Micro-Credit Defaulter Model

Submitted by:

Vedant Singh Shankar

ACKNOWLEDGMENT

I would like to thank my SME, Mr. Nitin Mishra, for providing me the opportunity and guidance to work on this project. The sample data is provided to me from the client database of Flip Robo Technologies, which was shared to me for this project. Also, the following are the website links as a reference that helped me in the project.

Website Link:

- https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/
- https://www.listendata.com/2015/03/weight-of-evidence-woe-and-information.html
- https://www.researchgate.net/publication/257436122_A_Review_of_Feature_Selection_Methods_Based_on_Mutual_Information
- https://scikit-learn.org/stable/modules/permutation_importance.html
- https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html#lightgbm.LGBMClassifier

Contents

1.INTRODUCTION

1.1 Background

A Microfinance Institution (MFI) is an organization that offers financial services to low-income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on.

Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low-income families and are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.

Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients.

In this project we will be working on the dataset of one of the Telecom Industry. They are a fixed wireless telecommunications network provider. They have launched various products and have developed its business and organization based on the budget operator model, offering better products at

Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

1.2 Problem Statement

Telecom Industry are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is claimed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah).

The sample data has been provided to us from the client database. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

1.3 Summary of the project

- In this project we have applied 4 different kinds of supervised Machine Learning Algorithms to build predictive models for the use case.
- The dataset needed further wrangling and cleaning, but the main problem was the data imbalance, however, we have applied appropriate sampling techniques to deal with this issue.
- We have done various statistical analysis and used techniques to find the data correlation and relationship between output and input variables.

- Taking account of data imbalance, the hyper-parameters of models have been tuned, and used appropriate metrics for evaluating model performance.
- Considering the trade-off between Type-I and Type-II in machine learning classifiers, we have tried to find the optimal probability threshold for model prediction. Also using this optimal threshold, we have re-evaluated all models for finalizing the model for the business problem.

1.4 Motivation for the Problem Undertaken

The development of this project is important for taking better business decisions in regards to customer selection for providing loans, particularly in huge industries like telecom industry, where the global telecom services market size was valued at USD 1.74 trillion in 2019.

Also, it may help in reaching out the poor families with low sources of income by providing them loans and also giving the confidence to the telecom service providers to improve their mobile financial services (MFS).

2. Analytical Problem Framing

2.1 Modelling of the Problem

In this project, we have used 4 kinds of different Machine Learning Models that is based on supervised learning algorithm. Which works by learning the

historical data with associated labels or outcomes, and predicts the possible outcomes, in our case whether the customer is Non-Defaulter or Defaulter. The high-level overview of each algorithm that is used to model the data is as follows:

Logistic Regression

- Logistic regression models the probabilities for classification problems with two possible outcomes. It's an extension of the linear regression model for the classification problems.
- In this model, the probabilities describing the possible outcomes of a single trial are modelled using a logistic or **sigmoid** function.
- The optimization problem to minimize the generalized cost function involving $\mathbf{l_1}$, $\mathbf{l_2}$ and **Elastic-Net** regularization is as follows:

$$\min_{w,c} \frac{1-\rho}{2} w^T w + \rho \|w\|_1 + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1)$$

  Here, the minimization of $\mathbf{w^T w}$ and $\mathbf{\|w\|_1}$ effects in $\mathbf{l_2}$ and $\mathbf{l_1}$ regularizations respectively. Also, the parameter $\boldsymbol{\rho}$ brings the effect of both in $\mathbf{l_1}$ and $\mathbf{l_2}$ regularizations, known as **Elastic-Net** regularization.
- Logistic regression is easier to implement, interpret, and very efficient to train. It is also less inclined to over-fitting but it can overfit in high dimensional datasets. One may consider Regularization ($\mathbf{l_1}$ and $\mathbf{l_2}$) techniques to avoid over-fitting in these scenarios.

Gaussian Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable Bayes' theorem states the following relationship, given class variable **y** and dependent feature vector $\mathbf{x_1}$ through $\mathbf{x_n}$:

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

and we can use Maximum A Posteriori **(MAP)** estimation to estimate **P(y)** and $\mathbf{P(x_i|y)}$, the former is then the relative frequency of class **y** in the training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $\mathbf{P(x_i|y)}$.

Gaussian Naive Bayes implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Here, the parameters $\boldsymbol{\sigma_y}$ and $\boldsymbol{\mu_y}$ are estimated using maximum likelihood.

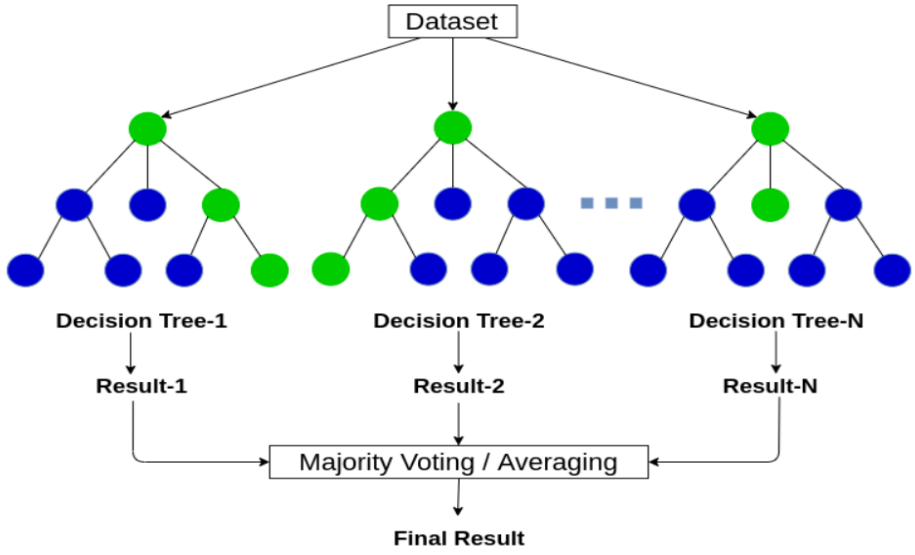
Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The **decoupling** of the class conditional feature

distributions means that each distribution can be independently estimated as a one-dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

Random Forest

Random forest, like its name implies, consists of a large number of individual **decision trees** which operates in Bootstrap aggregation, also called **bagging,** that is one of the **ensemble** techniques in Machine Learning. It can be used for both classification and regression problems. Here, each individual tree in the random forest gives out a class prediction and the class with the most votes becomes our model's prediction.

The individual decision trees typically exhibit high **variance** and tend to overfit. The injected randomness in forests yield decision trees with somewhat **decoupled** prediction errors. By taking an average of those predictions, some errors can cancel out. Hence, random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model.



Overview of Random Forest Algorithm

The above figure represents the high-level overview of working of Random Forest Algorithm, which is basically parallel decision trees working individually to yield out a prediction which is further averaged (or by finding the major votes), to give out the final prediction from the model.

LightGBM (Light Gradient Boosting Machine)

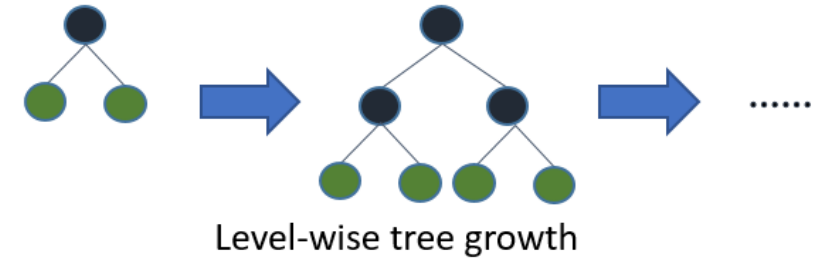
- It is also one of the ensemble algorithms that uses **boosting** technique to increase the predictive power. It is a gradient boosting framework that makes use of tree-based learning algorithms that is considered to be a very powerful algorithm when it comes to computation. It is also considered to be a fast processing algorithm.
- While other algorithms **trees** grow horizontally, LightGBM algorithm grows vertically meaning it grows **leaf-wise** and other algorithms grow **level-wise**. LightGBM chooses the **leaf** with large loss to grow. It can

lower down more loss (i.e. cost function of an algorithm) than a level wise algorithm when growing the same leaf.

