Part I

Predicting Churners from Banking Data

1 Predictive Modeling on Tabular Data

1.1 Introduction

As mentioned in the Task Description, the goal of this project is to predict a churn target. The dataset available to us consists of financial data from customers of a Belgian bank-insurer. More specifically, we have at our disposal customer data for 3 months: Present customer data with their targets and, in addition, data for the same customers in the 2 previous months.

Churn prediction might seem like an easy task for our advanced algorithms, but because of the very nature of churning itself it can prove quite challenging. The biggest issue in detecting churners is that they are very few; and therefore the working dataset will always be highly imbalanced. Moreover, since customer data is mostly added manually, it is very often the case that there will be many missing or incorrect values.

In this assignment, we were called to tackle these problems and try to predict churners as correctly as possible. The complete steps that we followed are listed below.

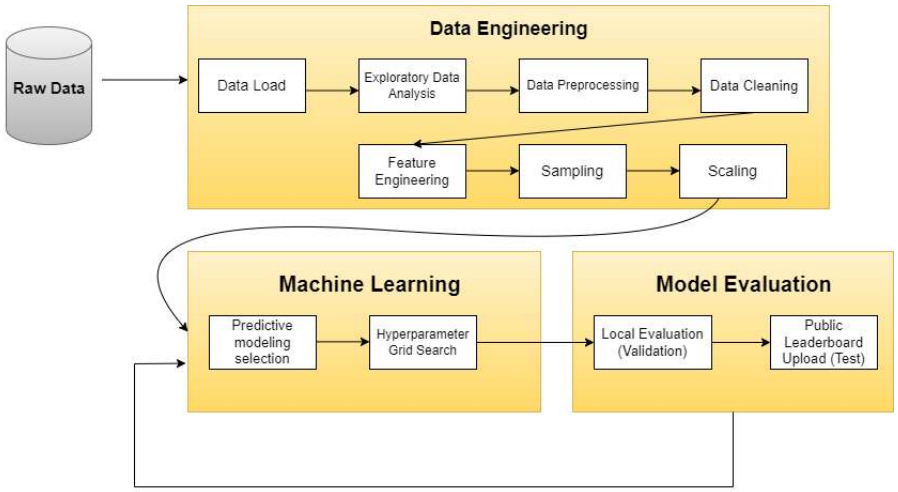


Figure 1: The Data Modelling Flowchart

1.2 Exploratory Data Analysis

Figures 2 , 3 , 4 below show a heatmap visualization of the correlation matrix of the feature spaces of month TT-3 , TT-2 and TT-1 respectively. Correlations where not considered with the target (churning rate column). The Pearson's correlation coefficient (r) utilized to compute correlations is a measure of linear correlation between two features. It's value lies between -1 and +1, -1 indicating total negative linear correlation, 0 indicating no linear correlation and 1 indicating total positive linear correlation. The correlation coefficient was computed for all feature pairs for the 3 past month data.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

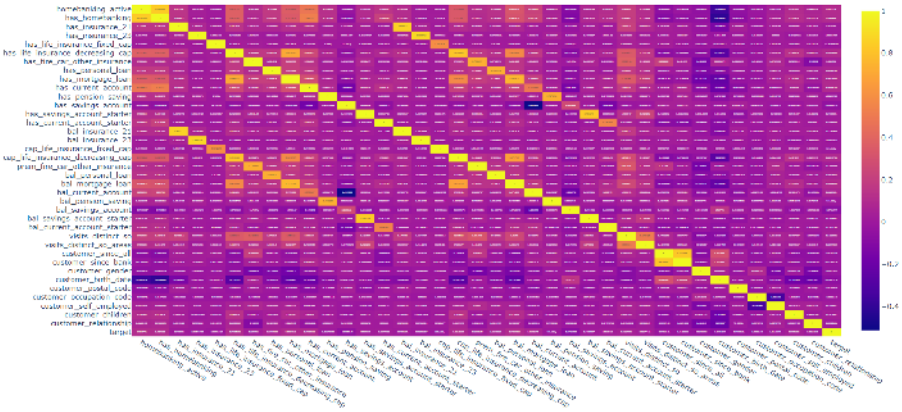


Figure 2: Correlation Matrix of column features of current month (TT-3) (Pearson's Correlation coefficient)

