FLIP ROBO

A Project Report on MICRO CREDIT DEFAULTER

Submitted by:

Satu Vinayak Pole

ACKNOWLEDGMENT

I would like to express my special gratitude to my SME Mr. Shwetank Mishra as well as “Flip

Robo” team for letting me work for a project named “Flight Price Prediction” . Also thanks to my institute ‘Data Trained’ .

Also I would like to thank websites such as StackOverflow, geeksforgeeks and Youtube who

has helped me in solving issues and errors.

[1] INTRODUCTION

{1.1} Business Problem Framing

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.

{1.2} Conceptual Background of the Domain Problem

This problem is regarding 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.

They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed 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).

{1.3} Review of Literature

Literature review covers relevant literature with the aim of gaining insight into the factors that cause loans default within micro finance institutions. The main aim of micro finance isto provide fundsfor investment in micro businesses that is expected to increase income to investor households and hence improve their livelihood. It has been observed that most borrowers use micro credit finances on food, shelter and clothing to meet their basic needs rather than investment. In order to overcome challenges of loan defaults, micro finance institutions use various credit lending models. One of the modelsin micro finance isrotating savings and credit associations(ROSCA). ROSCAs form groups of individuals who pay into an account on a monthly basis. Each individual then earns an opportunity to receive a relatively large loan with to invest. The group decides who receives the loan each term, often based on rotating schedule. The initial

money is either accumulation of the group members' individual deposits or more frequently, by an outside donation. Loan repayment is ensured through peer pressure. Anyone who does not repay the loan amount risks the privilege to borrow in the future.

{1.4} Motivation for the Problem Undertaken

The main objective of this study is to investigate which method from a chosen set of machine learning techniques performsthe best default prediction. This project was highly motivated project as it includes the real time problem for Microfinance Institution (MFI), and to the poor families in remote areas with low income, and it is related to financial sectors, as I believe that with growing technologies and Idea can make a difference, there are so much in the financial market to explore and analyse and with Data Science the financial world becomes more interesting.

[2] Analytical Problem Framing

{2.1} Mathematical/ Analytical Modeling of the Problem

I had to build a machine learning model based on the dataset provided to me. The dataset was loaded and checked for null values and duplicates of there any. This was further treated using necessary steps. Further some of the columns contained >90% zero values, hence they were dropped. Also some unwanted columns were dropped that would have played no role in model building. Further all the pre-processing steps including visualization was performed on dataset. This preprocessing steps also included using Variance Inflation factor(VIF) and Principal Component Analysis (PCA) to take of multicollinerity problem and select best columns for model making. A total of 6 machine learning algorithm were used to perform machine learning model. The best model was further tuned using GridSearch CV. The tuned model showed less accuracy, hence the original model was saved using Pickle library

{2.2} Data Sources and their formats

The whole dataset was provided to us as csv file. The dataset contained 209593 rows and 37 columns of which 1 column contains only index numbers hence it was dropped immediately. Most column had either integer or float type but 3 columns had datatype as object.

{2.3} Data Preprocessing Done

The data preprocessing was done using following methods/steps:

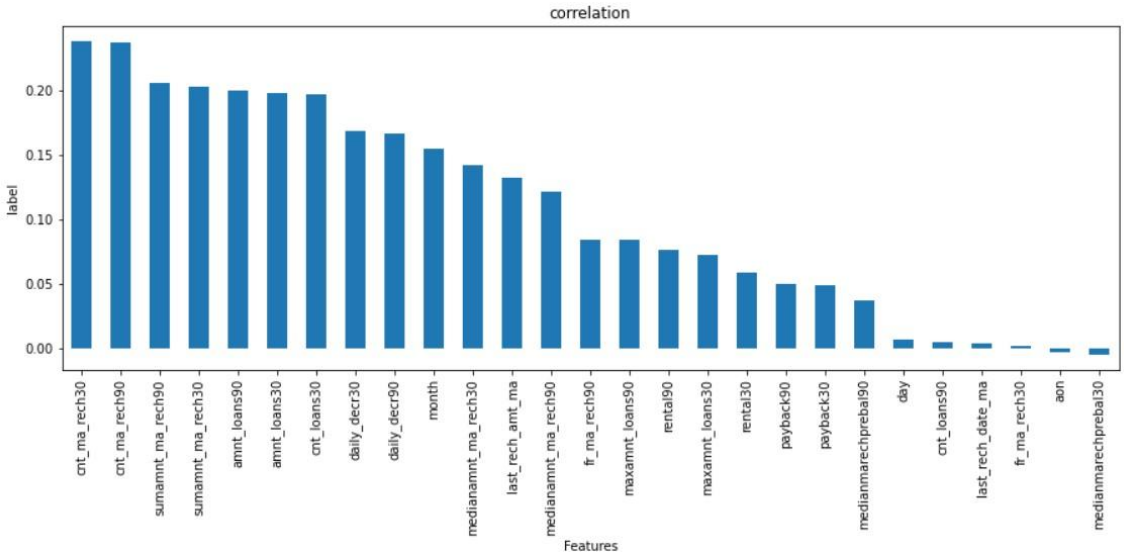
- The dataset provided named 'Data File' was uploaded using pandas library.
- The dataset contained 209593 rows and 37 columns of which it contained 1 non useful column such as 'Unnamed:0' . This column was dropped as this column just contained index numbers corresponding to rows.
- Further the dataset was checked for duplicate entries in dataset and it was found that it contained 1 duplicate entry which was further removed.
- Upon checking dataset for null values, it was confirmed that no null values were present in dataset.
- Further 2 columns i.e. 'pcircle' and 'msisdn' were dropped. The reason being that 'pcircle' which meant telecom circle of all users had only 1 name of telecom circle and 'msisdn' consisted of mobile number of users. Since this kind of singular and random data is not usable for model building, hence they were dropped.
- Next, Since 'pdate' column which contained date cannot be used directly , it was seperated into 3 different columns I.e. day, month and year. They year column now contained data of only 2016 year, hence it was dropped from dataset with date column.
- Further upon seeing dataset, it was seen that many column contained 0 values in them. Upon checking dataset for 0 values it was found that 7 colums named 'last_rech_date_da', 'cnt_da_rech30', 'fr_da_rech30', 'cnt_da_rech90', 'fr_da_rech90', 'medianamnt_loans30', 'medianamnt_loans90' had more than 90% values. These columns were dropped from dataset as model requires data to learn. Since the 0 values in this columns were more, the model would have learned nothing and might have behaved improperly. Hence they were dropped.