- LightGBM is not for a small volume of datasets. It can easily overfit small data due to its sensitivity. It can be used for data having more than 10,000+ rows. There is no fixed threshold that helps in deciding the usage of LightGBM. It can be used for large volumes of data especially when one needs to achieve a high accuracy.

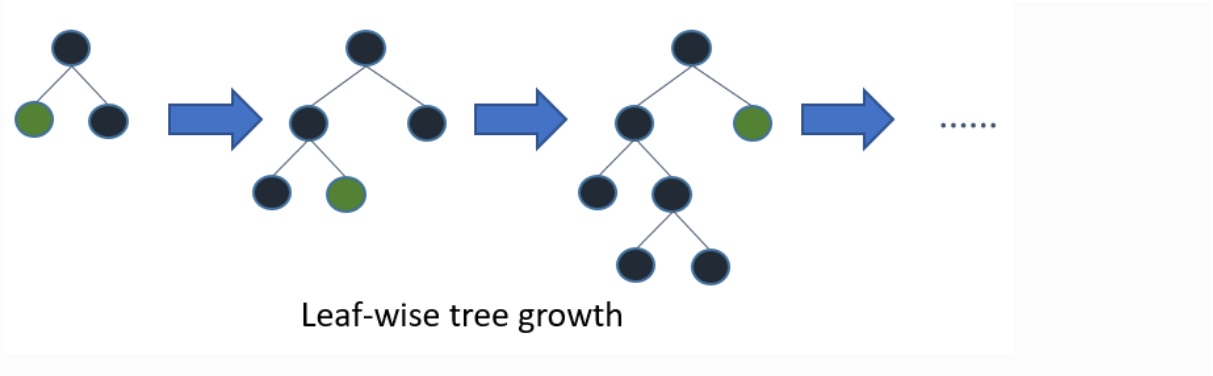
Optimization in Accuracy for LightGBM

Most decision tree learning algorithms grow trees by level (depth)-wise, like the following image:



LightGBM grows trees leaf-wise (best-first). It will choose the leaf with max delta loss to grow. Holding **leaf** fixed, leaf-wise algorithms tend to achieve lower loss (cost-function) than level-wise algorithms.

Leaf-wise may cause over-fitting when **data** is small, so LightGBM includes the **max_depth** parameter to limit tree depth. However, trees still grow leaf-wise even when **max_depth** is specified.



It has become difficult for the traditional algorithms to give fast results, as the size of the data is increasing rapidly. LightGBM is called "**Light**" because of its computation power and giving faster results. It takes **less memory to run** and is able to **deal with large amounts of data**. Most widely used algorithm in Hackathons because the motive of the algorithm is to get good accuracy of results and also brace GPU leaning.

2.2 Data Sources and Description

- The sample data has been provided to us from one of the client databases, which probably belongs to the telecom service providers
- The dataset is in csv format. The dataset is medium sized having 2,09,539 samples and 37 rows, and occupied roughly 34 MB of memory.
- Although it did not have any null values, but had many redundant features and data irregularities.

```
1  data.info()
```

