**INTRODUCTION TO DATA ANALYTICS**

**PROJECT REPORT**

**Group no. 14**

GROUP NO. 14

## **Data Overview**

The data contains 43 features and 83000 samples.

**G.Guna Sekhar (ME16B180)**

**Vivek Oommen (ME16B176)**

**Saye Sharan (ME16B177)**

**Rohith (ED16B009)**

**Battula Pranav (ME16B135)**

## **Objective:**

The task is to classify a person who needs credit from the bank will go default or not in the future based on various attributes of the person. This is to reduce the risk of default and loss the bank has to face for the credit given.

## **Data Overview:**

The data has in total 48 features (mostly quantitative), default\_ind feature (label) and 83000 samples. The 48 features specify different credit characteristics of a person.

## **Data Pre-Processing:**

Initially, there existed lot of missing data and variables had irregular datatypes so on looking the data, based on the values taken by a feature the datatypes were corrected and missing data were replaced by Nan values and imputation was carried out.

After going through the description of all the features, due to lack of proper domain knowledge and based on intuition most of the features seem important and so we selected our final set of features based on degree of missing values and also removed samples with missing values greater than a proportion.

We performed imputation using ‘MissForest’ from the missingpy library. The reason for not using a simple imputation strategy like Mean, Median etc is due to the very large no. of missing values resulting in poor model performances. So, a kind of intuitive method that captures patterns in the data and then predicting the missing values seems to be aligning with the data patterns and hence improves model performance (an assumption). So we used Random Forest (MissForest) for this purpose.

After imputation, further feature selection is done to improve model performance based on the correlation matrix of features and removed features with high correlation.

Finally, for modelling we used all the 83000 samples and we used 40 features.

## **Modelling:**

We have defined an objective function ‘Score’ which is used for comparing various methods and for tuning purposes:

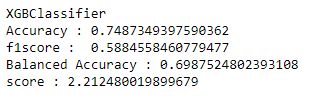
The reason for choosing such a function is that we want better accuracy, balanced accuracy as well as F1 score together since we know that increase in F1 score comes at a cost of loss in Balanced accuracy. The F1 score is weighted since because of the problem in hand i.e. we want to be careful about the default (priority) i.e. have a good recall score that in turn leads to good F1-Score. So, by this Score function recall, and accuracy are both taken care of.

For modelling we used 5 different classification models (each tuned) and finally performed stacking of the models to improve the performance by weighting (using a grid search) the probabilities of classes from all the models and the optimal set of weights are decided based on the ‘Score’ function.

## **Results:**

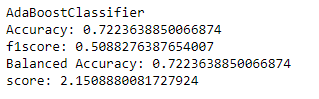
## **XGBoost Classifier:**

We have used XGBoost with the following parameters as ***max\_depth=3, estimators=250, scale\_pos\_weight = 1.9.*** The results can be summarised as follows:



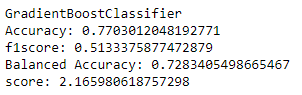
### **AdaBoost Classifier:**

We have used AdaBoost with the following hyper parameters as ***n\_estimators = 100.*** The results can be summarised as follows:



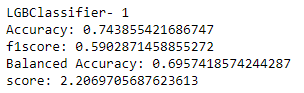
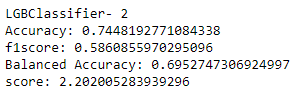
### **Gradient Boosting Classifier:**

We have used Gradient Boosting with the following parameters as ***n\_estimators = 200, max\_depth = 4, min\_samples\_leaf = 4.*** The results can be summarised as:



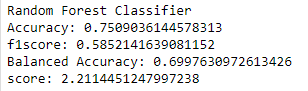
### **Light Gradient Boosting Classifier*:***

We have used Light Gradient Boosting with the default parameters but different boosting criteria. The results can be summarised as:

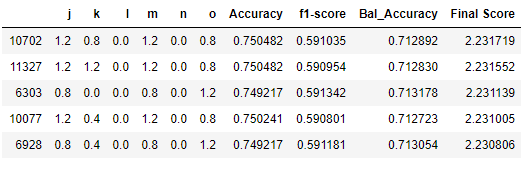
### **Random Forest Classifier:**

We have used the random forest classifier with the following parameters as ***n\_estimators = 300, max\_depth = 16, min\_samples\_leaf = 7***. The results can be summarised as:



### **Stacking:**

Our final model is the stacking of all the predicted class probabilities of above-mentioned models. As mentioned earlier, the probabilities are weighted and weights are assigned based on grid search in the interval (0,2) and optimal weights decided based on Score. A snip of the weights for each model and the best performances are shown below:



Weights and the corresponding classifiers:

j – XGB, k – ADB, l – GBM, m – Random Forest, n – LGBM- 1, o – LGBM-2

**From the above analysis, we get the region of weights where we further tuned and the weights finalised are j = 1.6, k = 0, l = 0, m = 1.2, n = 0.4, o = 2.0.**

The weights also tell us that inferior models are set to zero to have zero contribution.

**Our best model performed with the following results:**

***Accuracy: 0.749880***

***F1-Score: 0.592542***

***Balanced Accuracy: 0.714076***

***Score: 2.234260***

Further Improvements on Model performance can be done as follows:

* Better Feature Selection can improve the performance which is possible either by gaining domain knowledge or performing the brute force forward selection methods (which we performed but resulted in poor models).
* Outliers to be removed and this may improve the performance.
* Sampling of the data of the non-default class to introduce balance may lead to a better performance.
* Sampling of the whole dataset based on the Rank and plots and monotonicity may improve the model performance.