- Further after checking value counts and its decription for 'maxamnt_loans30' column, it was noticed that this column should contains data either as 6, 12 or 0. Hence remaining data with values other than mentioned above were replaced with 0.
- Further data was analysed using visualization plots such as countplot, scatterplot, barplots and heatmap.
- Upon describing the dataset, it was noticed that minimum values for approx. 8 columns were negatives, hence these data was made positive using numpy.absolute method.
- Further a heatmap was plotted to check for multicollinearity within features and relationship of features with label. On examining, many columns showed multicollinearity. This was further confirmed using Variance Inflation Factor (VIF) method.
- Further the dataset was checked for skewness and outliers and subsequently they were treated. After treting for outliers the data loss was around 5.6 % which was well within the range of 8%.
- The data was scaled or standardized and data was split into label and features.
- Once the VIF dataframe was printed which showed values of columns that showed multicollinearity, I did not dropped any columns. The only reason being most of the columns showed multicollinearity and dropping all columns might have indicated dropped columns that might be actually required for model building. Hence to take into account this multicollinearity issue I used Principal Component Analysis (PCA) method. Using PCA curve it helped in choosing columns that were enough to describe dataset which means I choose only 20 columns using PCA curve which showed at 20 columns the data covered is almost 100%.
- Further it was found that the target column 'label' was imbalanced with more number of non-defaulters present. Hence this issue was sorted using SMOTE method.
- The data was now ready for model building using only 20 columns.

{2.4} Data Inputs- Logic- Output Relationships

Looking at the target column it was known that this is a kind of classification problem hence use of classification algorithms will be used.

Since all the columns were continuous in nature and data was huge I used barplots to show relation of label with respect to different features. Distplots were used to check for skewness and boxplots were used to identify which columns had outliers present.



- The above figure shows relationship of features with respect to label.

As seen in above figure all the features shows positive corelation with label. Highest relationship with label is shown by 'cnt_ma_rech30' column.

{2.5}Hardware and Software Requirements and Tools Used

Hardware used to complete the analysis and model building:
Model: ASUS TUF A15
Processor: AMD RYZEN 5 4600H OCTA CORE
RAM: 8GB
ROM:500 GB SSD

Software used to complete the analysis and model building is:
Jupyter Notebook (via Anaconda Navigator)

Libraries used:
Seaborn and matplotlib for visualization
Numpy
Pandas
Sklearn
Imblearn & collection used for using SMOTE
Xgboost
Pickle

[3] Model/s Development and Evaluation

{3.1} Identification of possible problem-solving approaches (methods)

I have used both statistical and analytical approaches to solve the problem which mainly includes the pre-processing of the data also used EDA techniques and heat map to check the correlation of label and features. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models. The imbalanced target column was balanced using SMOTE method and final columns were selected using PCA method.

Since the target label was binary in nature, hence it was known that it will be a classifier type problem, hence classification algorithms were used for model building. Accuracy score, confusion matrix, F1 score , AUC score and cross validation score were used to identify best model. Random Forest was chosen as the best model and this was further tuned using GridSearchCV. Though the tuned model was not having good accuracy hence the original Random Forest model was saved.