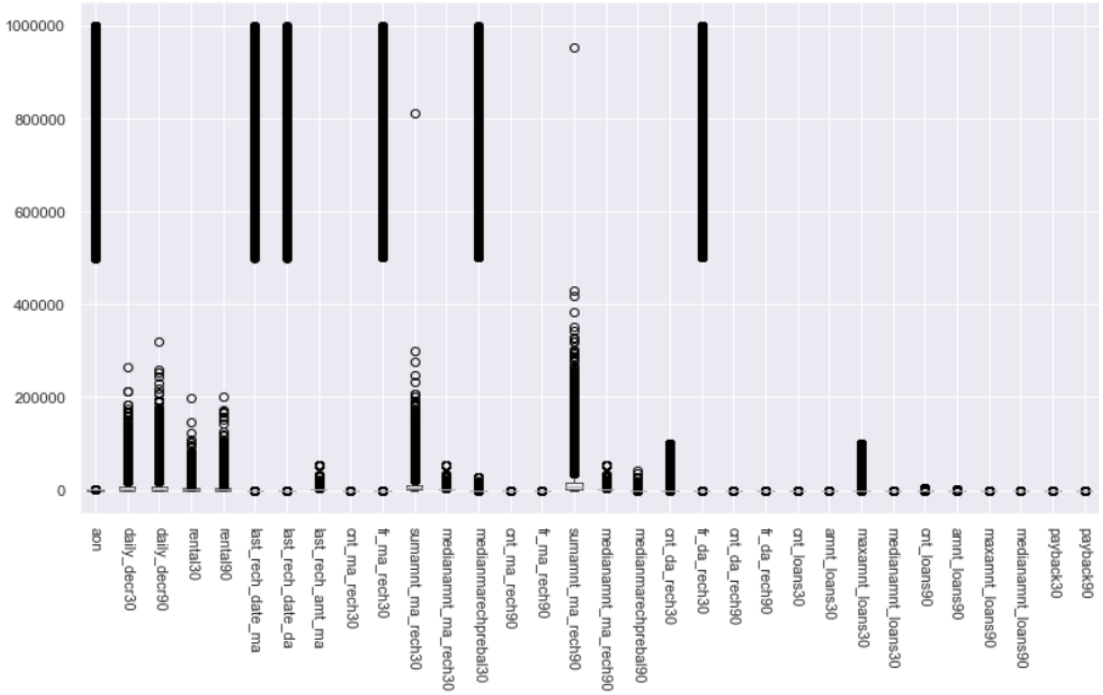
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 37 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Unnamed: 0              209593 non-null  int64
 1   label                   209593 non-null  int64
 2   msisdn                  209593 non-null  object
 3   aon                     209593 non-null  float64
 4   daily_decr30            209593 non-null  float64
 5   daily_decr90            209593 non-null  float64
 6   rental30                209593 non-null  float64
 7   rental90                209593 non-null  float64
 8   last_rech_date_ma       209593 non-null  float64
 9   last_rech_date_da       209593 non-null  float64
 10  last_rech_amt_ma        209593 non-null  int64
 11  cnt_ma_rech30           209593 non-null  int64
 12  fr_ma_rech30            209593 non-null  float64
 13  sumamnt_ma_rech30       209593 non-null  float64
 14  medianamnt_ma_rech30    209593 non-null  float64
 15  medianmarechprebal30    209593 non-null  float64
 16  cnt_ma_rech90           209593 non-null  int64
 17  fr_ma_rech90            209593 non-null  int64
 18  sumamnt_ma_rech90       209593 non-null  int64
 19  medianamnt_ma_rech90    209593 non-null  float64
 20  medianmarechprebal90    209593 non-null  float64
 21  cnt_da_rech30           209593 non-null  float64
 22  fr_da_rech30            209593 non-null  float64
 23  cnt_da_rech90           209593 non-null  int64
 24  fr_da_rech90            209593 non-null  int64
 25  cnt_loans30             209593 non-null  int64
 26  amnt_loans30            209593 non-null  int64
 27  maxamnt_loans30         209593 non-null  float64
 28  medianamnt_loans30      209593 non-null  float64
 29  cnt_loans90             209593 non-null  float64
 30  amnt_loans90            209593 non-null  int64
 31  maxamnt_loans90         209593 non-null  int64
 32  medianamnt_loans90      209593 non-null  float64
 33  payback30               209593 non-null  float64
 34  payback90               209593 non-null  float64
 35  pcircle                 209593 non-null  object
 36  pdate                   209593 non-null  object
dtypes: float64(21), int64(13), object(3)
```

Snapshot of data info table

2.3 Exploratory Data analysis and Data Pre-processing

- The numerical features contained negative values, which is not realistic according to the domain knowledge and can only have minimum value equal to zero. Therefore, assuming that the negative values are a typographical error, that represented the same value with negative sign, we have therefore changed only the sign of negative values to positive.
- Columns like **'Unnamed: 0'** and **'pcircle'** did not add any relevance in the data as column **'Unnamed: 0'** was only serial no. associated with each instance in data and **'pcircle'** has only one unique value, which is not useful for analysis and modelling. Therefore, we dropped these columns.

- We then defined a function that extracted date information like month, year and weekday, that had more than one unique value. Also, we have split the column **'msisdn'**(user mobile number), to extract information about a particular area code.



Boxplots of numerical features

- From above boxplots, we can see that all six features are having extreme values that is abnormally large to consider for modelling. Also, we should take care not to remove all outliers just on the bases of boxplot because removing all using standard methods like **z-score** or **IQR** will remove huge data. Therefore, from above plot, we have selected a particular threshold value like 4,00,000 above which we shall discard all values.
- Then further looking into the distribution of the variables, it is found that all columns were highly skewed with mean **skewness value** of **9.76** and maximum reaching up to **44.8**. Moreover, data transformation like

logarithmic or cubic transformation did not help much in reducing the skewness of the data.

- Although we will be using tree-based algorithms that are robust to outliers, but algorithms like Gaussian Naive Bayes and Logistic Regression are very sensitive to it, moreover, even outliers are important in some cases like fraud detection or generally when there are very less positive class (like in our case), therefore, to deal with high skewness and presence of outliers (if there is any), we have decided to bin the numerical data. The binning technique that we have applied here is **supervised binning**, that uses the target variable to bin the numerical/continuous variables, so that maximum information is retained in each bin about the target variable.