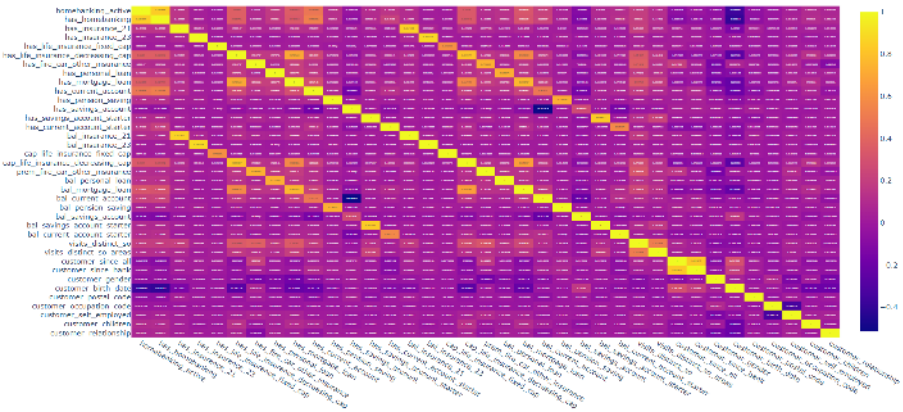


Figure 3: Correlation Matrix of column features of month (TT-1) (Pearson's Correlation coefficient)

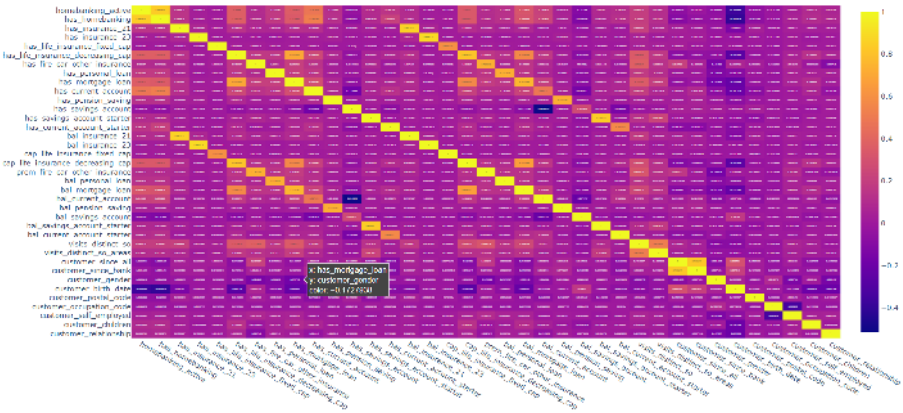


Figure 4: Correlation Matrix of column features of month (TT-2) (Pearson's Correlation coefficient)

1.3 Data Preprocessing and Cleaning

The given bank customer dataset columns(features) consist of a combination of continuous, boolean and categorical data types. Prior to data transformation and model deployment, data cleaning was deployed to unify the data representation to integers. Initially, data duplicates (customer details) are

identified and removed. Outliers on feature spaces are computed using the various methods listed in section 1.3.1 below. Following the data cleaning workflow, features are converted in a common integer representation prior to predictive model construction.

1.3.1 Outlier Detection and Treatment

The outliers were taken into consideration the continuous features (balances) of the datasets for each month.The methods used for outlier detection and handling are listed below:

- IQR range: Outliers outside the range: $R = [IQR - 3 * \sigma_X, IQR + 3 * \sigma_X]$, with IQR the inter-quartile range between data 3rd and 1st Quartile and σ_X the standard deviation of column feature X, were treated as outliers and discarded(replaced with NaN).
- RobustScaler: Sklearn RobustScaler routine transforms all data columns(features) in the range R listed above.
- Features representing age information (customer_since_all , customer_since_bank, customer_birth_date) which exceeded a specified age threshold : 120 were replaced with missing NaN values.

1.3.2 Categorical Feature Handling

Finally, categorical features(customer birth date, gender, postal code, occupation code, employment status , children count and relationship status) are transformed to a common integer type before we use them to perform predictive modelling. Also, for the categorical features: customer_since_all, customer_since_bank, customer_birth_date , the dates are converted to integer time difference numbers to today's date rounded to the nearest integer using a uniform %Y-M% format of pandas.to_datetime for its computation. Customer_children and customer_relationships categorical features were one-hot encoded to integers 0:M where M is the total count of distinct values the respective features can assume.