{3.2} Testing of Identified Approaches (Algorithms)

Since this was a classification kind of problem, I have used classifier algorithms to build machine learning models. After balancing the dataset I have used XGB classifier algoritm to find best random state under which the model has highest training and testing accuracy. The random state chosen was 13 for all subsequent algorithms. 6 algorithms were used to build ML model which are as follows:

1. Random Forest Classifier (rf)
2. Xtreme Gradient Boosting (xgb)
3. AdaBoost Classifier (ab)
4. Logistic Regression (lr)
5. Gradient Boosting Tees Classifier (gbdt)
6. Decision Tree Classifier(dt)

{3.3} Run and Evaluate selected models

The above mentioned algorithms were imported using their respective libraries and saved in variables mentioned in below snapshot.

```
rf=RandomForestClassifier()
ab=AdaBoostClassifier()
xgb=xgb.XGBClassifier()
lr=LogisticRegression()
```

```
dt=DecisionTreeClassifier()
gbdt=GradientBoostingClassifier()
```

1 . Xtreame Gradient Boosting (xgb)

```
xgb.fit(x_train,y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_threshold=64, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
#passing the function
metric_score(xgb,x_train,x_test,y_train,y_test,train=True) # this is for training score

metric_score(xgb,x_train,x_test,y_train,y_test,train=False) # this is for testing score
```

```
-----Training score-----
Accuracy score:89.10%
-----Testing score-----
Accuracy score:87.51%
classification report
              precision    recall  f1-score   support

           0       0.87      0.89      0.88     43179
           1       0.88      0.86      0.87     43554

    accuracy                           0.88     86733
   macro avg       0.88      0.88      0.88     86733
weighted avg       0.88      0.88      0.88     86733

confusion matrix  [[38281  4898]
 [ 5936 37618]]
F1 Score 0.8741257116300686
```

```
#cross validation score
print('Cross Validation Score for XGB model :- ',((cross_val_score(xgb,x_new,y_new,cv=4).mean())*100))
```

```
Cross Validation Score for XGB model :-  87.6145757670091
```

The xgb model provided testing accuracy of 87.51% with cross validation and F1 score as 87.61 and 87.41 respectively.

2 . Random Forest Model

Random Forests Classifier

```
#training model
rf.fit(x_train,y_train)
```

```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
# passing the function
metric_score(rf,x_train,x_test,y_train,y_test,train=True) # this is for training score

metric_score(rf,x_train,x_test,y_train,y_test,train=False) # this is for testing score
```

```
-----Training score-----
Accuracy score:100.00%
-----Testing score-----
Accuracy score:93.73%
classification report
              precision    recall  f1-score   support

           0       0.92      0.95      0.94     43179
           1       0.95      0.92      0.94     43554

    accuracy                           0.94     86733
   macro avg       0.94      0.94      0.94     86733
weighted avg       0.94      0.94      0.94     86733

confusion matrix  [[41094  2085]
 [ 3350 40204]]
F1 Score 0.9366867420756497
```

```
#cross validation score
print('Cross Validation Score for Random Forest model :- ',((cross_val_score(rf,x_new,y_new,cv=4).mean())*100))
```

```
Cross Validation Score for Random Forest model :-  93.9103340135819
```

The Random Forest model provided testing accuracy of 93.73% with cross validation and F1 score as 93.91 and 93.66 respectively.

3 . AdaBosst Model

AdaBoost Classifier

```
ab.fit(x_train,y_train)
```

```
AdaBoostClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
# passing the function
metric_score(ab,x_train,x_test,y_train,y_test,train=True) # this is for training score

metric_score(ab,x_train,x_test,y_train,y_test,train=False) # this is for testing score
```

```
-----Training score-----
Accuracy score:79.11%
-----Testing score-----
Accuracy score:78.79%
classification report
              precision    recall  f1-score   support

           0       0.78      0.79      0.79     43179
           1       0.79      0.78      0.79     43554

    accuracy                           0.79     86733
   macro avg       0.79      0.79      0.79     86733
weighted avg       0.79      0.79      0.79     86733

confusion matrix  [[34285  8894]
 [ 9500 34054]]
F1 Score 0.7873575177452545
```

```
#cross validation score
print('Cross Validation Score for AdaBoost model :- ',((cross_val_score(ab,x_new,y_new,cv=4).mean())*100))
```

```
Cross Validation Score for AdaBoost model :-  79.08408564214312
```

The Adaboost model provided testing accuracy of 78.79% with cross validation and F1 score as 79.08 and 78.73 respectively.

4 . Logistic Regression

Logistic Regression

```
lr.fit(x_train,y_train)
```

```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
# passing the function
metric_score(lr,x_train,x_test,y_train,y_test,train=True) # this is for training score

metric_score(lr,x_train,x_test,y_train,y_test,train=False) # this is for testing score
```

```
-----Training score-----
Accuracy score:77.04%
-----Testing score-----
Accuracy score:76.93%
classification report
              precision    recall  f1-score   support

           0       0.76      0.78      0.77     43179
           1       0.78      0.75      0.77     43554

    accuracy                           0.77     86733
   macro avg       0.77      0.77      0.77     86733
weighted avg       0.77      0.77      0.77     86733

confusion matrix  [[33857  9322]
 [10689 32865]]
F1 Score 0.7666110728822849
```

```
#cross validation score
print('Cross Validation Score for Logistic Regression model :- ',((cross_val_score(r,x_new,y_new,cv=4).mean())*100))
```

```
Cross Validation Score for Logistic Regression model :-  77.02604544982879
```