Correlation heatmap and Variance Inflation Factor (VIF) for Multi-collinearity

In multicollinearity, when multiple features are correlated to each other, then there arises feature redundancy, that both unnecessarily increases the feature space and also overfitting in certain models.

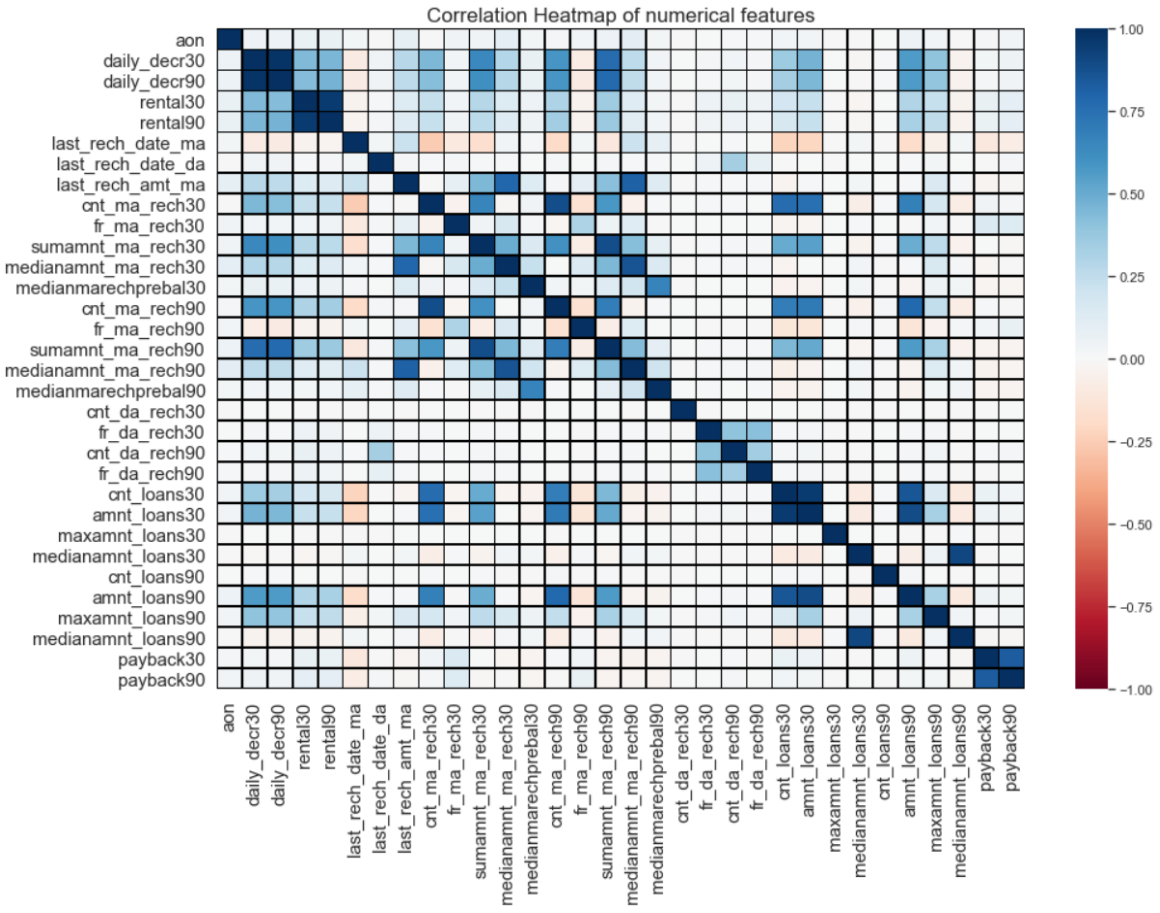
Although multi-collinearity in not a problem for a tree-based algorithm like Random Forest and LightGBM in regards to model's performance, but algorithms like Logistic Regression and Gaussian Naive Bayes are quite sensitive towards it. Moreover, multi-collinearity reduces model's interpretability, that is, we will not know definitely how a particular feature may affect the model.

Therefore, to counter this problem we have analysed the correlation heatmap and used statistical method like **VIF**, that determines the strength of the correlation between the independent variables. It is predicted by taking a variable and regressing it against every other variable. In other words, VIF score of an independent variable represents how well the variable is explained by other independent variables.

Also, $\mathbf{R^2}$ value is determined to find out how well an independent variable is described by the other independent variables. A high value of $\mathbf{R^2}$ means that the variable is highly correlated with the other variables. This is captured by the **VIF** which is denoted below:

$$\text{VIF} = \frac{1}{1-R^2}$$

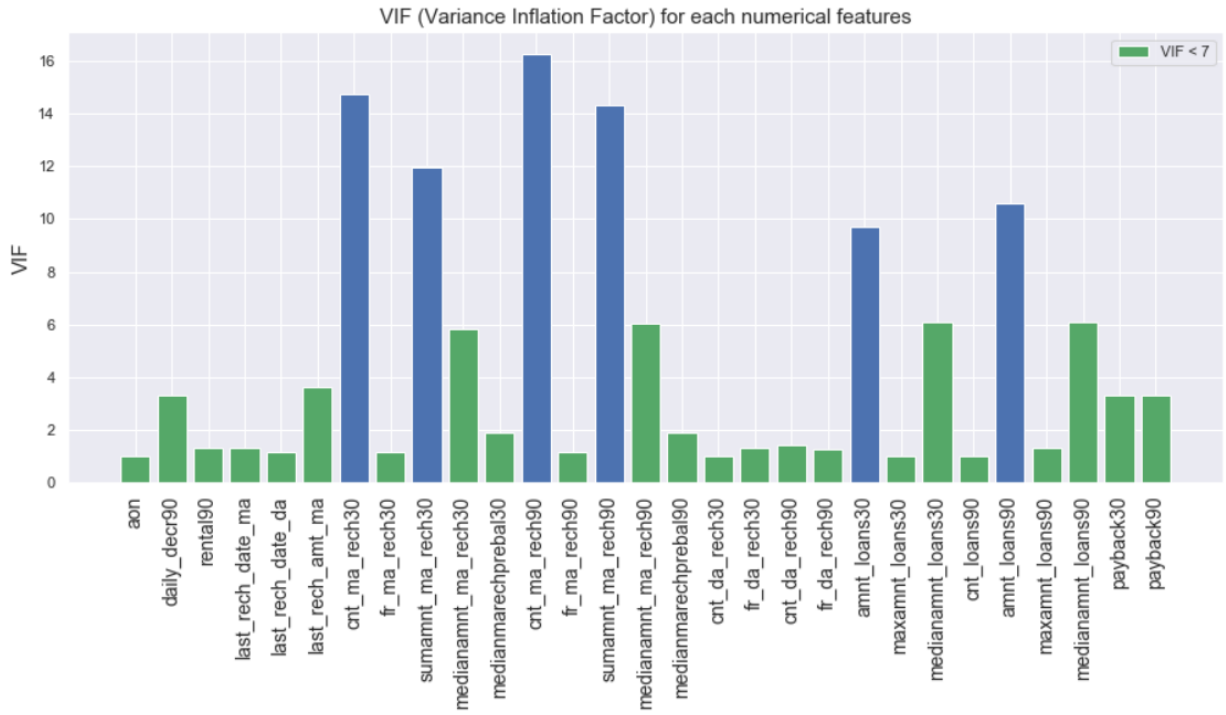
Therefore, the closer the $\mathbf{R^2}$ value to 1, the higher the value of **VIF** and the higher the multi-collinearity with the particular independent variable with respect to all other variables.



Correlation Heatmap

- From above feature correlation heatmap, we have directly dropped highly correlated features like **'daily_decr30', 'rental30' and 'cnt_loans30'**, which have shown correlation of **over 0.95** with its adjacent features.

- We have then calculated VIF using **sklearn** class **Linear Regression** and plotted the VIF values for each predictor. Also, we have set a VIF threshold of 7, i.e., features having VIF score greater than or equal to 7 is considered highly correlated and can be dropped from the dataset.



VIF score plot

- From above plot we can see that features like **'cnt_ma_reach30', 'sumamnt_ma_reach30', 'cnt_ma_reach90', 'amnt_loans30' and 'amnt_loans90'** are having VIF values greater than 7, so therefore we have dropped them from the dataset.

Feature Importance Using scikit-learn module (by SelectKBest and mutual_info_classif)

SelectKBest is a **class** that scores the features using a function (in this case **mutual_info_classif**, but could be others) and then removes all but the **k** highest scoring features.

Here **mutual_info_classif** is a scoring function that can be passed as a parameter to SelectKBest. This function implements important concept from probability and Information theory known as mutual information (MI).

The **mutual information (MI)** is a measure of the amount of information that one random variable has about another variable. This definition is useful within the context of feature selection because it gives a way to quantify the relevance of a feature subset with respect to the output variable. Mathematically, the **MI** is defined as:

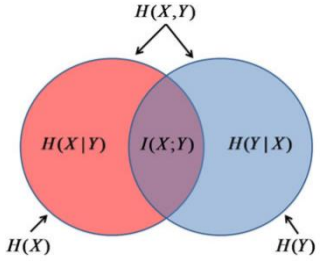
$$I(x;y) = \sum_{i=1}^{n}\sum_{j=1}^{n} p(x(i),y(j)) \cdot \log\left(\frac{p(x(i),y(j))}{p(x(i)) \cdot p(y(j))}\right)$$

Here, ***p(x(i), y(j))*** and ***p(x(i)), p(y(j))*** are joint and marginal probability distributions of random variables respectively.

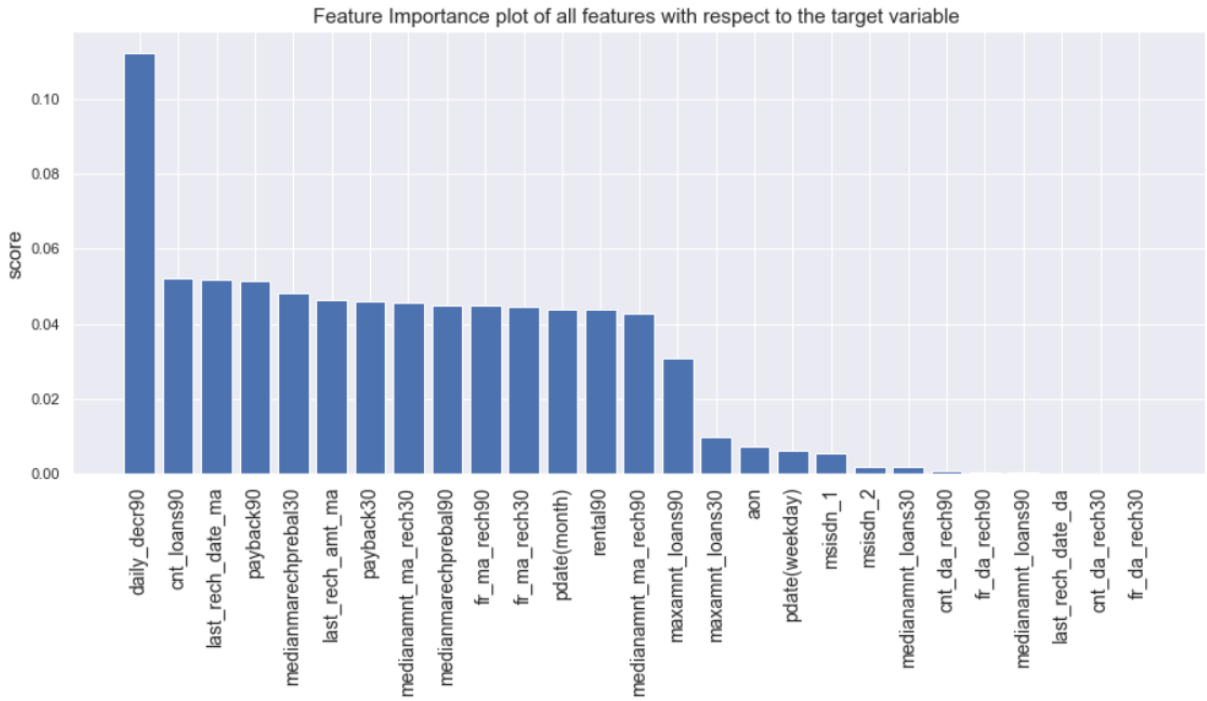
Note that the above equation is valid for probability mass functions (**PMFs**) of two discrete random variables **x** and **y**. For probability density functions (**PDFs**) of continuous variables, we just have to replace double summation with double integral.

Also, **MI** is zero when two random variables **x** and **y** are statistically independent, i.e., ***p(x(i), y(j)) = p(x(i)).p(y(j)).*** The **MI** is related linearly to **entropies** (i.e., measure of randomness) of the variables through the following equations:

$$I(x;y) = \begin{cases} H(x) - H(x|y) \\ H(y) - H(y|x) \\ H(x) + H(y) - H(x,y) \end{cases}$$



Venn diagram showing relationships between MI and entropies



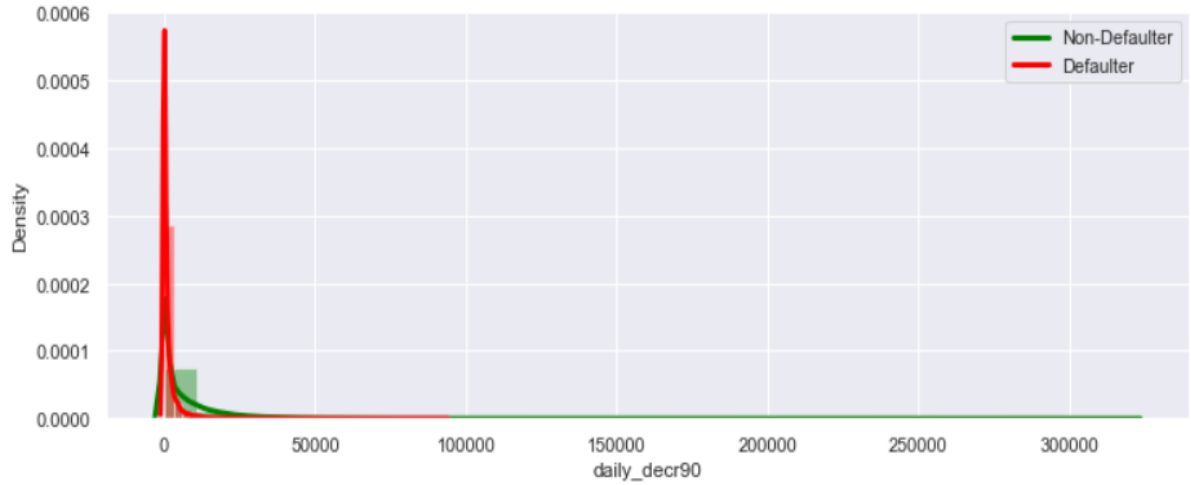
Feature Importance Plot

- From above feature importance plot is obtained using **SelecKBest** and **mutual_info_classif** under **scikit-learn** library. As mentioned earlier, the scoring of these features is based on mutual information **(MI).**

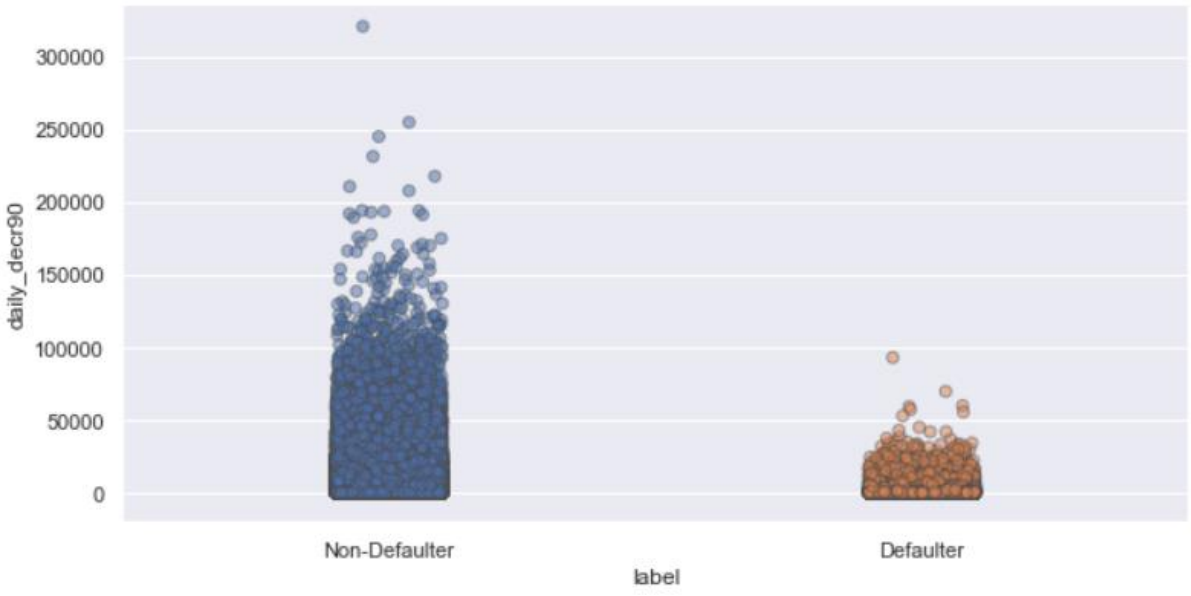
- Here, we have discarded features having relatively lower feature importance, therefore, according to above plot, we have discarded the last 12 features.

- Also, we see that, feature daily_decr90 (i.e., Daily amount spent from main account, averaged over last 90 days) is most important according to **MI**, where it's importance score is almost twice than that of other features.

- Therefore, to further analyse the distribution of this feature (i.e. daily_decr90) with respect to target variable, we plotted the following plots:



Distribution plot of daily_decr90

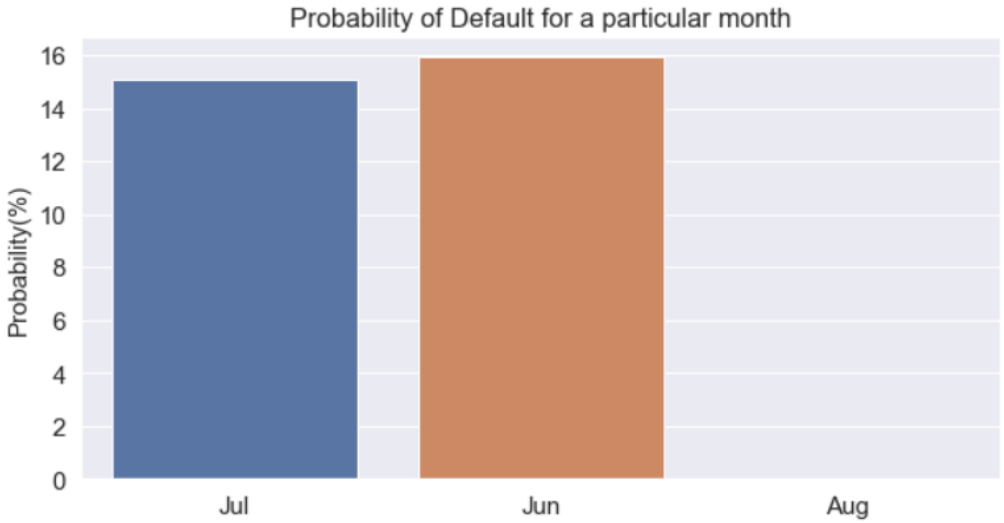


1-D Scatterplot of daily_decr90

- According to above plots of daily_decr90, we can conclude that, majority of customers that are defaulters are within the range of 50,000 Indonesian rupiah, in other words, defaulters usually tend to spend less from their main account.

Target variable distribution

- From above plot, we can see that our target variable is highly imbalanced, with '**Non-Defaulter'** class being almost 7 times more than the '**Defaulter**' class. Therefore, to treat this data imbalance problem, we have applied oversampling technique known as **Synthetic** Minority **Oversampling** Technique (**SMOTE**).

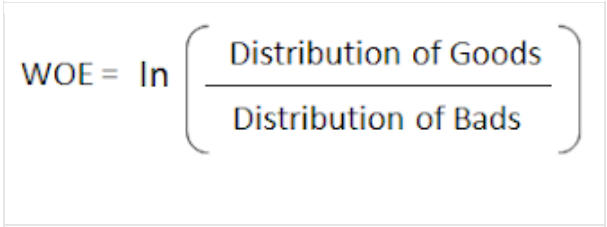


- From above plot, we see that, for July and June month, the probability difference is not significant to come to a conclusion, however, during August month it is quite surprising to see no default cases reported. This could either be due to no availability of micro-credit services during this time or no record of the default case.
- Now, having done data analyses and data cleaning, we have split the dataset into train-test split, where the train data is used for training the model and test data for testing it.

- Then, we have used a python library known as **OptBinning**, that implements **supervised binning** algorithm to transform a continuous variable into bins, that can retain maximum information about the target variable. Also, these bins are then transformed into its **Weight of Evidence (WOE)**.

Weight of Evidence (WOE)

The weight of evidence tells the predictive power of an independent variable in relation to the dependent variable. Since it evolved from credit scoring world, it is generally described as a measure of the separation of good and bad customers. And in our case, **"Bad Customers"** refers to the customers who defaulted on a loan and **"Good Customers"** refers to the customers who paid back loan.



WOE Calculation

Distribution of Goods - % of Good Customers in a particular group **Distribution of Bads -** % of Bad Customers in a particular group **ln -** Natural Log

Usage of WOE

Weight of Evidence (WOE) helps to transform a continuous independent variable into a set of groups or bins based on similarity of dependent variable distribution.

For continuous independent variables: First, create bins (categories / groups) for a continuous independent variable and then combine categories with similar WOE values and replace categories with WOE values. We use WOE values rather than input values in our model.

- As we have filtered our categorical variable during feature to only one (i.e.- **'pdate(month)**'), and as this variable can be treated as nominal variable, therefore, we have then dummy encoded it use **pandas** '**get_dummies ()'** method**.**

- And then finally, we have applied **SMOTE** for treating data imbalance problem.

Application of Synthetic Minority Over-sampling Technique (SMOTE) for imbalanced dataset

As we have noted earlier, that our dataset is highly imbalanced and imbalanced classification involves developing predictive models on classification datasets that have a severe class imbalance. The challenge of working with imbalanced datasets is that most machine learning techniques will ignore, and in turn have poor learning performance on the minority class, and in our case, since the minority class is 'Defaulter', therefore, it is very crucial to properly capture its importance.