1.3.3 Mitigating Dataset Class Imbalance

Class Imbalance arises when there exist many more examples for one class, referred to as the majority class, where as the other class includes much less examples in total. According to the target column labels (churners/non-churner) tabulated in the given customer details month-3 (TT-3) data spreadsheet, the count of churners / non-churners are $N_{churners} = 1913, N_{non-churner} = 61784$imbalance ratio(IBR) [10] of (past) churning to non-churning customers is given by:

$$IBR = \frac{1913}{61784} = 0.0309 \approx \frac{1}{31}$$

therefore the month TT-3 dataset is highly skewed/biased toward the majority class, i.e. the non-churning customers. Class imbalance directly compromises prediction results on test data [12] , since the classifier is trained on dis-proportionally less examples of the minority class resulting in high accuracy by correctly predicting the majority class, but failing to capture the minority class due to the skewed imbalanced target distribution. The optimal data preprocessing and model classifier choice depends on the imbalanced ratio number as shown above [12]. To mitigate the effect of target

class imbalance, undersampling techniques were utilized - i.e. training on a lower subset of the majority class instances than the original dataset - Similarly to what was done before [3]. We considered various combinations of majority class undersampling and minority class oversampling methods with different post-sampling imbalanced ratios (sampling strategies). Further we utilized internal classifier parameter configuration(assigning weights to logistic regression data inputs to balance their contribution according to class), ensemble learning and boosting(weak classifier combinations) to address the imbalanced class distribution problem, as recommended before [11]. The statistics(histogram) of each column feature was considered for the TT(current) month of the training dataset to for exploratory data analysis and inform the construction of the analytics process pipeline (Data Cleaning, Data Transformation, Data Analysis, Prediction Model Deployment).

1.3.4 Sampling Strategies

The imblearn library for machine learning in imbalanced datasets was utilized to perform two types of data sampling prior to model selection and hyperparameter tuning(Fig. xyz):

- Undersampling: Reduction of training data labelled as the majority class(non-churners) was performed with RandomUndersampler which randomly choses data samples labelled as minority class without replacement. Given that the IBR (minority to majority class) ratio given by eqn.**??** is $IBR \approx 0.03$ the sampler was called (with sampling_strategy = 0.3) to sample accordingly to obtain a post-sampling $IBR_{desired} = 0.3$.
- Oversampling: Augmentation of the dataset with synthetic samples of the minority class(churners) was performed with the SMOTE, ADASYN. In particular SMOTE creates synthetic data instances along the line segments joining the minority class nearest k neighbors [9], where k is initialized according to sampling_strategy called.

1.3.5 Handling Missing(NaN) values of training features

Missing values in training (column) features were handled initially by interpolant imputer, kNN-Imputer and further by iterative imputation using the sklearn [5] library.Fig. 6 below visualizes the difference of linear interpolation vs nearest neighbour approaches considered for missing value interpolation. For imputation we utilized the Iterative Imputer which was determined as the most efficient imputation method since all data features of given bank clients iteratively contribute to estimating the missing(NaN) values. The imputer operates by iteratively modelling column features with missing values as regression targets with the remaining column features as predictors.

In Fig.5 a missing values plot for month 3 (TT) is illustrated. It is observed that columns: customer_education, customer_children , customer_relationship contain the largest percentage of missing NaN values.

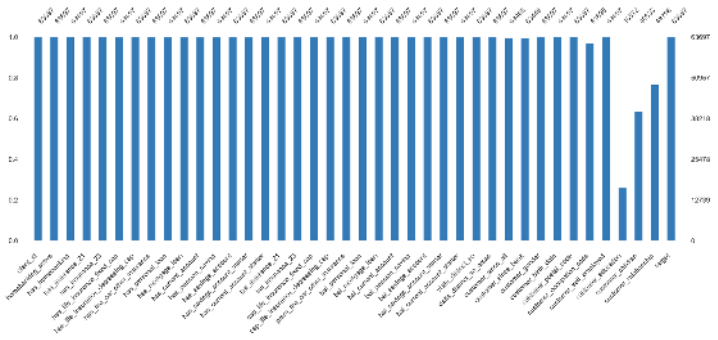


Figure 5: Percentage and Count Bar plot of non-NaN features in month-3 excel spreadsheet