5 . Decision Tree Model

Decision Tree Classifier

```
dt.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
# passing the function
metric_score(dt,x_train,x_test,y_train,y_test,train=True) # this is for training score

metric_score(dt,x_train,x_test,y_train,y_test,train=False) # this is for testing score
```

```
-----Training score-----
Accuracy score:100.00%
-----Testing score-----
Accuracy score:87.47%
classification report
              precision    recall  f1-score   support

           0       0.86      0.90      0.88     43179
           1       0.89      0.85      0.87     43554

    accuracy                           0.87     86733
   macro avg       0.88      0.87      0.87     86733
weighted avg       0.88      0.87      0.87     86733

confusion matrix  [[38647  4532]
 [ 6339 37215]]
F1 Score 0.8725571798689349
```

```
#cross validation score
print('Cross Validation Score for Decision Tree model :- ',((cross_val_score(dt,x_new,y_new,cv=4).mean())*100))
```

```
Cross Validation Score for Decision Tree model :-  87.83046821855581
```

The decision tree model provided testing accuracy of 87.47% with cross validation and F1 score as 87.83 and 87.25 respectively.

6 . Gradient Boosted Trees Model

Gradient Boosting Classifier

```
gbdt.fit(x_train,y_train)
```

```
GradientBoostingClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

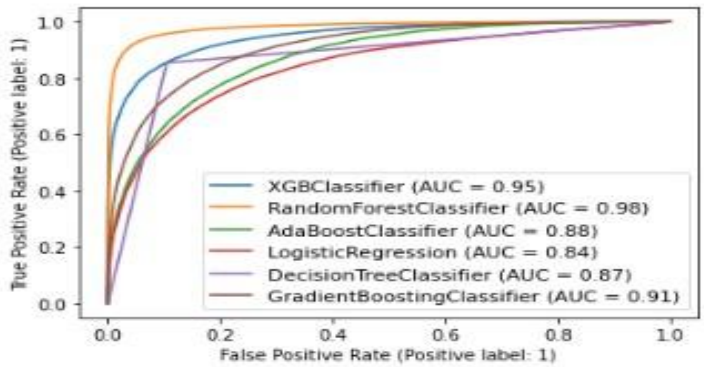
```
# passing the function
metric_score(gbdt,x_train,x_test,y_train,y_test,train=True) # this is for training score

metric_score(gbdt,x_train,x_test,y_train,y_test,train=False) # this is for testing score
```

```
-----Training score-----
Accuracy score:82.74%
-----Testing score-----
Accuracy score:82.40%
classification report
              precision    recall  f1-score   support

           0       0.83      0.81      0.82     43179
           1       0.82      0.83      0.83     43554

    accuracy                           0.82     86733
   macro avg       0.82      0.82      0.82     86733
weighted avg       0.82      0.82      0.82     86733

confusion matrix  [[35115  8064]
 [ 7197 36357]]
F1 Score 0.826530264279625
```

```
#cross validation score
print('Cross Validation Score for Gradient Boosting model :- ',((cross_val_score(gbdt,x_new,y_new,cv=4).mean())*100))
```