And this can be done by simply increasing the examples of minor class, which can be done by the simplest approach involving duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short.

In python, under **imbalanced-learn** API, we have used over-sampling methods like **SMOTE-NC** for this task. Here, note that **NC** stands for nominal and continuous, as this sampling algorithm can treats nominal and categorical variables separately.

Hardware and Software Tools Used

Hardware used:

Processor: intel i7-9750 H

System type: 64-bit operating system, x64-based processor

Installed RAM: 16 GB

GPU: GeForce GTX 1650 (supports CUDA libraries)

Software and tools used:

Operating Software: Windows 10

Programming Language: Python

IDEs/Framework: Jupyter notebook using Anaconda Framework

Libraries and Packages used:

NumPy: Contains a multi-dimensional array and matrix data structures. It is utilised to perform a number of mathematical operations on arrays.

Pandas: It contains inbuilt Data-structures like Data-Frames and Series, which is used for data manipulation and analysis.

Matplotlib and Seaborn: Used as a data visualization tool.

Scikit-learn: It is used in implementation of various machine learning algorithms and it also contains various classes and functions for data pre-processing metrics for model evaluation.

OptBinning: It is a python library for implementing supervised binning.

Imbalanced-learn: This python library is used for implementing sampling algorithms to imbalanced dataset.

LightGBM: It is a gradient boosting framework that uses tree-based learning algorithms. In our case, we used one of the class called **LGBMClassifier** for our classification problem.

3. Model Development and Evaluation

3.1 Problem-solving approaches

- The first step is to identify the irregularities in the dataset and although in the dataset description, there was no mention of null values, but the dataset had non-realistic values for some features, to which it was handled accordingly, as mentioned earlier.

- To look out for features that do not add any value to the analysis and modelling, for example, features having only one unique value or all unique values, so that we can discard it before proceeding.

- To look for features, particularly in categorical features where additional information may be encoded, for example, a date column may have information regarding year, month, weekdays, etc., which can be further extracted to form useful features.

- To separately analyse categorical and numerical features using appropriate visualization techniques for each.

- Tried to identify possible outliers using boxplots and only remove them if we use linear models that are sensitive to outliers, or if the outliers have no relevance according to the domain knowledge. But in our case, since large percentage of data consisted of outliers, and it was not wise to remove all, therefore, it was better to rather bin those numerical features.

- Check data skewness of numerical features, and if skewness is within acceptable range then use transformation techniques to reduce data skewness.

- Multivariate analysis with pair-plots or heatmaps of continuous/numerical variables. And look for the most correlated features, and if the correlation is above 0.95, then we can drop any one of the correlated features.

- And if there are multiple correlations between variables, then it is quite difficult to remove the top correlated variables. So, in this case, we can

use Variance Inflation Factor (VIF) score to identify the most redundant variables. Also, identification and removal of these variables is very crucial for interpretability of model.

- If the feature space is fairly medium to large, then before further analyses of each and every feature in the dataset, it is wise to understand the relationship between input and output variables, and to find out the relevance of an input feature with the output, therefore, we can apply feature selection methods that incorporates various statistical and analytical techniques, like in this project we have applied mutual information(MI) between input and output variable to find out the importance of an input variable with respect to the output variable.

- Then we can further analyse only those variables that are very important with respect to output variable.

