Business loan default prediction

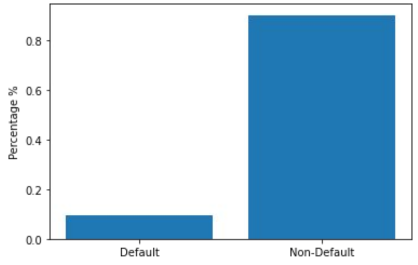
2022-05-22

Exploratory Data Analysis

- The case data set contains of total **4929** records.
- Out of those some of the data fields contain missing data:

○ uc_risk_class:	**3176** data points missing (or **76.66%**)
○ net_turnover:	**891** data points missing (or **21.51%**)
○ person_scoring:	**44** data points missing (or **1.06%**)
○ company_rating:	**17** data points missing (or **0.41%**)
○ incorporation_date:	**15** data points missing (or **0.36%**)

- About 10% of records are Default cases



Data Cleaning

- Columns containing dates ***applic_date*** and ***incorporation_date*** do not seem to be that useful for further analysis on their own. However the difference between those dates - number of days from the incorporation date until the company applied for the loan (***incorp_to_applic_days***) should have more value.
- ***uc_risk_class*** - missing data is a mix of strings and empty fields. Data type fixed and missing data converted to NaN
- ***net_turnover*** column is missing ~21% of data. We can try to replace missing data with 0 instead of dropping those rows. This might potentially improve the accuracy of further analysis since we will retain a lot of other valuable data
- ***company_rating*** as well as ***person_scoring*** both contain negative values, which are not actual scores, but rather some "error codes". Since the meaning of those “error codes” is unknown the negative values are converted to NaN
- After above steps the missing data overview looks as:

uc_risk_class:	3210 data points missing (or 81.55%)
person_scoring:	419 data points missing (or 10.65%)
company_rating:	292 data points missing (or 7.42%)
incorp_to_applic_days:	15 data points missing (or 0.38%)

- ***uc_risk_class*** is missing a lot of data (~82%). At the same time it has 51% correlation with the ***company_rating*** column. Should be safe to drop ***uc_risk_class*** at this stage to get better accuracy with the rest of the data
- Last step in data cleaning is to drop all remaining rows containing missing data. That leaves us with **4234** records, which is ~86% of the initial records

Treatment of Imbalanced Data

In provided data ~90% are the non-default cases, while only about ~10% are default.

Such data imbalance may lead to poor prediction of new observations of the minority class (“default”).

To improve the performance of the algorithms we can use certain sampling techniques to ensure nearly equal weights of minority and majority classes.

- **Data Undersampling**
  Employ **NearMiss** algorithm to bring down the amount of the non-default data
  - *Pros*: we use only real data for model training / evaluation
  - *Cons*: dataset gets significantly reduced (down to ~10%). It may lead to poor accuracy of predictions
- **Data Oversampling**
  Employ Synthetic Minority Oversampling Technique (SMOTE) to generate synthetic data for defaulted data
  - *Pros*: working with full dataset, which is good to get better accuracy of predictions
  - *Cons*: synthetic data added may affect the quality of the predictions

MinMaxScaler is used to normalize all the data in order to avoid any of the variables biasing the prediction

Data Modelling: Model selection

From the nature of the test case and preliminary data analysis we know that our problem is a binary classification problem with lots of non-linear relationships.

Most suitable classification methods for the task:

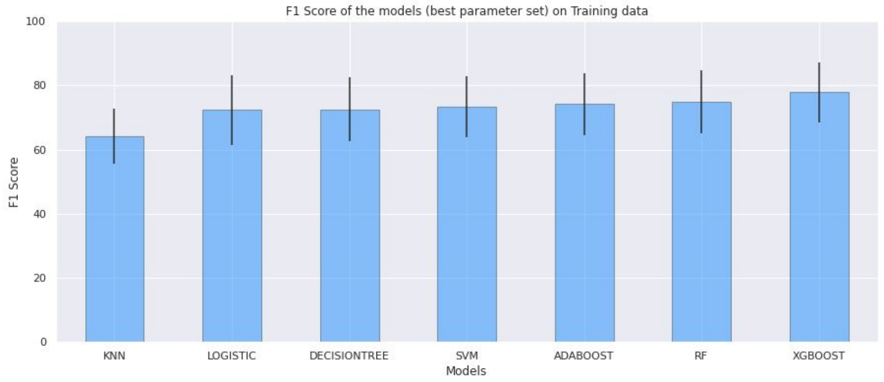
- **k-nearest neighbors**
  uses proximity to make classifications or predictions about the grouping of an individual data point
- **Logistic Regression**
  aims to classify an observation based on its modelled posterior probability of the observation belonging to a specific class
- **Support Vector Machines**
  objective is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points and has the maximum margin
- **Decision Trees**
  binary splits the feature space into subsets in order to divide the samples into more homogeneous groups
- **AdaBoost** (Adaptive Boosting)
  uses an iterative approach to learn from the mistakes of weak classifiers, and turn them into strong ones
- **XGBoost** (eXtreme Gradient Boosting)
  One of the advantages of XGBoost is its scalability and faster model exploration due to the parallel and distributed computing

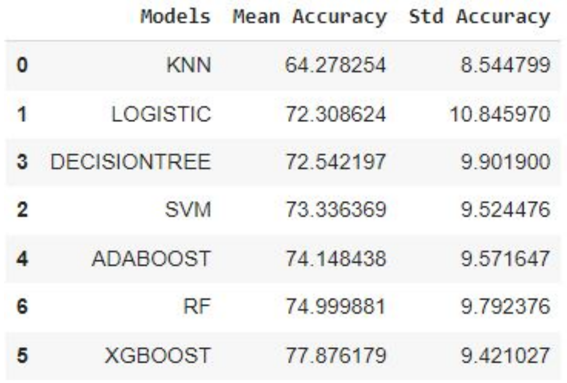
For model training and testing the dataset is split as: Train: 80%, Test: 20%

For evaluation the *F1*-score was used as the measure of test’s accuracy

Data Modelling: using undersampled data

As expected, the accuracy of the modelling with the undersampled data was rather poor for all algorithms.