```
Cross Validation Score for Gradient Boosting model :-  82.49311104193329
```

The gradient boosted tees model provided testing accuracy of 82.40% with cross validation and F1 score as 82.49 and 82.65 respectively.



The above figure shows the ROC curve with AUC score for all trained models.

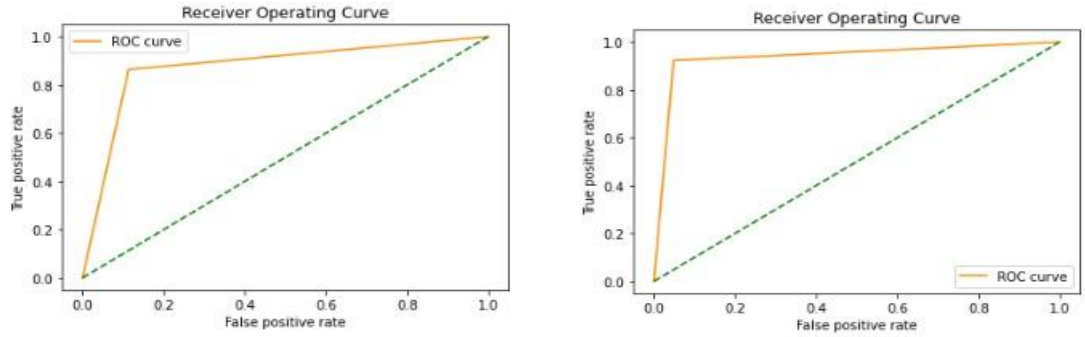
Looking at above curves it can be seen that Random Forest Classifier model covers most of the area. Hence according to this Random Forest model is the best model.

	Model	Training scores	Testing scores	Cross validation score	F1 score	AUC score
0	XGB	89.07	87.50	87.55	87.40	95
1	Random Forests	100.00	93.73	93.91	93.66	98
2	AdaBoost	79.11	78.79	79.08	78.73	88
3	Logistic regression	77.04	76.93	77.02	76.66	84
4	Decision Trees	100.00	87.47	87.83	87.25	87
5	Gradient Boosted Tees	82.74	82.40	82.49	82.65	91

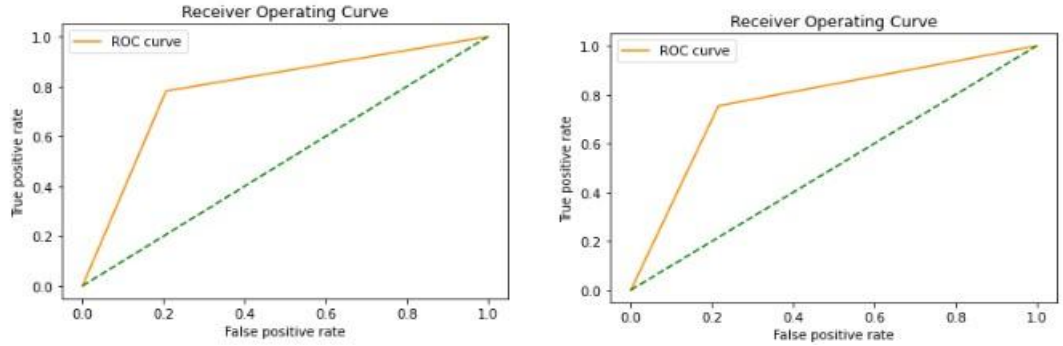
The above table shows the training, testing cross validation scores. F1 scores and AUC score as the basis on which the best model was selected.

Based on above table Random Forest model was selected as the best model as it showed highest testing accuracy with better cross validation, F1 score and AUC score with respect to other models. The hyperparameter tuning was then performed on this model.

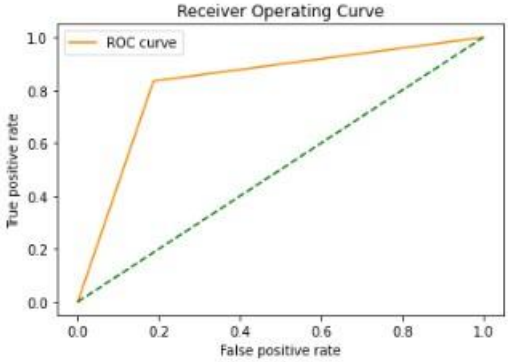
ROC curve for all models



Above plots shows ROC curve for xgb (left) and Random Forest model (right)



Above plots shows ROC curve for adaboost (left) and logistic regression model (right)



Above plot shows ROC curve for Gradient Boosted Tees model (right)

Hyperparameter tuning of Random Forest Model

The hyperparameter tuning was performed using GridSearchCV. The parameters were fed into grid model and best parameters were find out. This parameters were then used on Random Forest model as shown below and were trained. The accuracy score and other metrics are seen below.

```
#using best parameters to train
rf1=RandomForestClassifier(max_depth=3,
                           criterion='gini',
                           min_samples_split=2,
                           max_features='log2')
```

```
rf1.fit(x_train,y_train)
```

```
RandomForestClassifier(max_depth=3, max_features='log2')
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.
```

```
#passing the function
metric_score(rf1,x_train,x_test,y_train,y_test,train=True) #for training data
metric_score(rf1,x_train,x_test,y_train,y_test,train=False) # for testing data
```

```
-----Training score-----
Accuracy score:76.42%
-----Testing score-----
Accuracy score:76.36%
classification report
              precision    recall  f1-score   support

           0       0.78      0.73      0.75     43179
           1       0.75      0.80      0.77     43554

    accuracy                           0.76     86733
   macro avg       0.76      0.76      0.76     86733
weighted avg       0.76      0.76      0.76     86733

confusion matrix  [[31491 11688]
 [ 8816 34738]]
F1 Score 0.772127139364303
```

```
#cross validation score
print('Cross Validation Score for tuned Random Forest Classifier model :- ',((cross_val_score(rf1,x_new,y_new,cv=4).mean())*
```