- After complete analysis of the data, we can then split the dataset into training and testing set, and then we can have dummy encoded nominal variables. Also, in order to prevent data leakage, it is very important to apply any supervised data transformation techniques (in our case, supervised binning) after the train-test split using training set and then to transform both training and testing set using statistics of training set.

- Finally, before modelling the data, we can apply oversampling techniques like SMOTE to balance the class in our target variable.

Algorithm used for Data Modelling

As mentioned earlier, we have used supervised learning algorithms that consists of simple algorithms like:

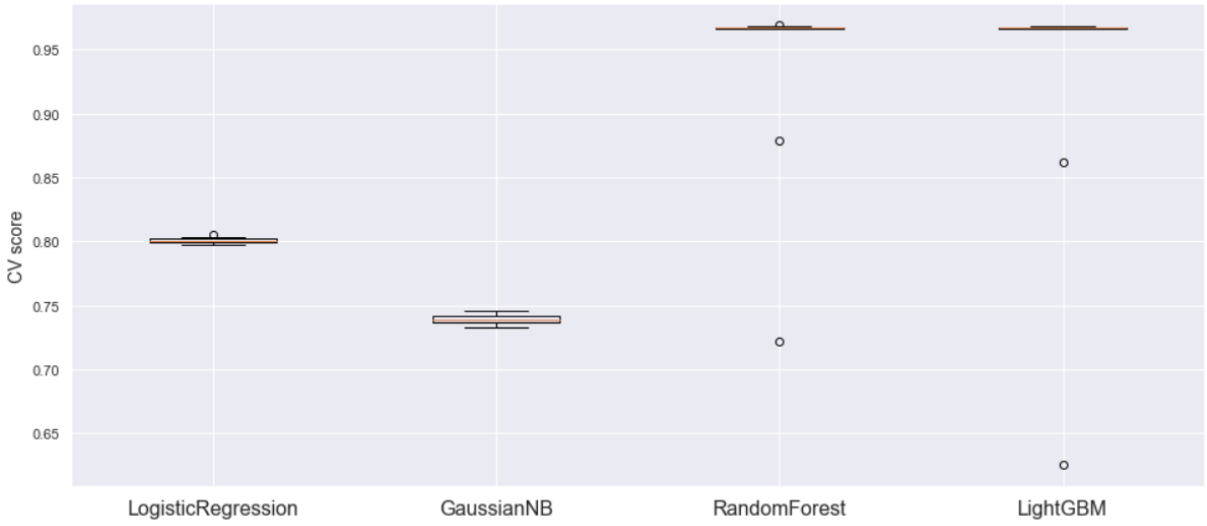
- Logistic regression
- Gaussian Naive Bayes

And also, relatively complex tree-based algorithms like:

- Random Forest
- Light Gradient Boosting Machine (LightGBM)

3.2 Base Model Evaluation

Base model evaluation was done by cross validation on training set using k-fold = 10, that implies, 90% of data was used in training and 10% on testing, which was iteratively done on each of k^{th} fold, and the final score being the average test score of all these folds. Also, we have used scoring function as f1 score, which is better metrics than accuracy, particularly in case of data imbalance.



Base Model Comparison

From above boxplots, we see that, Random Forest and LightGBM performed better overall than the simple models like Logistic Regression and Gaussian Naive Bayes. But for two instances, we see Random Forest and LightGBM performed even poorer than the other two models, which can be noted by the outlier points. This could probably due to lack of regularization, which can we further tuned by hyper-parameter tuning.

3.3 Train and Test set results

Metric	**Logistic Regression**	**Gaussian Naive Byes**	**Random Forest**	**LightGBM**
Accuracy %	79.6	75.2	98	93.7
F1(macro-average) %	79.6	75.2	98	93.7
Recall %	82	70.3	98.7	92.5

Train Results

Metric	Logistic Regression	Gaussian Naive Byes	Random Forest	LightGBM
Accuracy %	77.8	78.8	89.4	89.7
F1(macro-average) %	66.9	66.2	74.4	75.6
Recall %	82	71	51.5	54.8
ROC AUC %	88.2	82.8	87.9	89.5

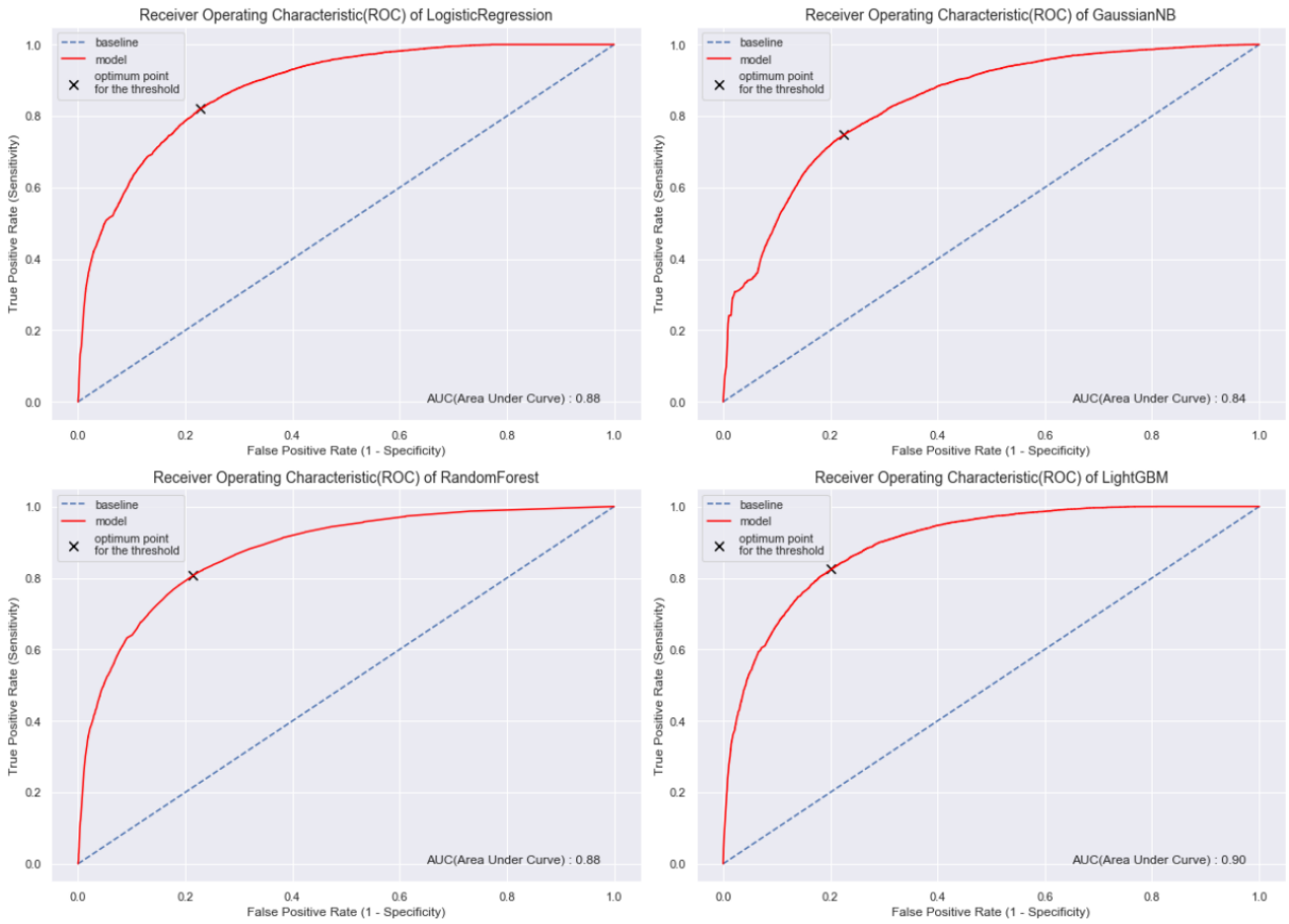
Test results

By comparing train and test results, as expected, tree-based complex models like Random Forest and LightGBM overfitted more than that of other two models. Particularly these complex models have shown low recall score on test results.

3.4 Hyper-Parameter Tuning and Threshold Adjustment

To optimize the overall performance, we have hyper-parameter tuned each model. Also, keeping in mind about data imbalance in test set, we then with the help of Receiver Operating Characteristic (**ROC**) curve of a model, have adjusted the probability threshold of each model to get optimal performance. Therefore, these threshold values have been shifted to the optimal value corresponding to the highest geometric mean or **G-mean** between **sensitivity** and **specificity.**

The main reason for a threshold adjustment is to keep balance between **sensitivity** and **specificity**, that increases model's ability to correctly classify the **positive class** (i.e., Defaulter) without much compromising about **False positive** (i.e., to classify Non-Defaulter to Defaulter).

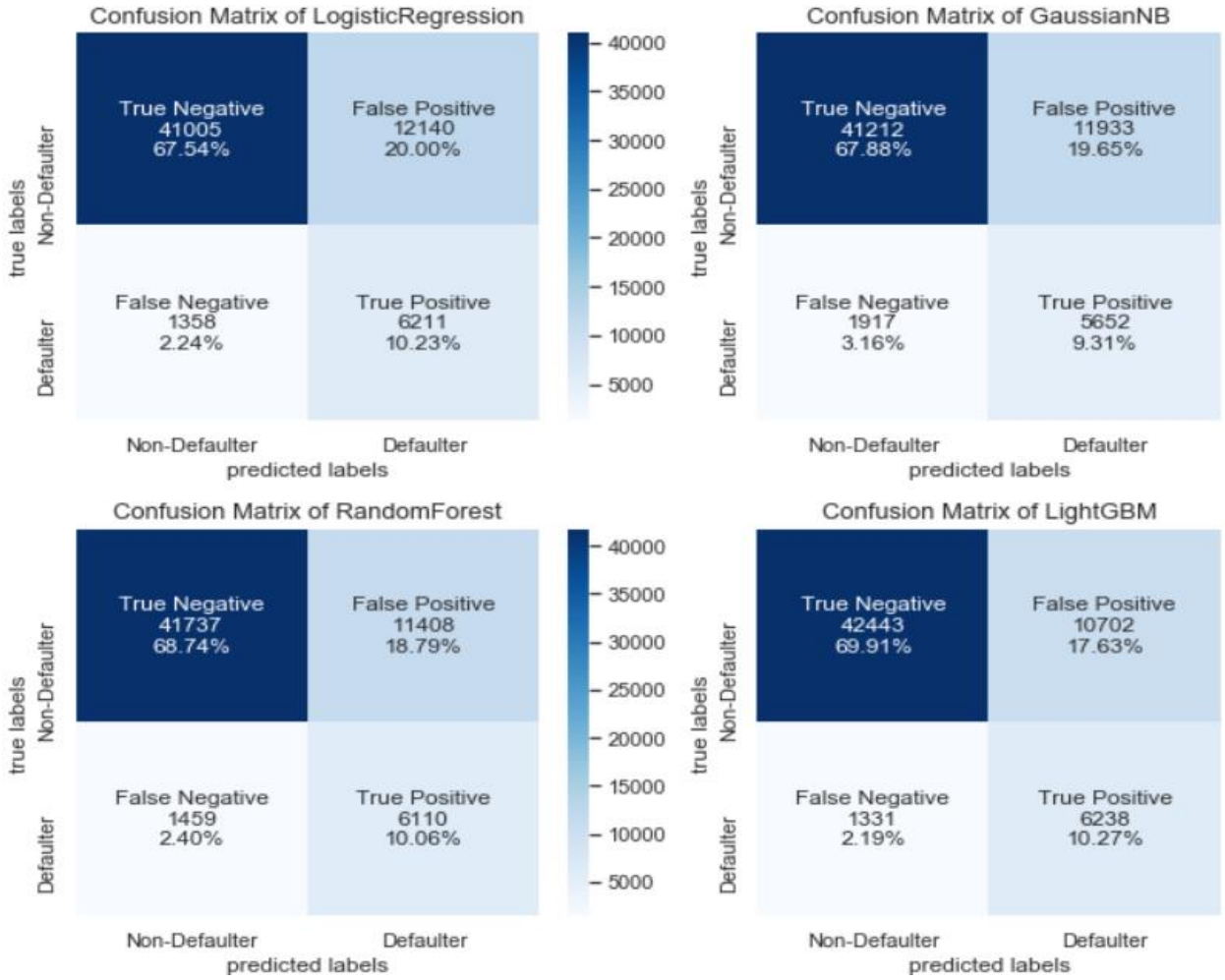


ROC curves of all models with their optimum threshold point after hyper-parameter tuning

Metric	Logistic Regression	Gaussian Naive Byes	Random Forest	LightGBM
Accuracy %	77.8	77.2	78.8	80.2
F1(macro-average) %	66.9	65.3	67.9	69.2
Recall %	82	74.7	80.7	82.4
ROC AUC %	88	84	88	90

Test Results after hyper-parameter tuning and threshold adjustment

Visualizing Confusion Matrix of Final Test Results with the help of Heatmaps.



3.5 Understanding Key metrics for success and the interpretation of the final test results

From a business perspective, in order to prevent the further revenue loss by lending loans to defaulters, it is very important to correctly identify the customers who are 'Defaulter' and to reduce **false negatives**, therefore, our model should have high **recall** as possible, given by:

Recall = $\frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ Positives + False\ Nagatives}}$

Here**,**

True Positives = Number of correctly identified customers how are 'Defaulter'.