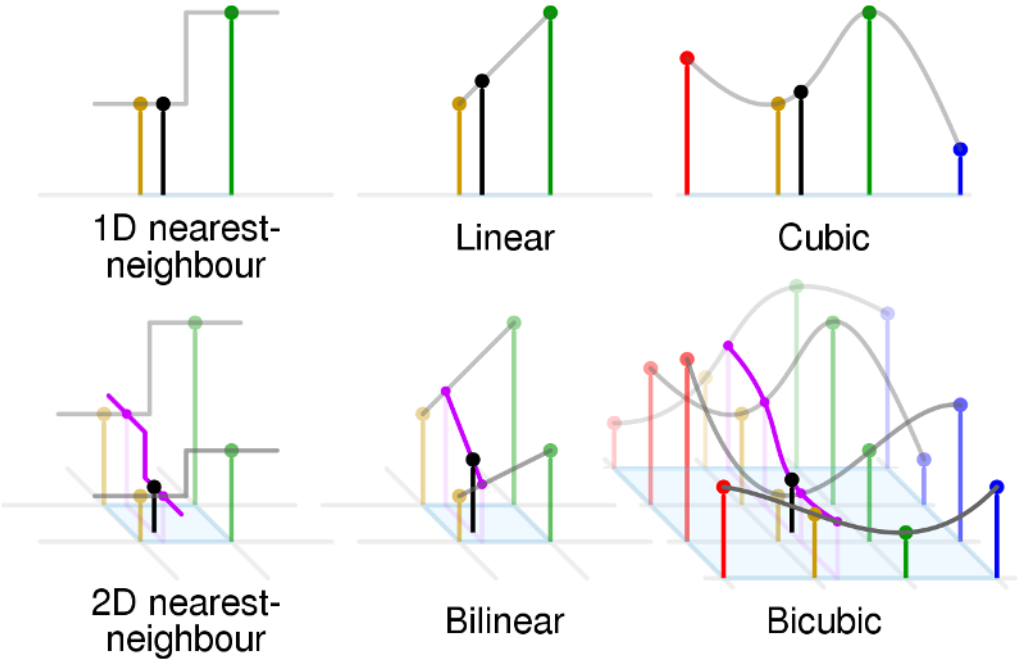


Figure 6: Interpolation Methods considered for NaN value imputation.

1.4 Feature Engineering, Feature Selection

As mentioned before, feature space multicollinearity(linear correlation) can be utilized to achieve higher data compression which as result can enable selection of simpler models [1] , which can yield better predictions according to Occam's Razor and Minimum Description Lengh (MDL) criterion. Adhering to the Occam's Razor model selection principle [4], we removed a single instance of each tuple of collinear features identified via the .corr() method of pandas.

Correlations were further used to inform the choice of feature engineering pipeline: Chi2-test used for removal of features with high correlation and ANOVA for identification and removal of comparatively non-discriminative features (low variance, overlapping distributions) for classification [2].

Further, sklearn in-built methods for determining the optimal minimum set of features required to maximize performance on a pre-set simple model(Linear Regression) with default hyperparameters were considered as listed below:

- Recursive Feature Elimination (RFE): Selects n out of N features of given training dataset, to maximize training accuracy on the default Linear Regression model by recursively pruning the least important features from the dataset(ranked by their corresponding model classification accuracy criterion)
- Recursive Feature Elimination with cross validation (RFECV):Similar to RFE described above feature removal on each iteration is guided by the cross validation score on a 5-fold dataset split.
- Select From Model:

After multiple experimentation iterations, the RFE feature engineering method was evaluated as the best based on validation accuracy from train-test split sklearn function and test performance on the public leaderboard, when opting for retaining n=20 out of N=38 total features of the dataset.

1.4.1 Past Month Features Integration

Features from months TT-2 , TT-1 were integrated in the TT training dataset utilizing two approaches. In the first approach the features capturing account balance information among different bank products of the customers in TT-2,TT-1 were integrated unprocessed in the TT training matrix. In the second approach, balance features in the TT month were removed and replaced by the balance difference between TT and TT-2 and TT-1:

$$balance(TT) \longleftarrow balance(TT) - balance(TT-1) - balance(TT-2)$$

to capture the temporal balance differential which was hypothesized to be a discriminative and robust feature for predictive modelling of the churning rate.

Another approach that was tried was to calculate the slope in the number of products owned per customer for each month. A negative slope in the third month would imply that the customer's products are decreasing, while a positive slope that the customer's products are increasing. Lagrange interpolation was used to calculate the function from the 3 points. Afterwards, the derivative of this polynomial was calculated to find the slope. Unfortunately, this method didn't produce very useful results, and therefore it was not included in this assignment.