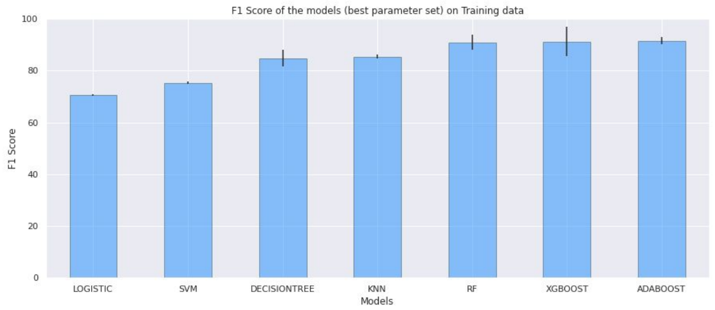


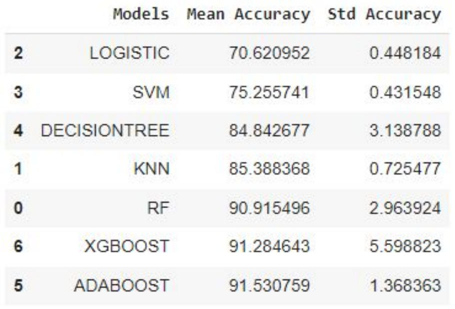


	Models	Mean Accuracy	Std Accuracy
0	KNN	64.278254	8.544799
1	LOGISTIC	72.308624	10.845970
3	DECISIONTREE	72.542197	9.901900
2	SVM	73.336369	9.524476
4	ADABOOST	74.148438	9.571647
6	RF	74.999881	9.792376
5	XGBOOST	77.876179	9.421027

Data Modelling: using oversampled data

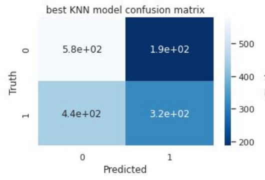
With the oversampled data the F1 score used for evaluation got significantly improved for some of the algorithms

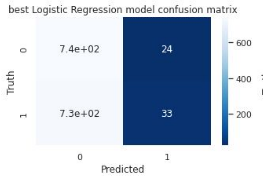


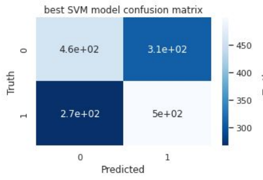


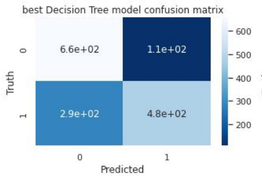
	Models	Mean Accuracy	Std Accuracy
2	LOGISTIC	70.620952	0.448184
3	SVM	75.255741	0.431548
4	DECISIONTREE	84.842677	3.138788
1	KNN	85.388368	0.725477
0	RF	90.915496	2.963924
6	XGBOOST	91.284643	5.598823
5	ADABOOST	91.530759	1.368363

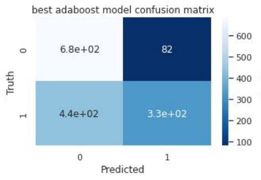
While the accuracy of Random Forest, AdaBoost and XGBoost exceed 90%, the level of false negative results is rather high (with lowest for XGBoost)

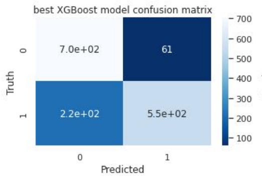


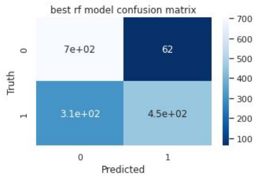






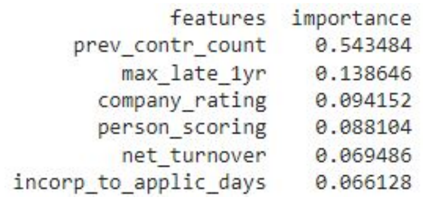






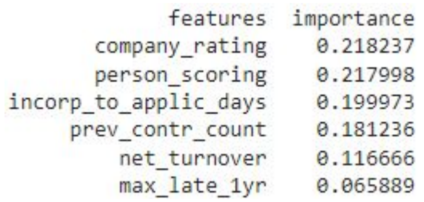
Data Modelling: Feature importance

- **AdaBoost**: feature importance was not identified for some reason
- **XGBoost**:



```
           features  importance
    prev_contr_count    0.543484
        max_late_1yr    0.138646
      company_rating    0.094152
      person_scoring    0.088104
        net_turnover    0.069486
incorp_to_applic_days   0.066128
```

- **Random Forest**:



```
           features  importance
      company_rating    0.218237
      person_scoring    0.217998
incorp_to_applic_days   0.199973
    prev_contr_count    0.181236
        net_turnover    0.116666
        max_late_1yr    0.065889
```

Although XGBoost algorithm had slightly higher accuracy (91.3%) compared to Random Forest (90.9%), the identified feature importance looks more intuitive for the RF algorithm.

Next steps to improve predictions

- Use larger dataset
- Try to further optimize parameters of the most successful model by extending the grid search around the best parameters
- Understand the reason for high level of false negative results and find a way to reduce it
- Evaluate model performance with the different data scaler
- Try to use Artificial Neural Network to evaluate its performance for the case