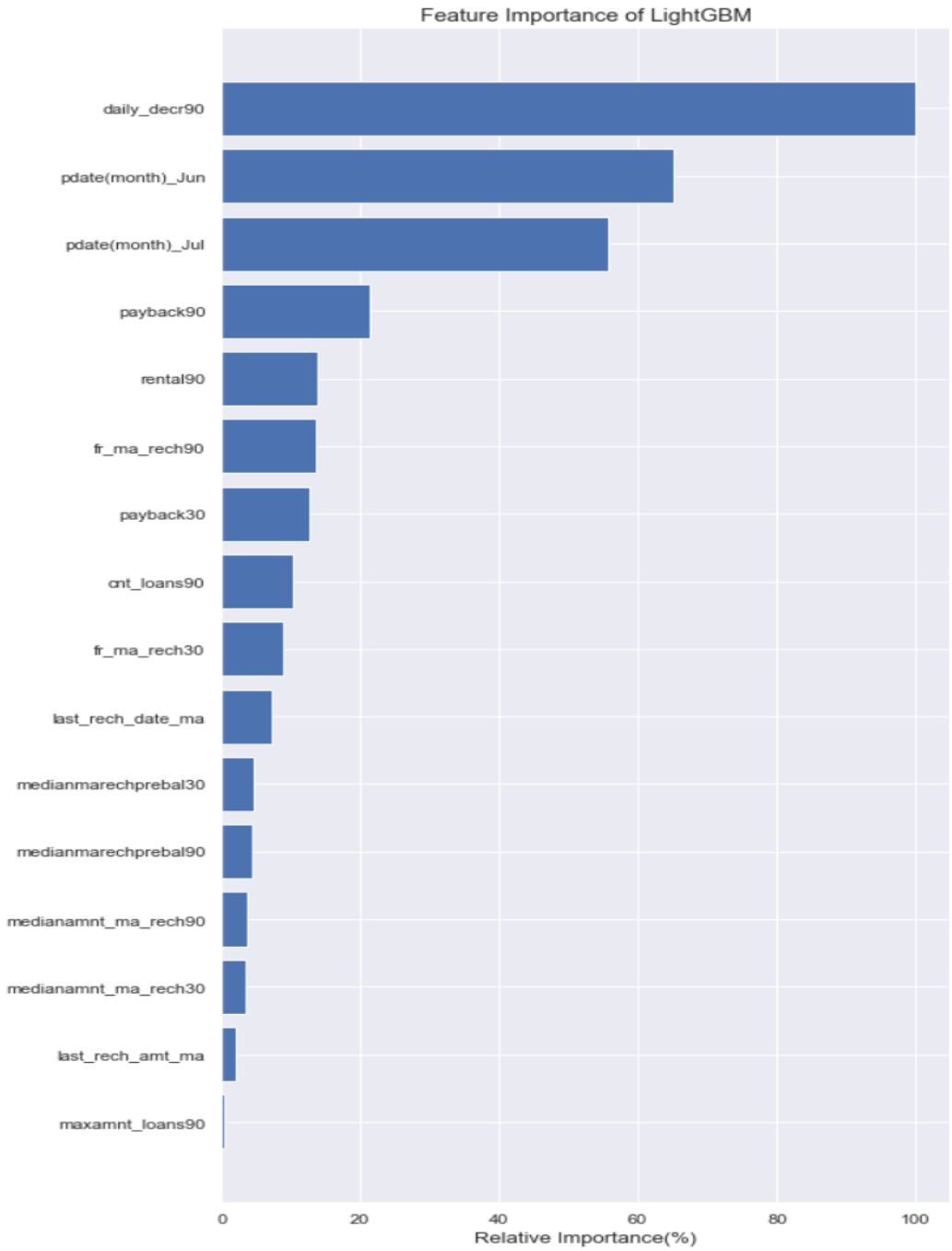
False Negatives = Number of customers how are 'Defaulter', but falsely classified as 'Non-Defaulter', also known as **Type-II error.**

However, we must also take care to not have very high ***False Positives*** (also known as **Type-I error**), i.e., to falsely classify 'Non-Defaulter' as 'Defaulter'. Therefore, we have considered metric like **F1 score** which also takes number of ***False Positives*** into account.

Also, considering imbalanced data-set we have used metrics like **F1 macro-average**, that independently calculates the F1 score for **each label** and then finds out the arithmetic **mean** between these F1 scores

Although, the accuracy of the all models decreased after hyper-parameter tuning and probability threshold adjustments, but considering heavy data imbalance, we must always decide the final model on the bases of other metrics, like in our case, high recall, f1 score and ROC AUC is much favourable.

Therefore, it can be clearly seen from above test results, that the **LightGBM** model performed the best out of all the models with respect to all metrics, also by analysing the above heatmaps, we see that LightGBM has the lowest **Type-I** and **Type-II** errors as well. Also, LightGBM, being highly efficient for medium to large datasets, it is therefore favourable for this project.



Feature Importance of LightGBM

The above figure is the Feature Importance plot of the LightGBM model, which is calculated with the help of **scikit-learn** class **permutation importance.**

Permutation feature importance is a model inspection technique that can be used for any fitted model. This is especially useful for non-linear or

opaque models. The **permutation_importance** function calculates the feature importance of the estimators/models for a given dataset. The **n_repeats** parameter of the function sets the number of times a feature is randomly shuffled and returns a sample of mean feature importance.

4. CONCLUSION

- It was observed that, through statistical analysis and feature importance calculations, the feature **daily_decr90** (i.e., Daily amount spent from main account, averaged over last 90 days) has shown to be most valuable feature for prediction of the customer type (i.e., Non-Defaulter or Defaulter), also from the previous figures of **daily_decr90**, it is evident that the most of the customers who were defaulters, were spending relatively less from the main account.

- Also, during August month, there were interestingly no defaulter cases reported. This could either be due to no availability of micro-credit services during this time or no record of the default case.

- The various visualizations tools and software helped to understand the dependencies of variables with respect to each other and the target variable (i.e., 'Non-Defaulter' and 'Defaulter') and also about customer segmentation with respect to the target variable for the most important feature like **daily_decr90**.

- The main criteria for selecting the algorithms used for this project was to try mix of different kinds of algorithms, that are simple such as Logistic Regression and Gaussian Naive Bayes and are also relatively complex algorithms like Random Forest and LightGBM. Also, it was found that, even though the data were highly skewed and contained several outliers, but after supervised binning these numerical variables, considerably increased the performance of simple algorithms like Logistic Regression, that almost performed comparable to complex algorithm like Random Forest.

- Just like for any data science project, before modelling and prediction part, the most important and time-consuming part was the data analysis, cleaning and pre-processing with appropriate assumptions and methods, which requires the both, respective domain knowledge and efficiency in data manipulation.

- As the dataset contained irregularities that was dealt with some assumptions, and also the main assumptions that was made is that the sample data would represent the entire population, which may not necessarily be true. Therefore, it is recommended to use similar kinds of different datasets from other sources (possibly from other telecom service providers) and combining them to start all over again. This could therefore reduce the bias in results for a particular population and can possibly give much better and more concrete scenario.