```
Cross Validation Score for tuned Random Forest Classifier model :-  76.02584944600095
```

The tuned model provided testing accuracy of 76.36% with cross validation and F1 score as 76.02 and 77.21 respectively.

It was seen that the tuned model does not give good accuracy then the original model hence the original Random Forest model is saved as the best model.

{3.4} Key Metrics for success in solving problem under consideration

The key metrics used in this project are:

The key metrics used here were Accuracy Score, F1 score, Cross Validation Score, Roc Auc Score and Confusion Matrix. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and used RandomizedSearchCV method.

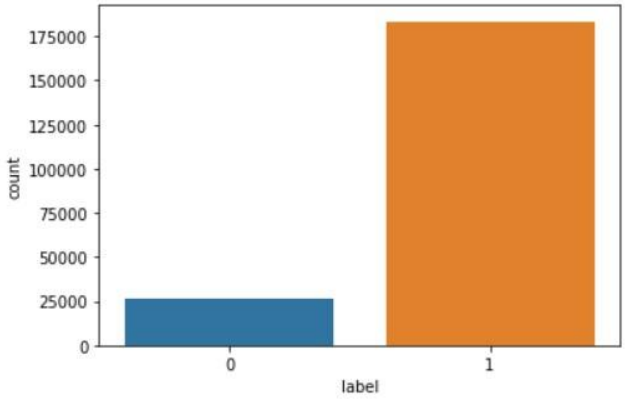
- Accuracy score means how accurate our model is that is the degree of closeness of the measured value to a standard or true value.
- F1 Score is used to express the performance of the machine learning model (or classifier). It gives the combined information about the precision and recall of a model. This means a high F1-score indicates a high value for both recall and precision.
- Cross Validation Score is a technique used to protect against overfitting in a predictive model, particularly in a case where the amount of data may
be limited.
- Roc Auc Score: The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various