1.4.2 Feature Scaling

In order to ensure correct classifier construction, as well as speed up training and convergence time, the features considered post feature engineering were transformed to standard ranges. The following scaling procedures were considered and implemented:

- StandardScaler(sklearn):Transforms features to zero mean, unit variance data, applies operation: $X' = \frac{X-\bar{X}}{\sigma_X}, \bar{X}$ =feature mean, σ_X = feature std.
- MinMax(sklearn):Transforms features to the range [0,1], by transforming data points between its minimum and maximum values, applies operation:
- RobustScaler(sklearn):Transforms features to zero median, constraining in the $[IQR - 3 * \sigma_X, IQR + 3 * \sigma_X]$range, applying the transformation:

Due to compatibility with specific classifier models and due to its ability to preserve the features underlying distributions, the MinMax Scaler was opted for the scaling method utilized in the data modelling workflow of this project.

1.4.3 Train-Validation Split

The sklearn train-test split routine was utilized to perform stratified train-validation split of the dataset , with the churners class (class=1) for stratification parameter, as seen on code block below:

```
# test train split
X_new, X_test, y_new, y_test = train_test_split(
                X_train, y, test_size=0.33, random_state=42 ,stratify =y)
```

1.5 Classifier Models, Pipelines

UnderSampling Strat.	**Classification Model**
Undersampling	Logistic Regression
Undersampling	RandomForest(depth = 4 , num_trees = 80)
Undersampling	RandomForest(depth =4) ,num_trees =100)
Oversampling(SMOTE)	Logistic Regression
Oversampling(ADASYN)	RandomForest(depth=4,num_trees=80)

Table 1: Model Training and Deployment of Pipelines implemented

The models aforementioned were deployed iteratively and the model hyperparameters tuned to obtain optimal accuracy , precision and f1-scores on the validation datasets.

1.6 Classification Results - Pipeline Evaluation

Table 2 below shows the validation accuracy,recall, precision,F1 score and ROC AUC obtained via different pipelines, as described in Table 1 above. Predictive models with higher precision are expected to perform better due to an increase in the hard-to-classify zero class(non-churners) examples performance,as explained in section 1.6.2 below.
However, F1 score in the validation set provides a robust metric for classification in imbalanced datasets [6] which is the harmonic mean of precision and recall. Higher F1-scores correspond to better model classification capability in imbalanced datasets.

Valid. Accuracy.	Train. Precision	Valid. Precision	Valid. F1 score	Test Precision(@250)	Test AUC
0.68	0.388	0.057	0.5	35	0.701
0.682	0.107	0.058	0.107	33	0.7031592
0.388	0.388	0.057	0.105	32	0.701
0.19	0.1	0.051	0.08	23	0.503
0.1	0.08	0.037	0.065	16	0.306

Table 2: Evaluation metrics on training and validation sets for pipeline number:

We observe according to tabulated metrics in table 2 above that the simple logistic regression model with balancing weight factors for data points results in the best classification performance in train-validation-test sets. Further undersampling the majority class has better performance comparing to Oversampling , potentially since the synthetic generated data do not result in predictors with discriminatory nature and ability to generalize to unseen data instances.

1.6.1 Pipeline Performance on Initial and Hidden Datasets

Dataset	Test Precision(@250)	Test AUC
Original	36	0.701
Hidden	26	0.70994031

Table 3: Evaluation of Best model performance on original and hidden datasets(public learderboard):

According to the table 3 above the precision score @250 correct churners predicted decreases substantially upon revelation of the hidden dataset. This occurs due to overfitting which prohibits the generalization capacity of the classification model to 100% of the examples tested on. Overfitting can be reduced by:

- Reducing Model Complexity: Opting for a model (e.g. Random Forest Classifier) with less learnable parameters (smaller number of estimators-trees, smaller depth) results in simpler models with higher generalization capability
- Feature Engineering/Elimination: Selecting Less features in the Feature Engineering stage of the modelling pipeline leads to simpler model selection, which (according to Occam's Razor) results in a better model with respect to performance in unseen test data.

As observed from table 2 , for pipelines 2 and 3 (random Forest classifier, max_depth 80 , 100 respectively) the Precision score and AUC decreases. This is an overfitting effect where increasing model complexity decreases the test performance of the classifier, which is further reflected in the increase of false positives in the confusion matrices.

1.6.2 Confusion Matrix

The confusion matrices plotted below correspond to the data modelling pipelines tabulated above (Table 1) , utilizing Undersampling as Sampling method and are evaluated on the validation set from train_test split. The confusion matrices give quantitative insight in the generalization performance of the modelling pipeline in unseen data instances. Due to the class conditional probability distribution skew (stemming from the class imbalance) the constructed classifiers are biased to inherently have high accuracy but low precision since it is bound to produce many False negatives(FN) but not the correct amount of False positives(FP) as they are biased to predict the majority class.
Therefore, a heuristic to evaluate classification performance corresponds to increasing the FP while decreasing the FN to a satisfactory precision level.

