute to unpaid loans include: having a small volume of transactions (*sumAllTransactions*); having high monthly loan payments (*loanPayments*).

Even if the model isn't as performant as the other ones, we can still extract some business value from the information it provides. For example, since an high monthly payment amount contributes to loan defaulting, the bank can instead propose a plans with more payments of lesser monthly value to their customers.

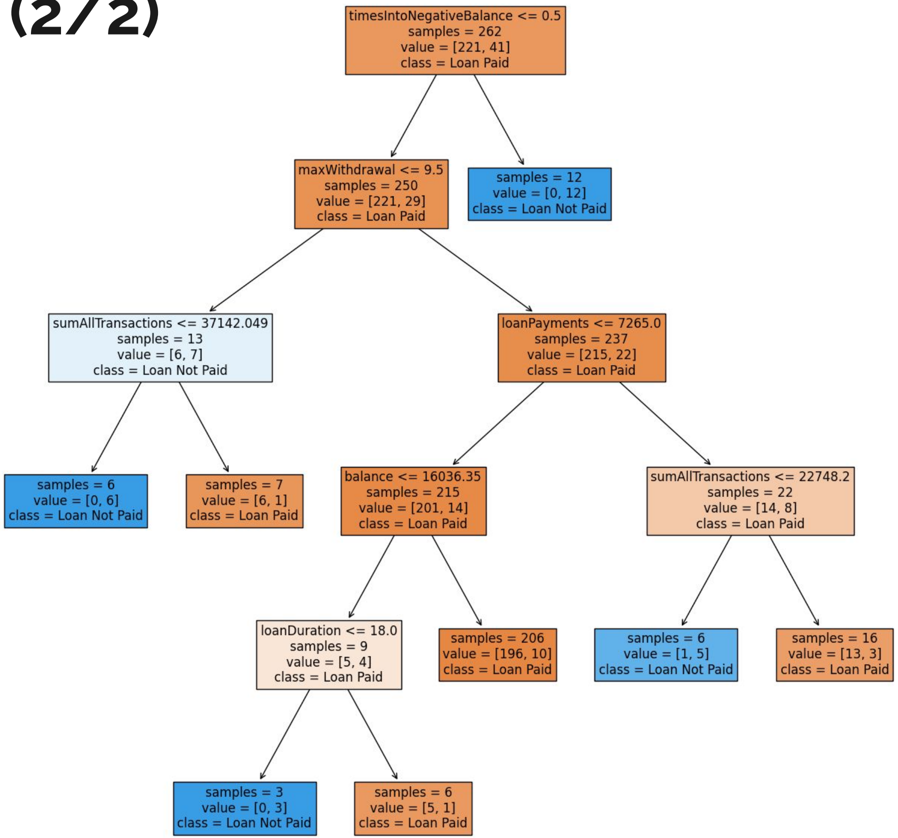


Figure 26 - Explainable Decision Tree

Methodology, plan and collaboration

- We followed the CRISP-DM methodology presented in the theoretical classes, illustrated in Figure 27
- We backtracked regularly to the early stages of the CRISP-DM to improve the results
- Our project log/report was weekly updated, as recommended
- Some of the work (mainly at the beginning) was developed synchronously, while the remaining was divided and organized through GitHub issues. For instance, anytime we splitted the work of each phase by assigning a group member a group of tables, models and algorithms a new issue was created
- Furthermore, we used DVC to track the results obtained in each pipeline execution so we could compare different models and techniques more easily

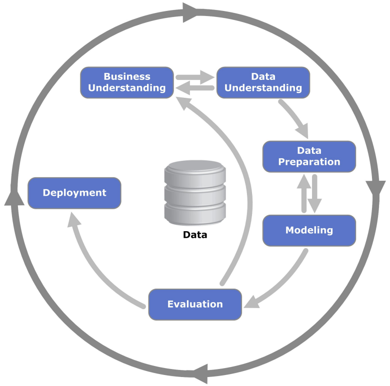


Figure 27 - CRISP-DM Methodology

Week	Main Accomplishments
3/10	Data and Business understanding
10/10	
17/10	Data Preparation
24/10	
7/11	Modeling Setup
14/11	Modeling Evaluation
21/11	Modeling with Kaggle Dataset Descriptive Problem Discussion
28/11	Presentation and Discussion
5/12	Report Finalization

Table 7 - Project Timetable

Conclusions, limitations and future work

- All our data mining goals were successfully met, which translates into successful business goals

- Feature engineering was the single most important step and lead to the greatest increases in model performance

- Despite trying to act in accordance to European regulations, the only explainable model we could achieve was through the Decision Tree classifier with very limited depth and leaf nodes that yielded substantially worse results than the other models tested, therefore discarded during the competition

- To apply this work in a real world scenario we could deploy simple models, such as Decision Trees, monitor their performance and slowly add more complexity by either increasing their depth or by testing different algorithms and hyperparameters. This would slowly build trust that our less understandable models were actually making accurate predictions