threshold values. The Area Under Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

• Confusion Matrix is one of the evaluation metrics for machine learning classification problems, where a trained model is being evaluated for accuracy and other performance measures. The ML model is said to be good if its False Positive and False Negatives are less compared to True Negative and True Positive

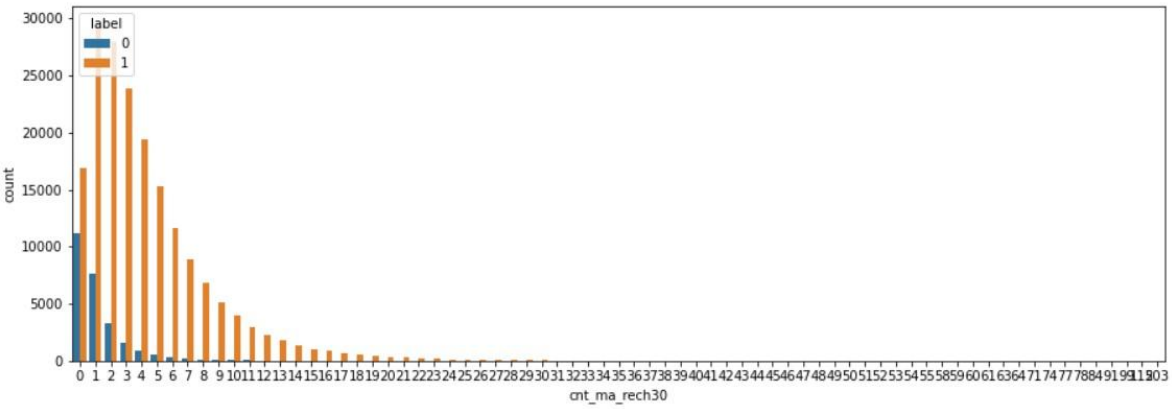
{3.5} Visualizations and its interpretations

Univariate Analysis



Observation:

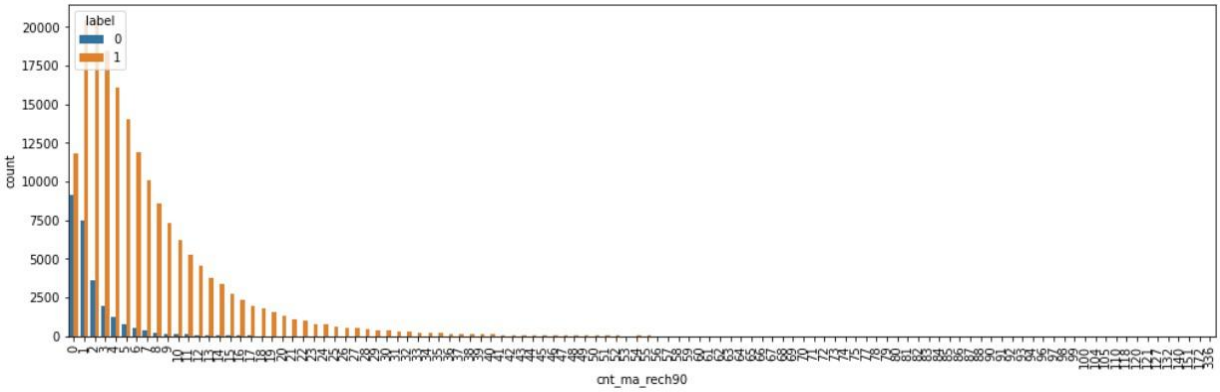
The above plot shows most of the people who took loan repayed on time while very less number of people are loan defaulters.



Observation:

It can be seen from the plot that highest number of non- defaulters had their main account recharged in last 30 days.

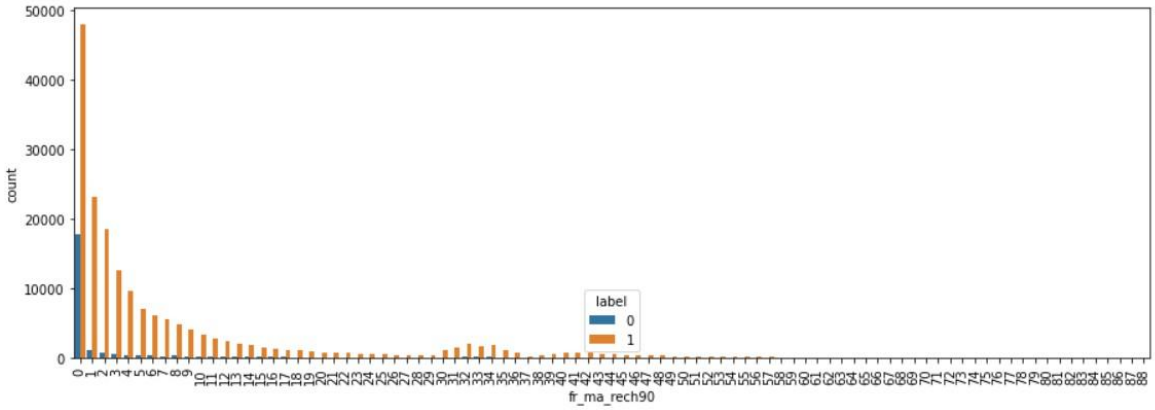
Most single number of people had their main account recharged in last 30 days, while as the number of persons the count of persons decreases



Observation:

It can be seen from the plot that highest number of non- defaulters had their main account recharged in last 90 days.

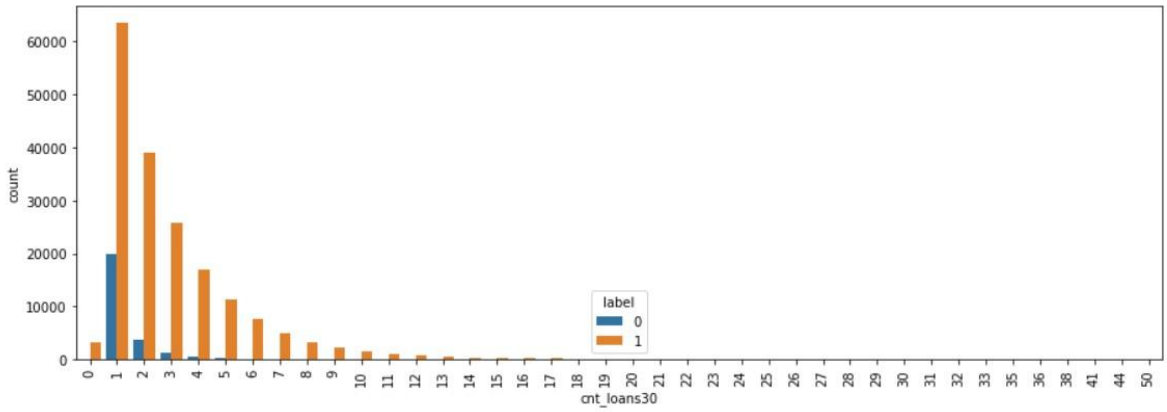
Mostly 1-3 number of people had their main account recharged in last 90 days, while as the number of persons the count of persons decreses