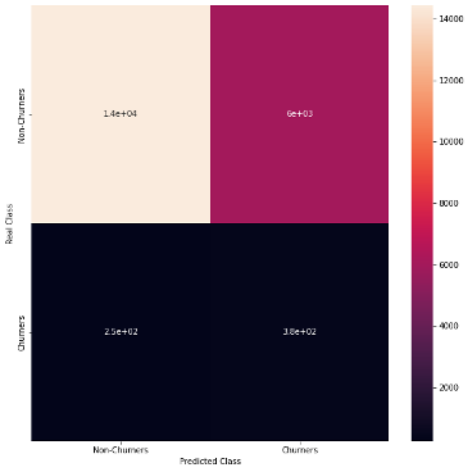
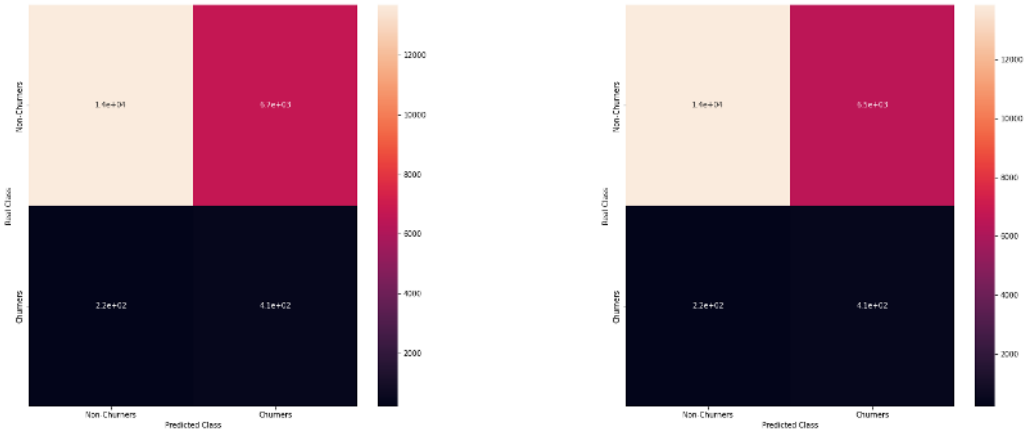


Figure 8: Confusion Matrix on Validation set using Logistic Regression Classifier Model and Undersampling on data.



(a) confusion Matrix for random Forest, num_estimators=80

(b) confusion Matrix Random Forest, , num_estimators=80

As observed from figure 8 confusion matrices above, increasing the total estimators of the ensemble of decision Trees(Random Forest) results in a more complex models which decreases the FP of the model. C

1.6.3 Grid Search

Grid search was performed on the Random Forest Classifier Hyperparameters to determine the optimal hyperparameter set with objective maximizing the precision on validation set. The code snippet below shows the parameters grid search is run on:

```
param_grid={'n_estimators':[int(x) for x in np.linspace(start=50,stop=100,num=11)],
```

```
'max_features':['auto','sqrt'],
'max_depth':[int(x) for x in np.linspace(start=1,stop=100,num=11)],
'min_samples_leaf':[1,2,3,5],
'min_samples_split':[2,5,10,15]}
```

1.7 Preliminary Results

1.7.1 Random Forest Pipelines

According to literature, Random Forest classifiers are suitable for classification tasks on structured tabular data, due to the low correlation between columns or features of the predictor set, utilizing the pre-processing feature engineering methods of the pipeline.

1.7.2 SVC Pipelines

Linear support vector machine pipelines consisted of feature engineering followed by min-max scaling followed by undersampling and classification with results tabulated above.

1.8 Conclusion

The tabular predictive modelling project is summarized as listed below:

- Model Simplicity Importance: As reflected in predictive modelling results(metrics), development of simpler models in terms of hyperparameters and model architecture results in better classification performance in customer details data not utilized during model training
- Feature Selection Importance It was observed that utilizing all raw data features without feature engineering(section 1.4) resulted in poor model performance on both validation (local evaluation) and test set (public leaderboard) performance. Selecting less , uncorrelated Features in the preprocessing stage of the workflow maximizes the predictive potential of the downstream classification models.