Observation:

The above plot shows the frequency of main account recharged in last 90 days.

It can be seen that most non-defaulters account recharged single time which is more than defaulters. This trend is seen throughout.



Observation:

It can be seen from the plot that most non-defaulters took only 1 loan in last 30 days than defaulters.

Bivariate Analysis

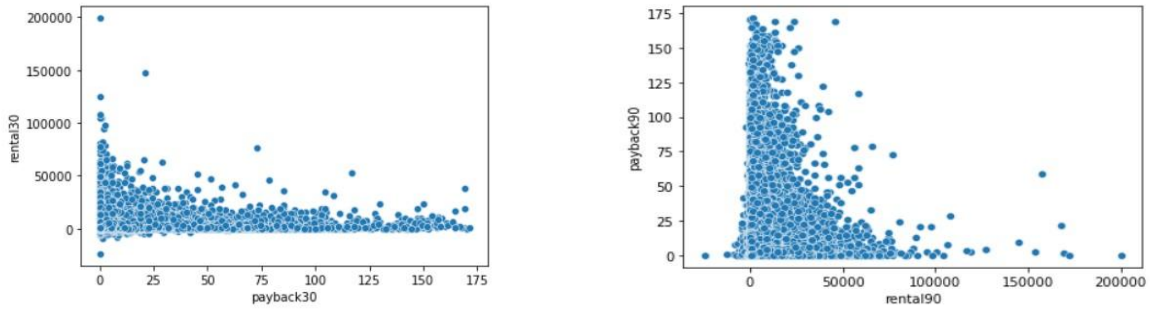


Figure 1 and 2 :rental30 vs payback30 (left) and rental90 vs payback90 (right)

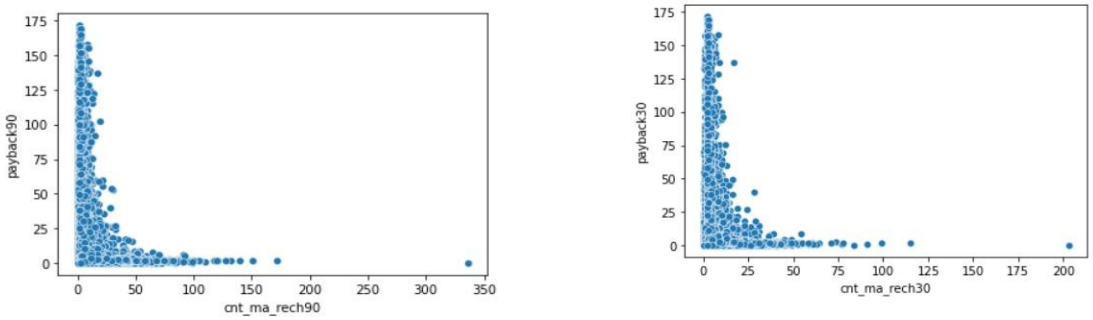


Figure 3 and 4:payback90 vs cnt_ma_rech90 (left) and payback30 vs cnt_ma_rech30(right)

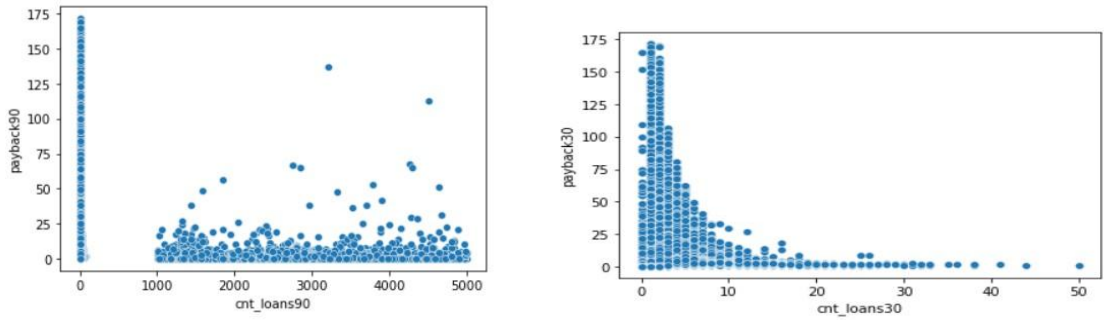


Figure 5 and 6 :payback90 vs cnt_loans90 (left) and payback30 vs cnt_loans30(right)

Observations for above plots:

Figure1: It can be seen from plot that if average main account balance over 30 days is less then average payback time in days for past 30 days increases.

Figure 2: It can be seen from the plot that if average main account balance over 90 days is less then average payback time in days for past 90 days also increases.

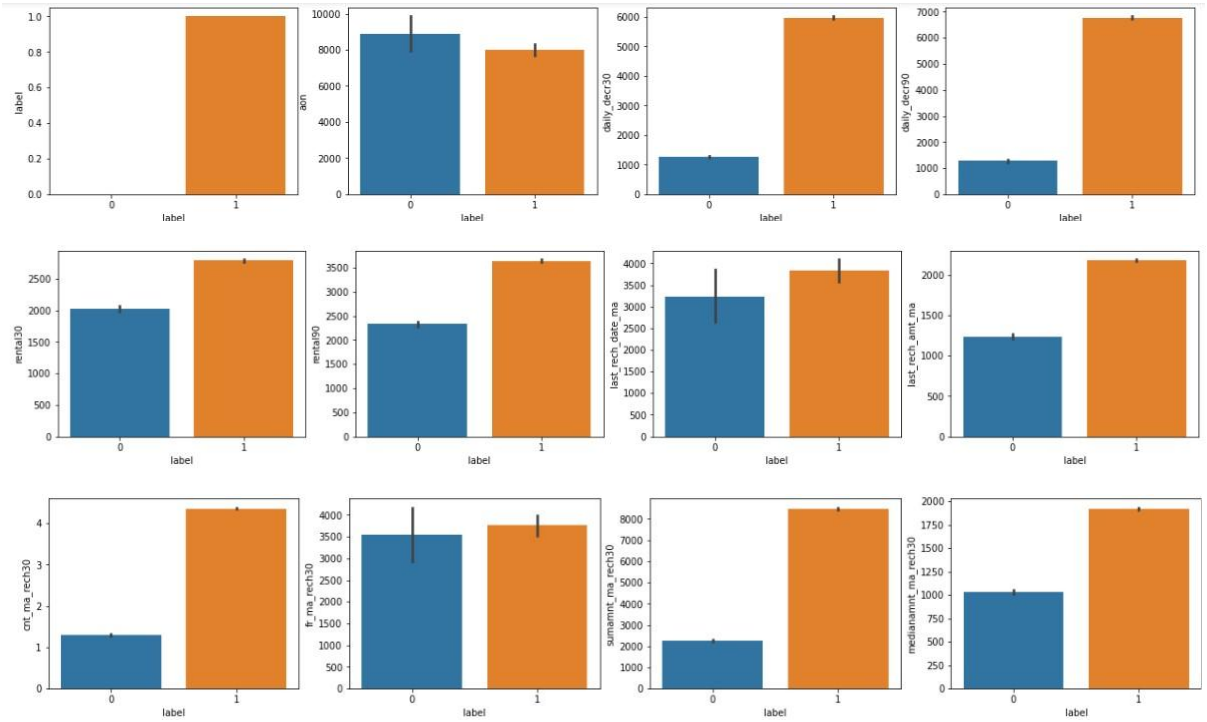
Figure 3: It can be seen from plot that if more number of times account got recharged in last 30 days then less then high number of payback time in days is seen.

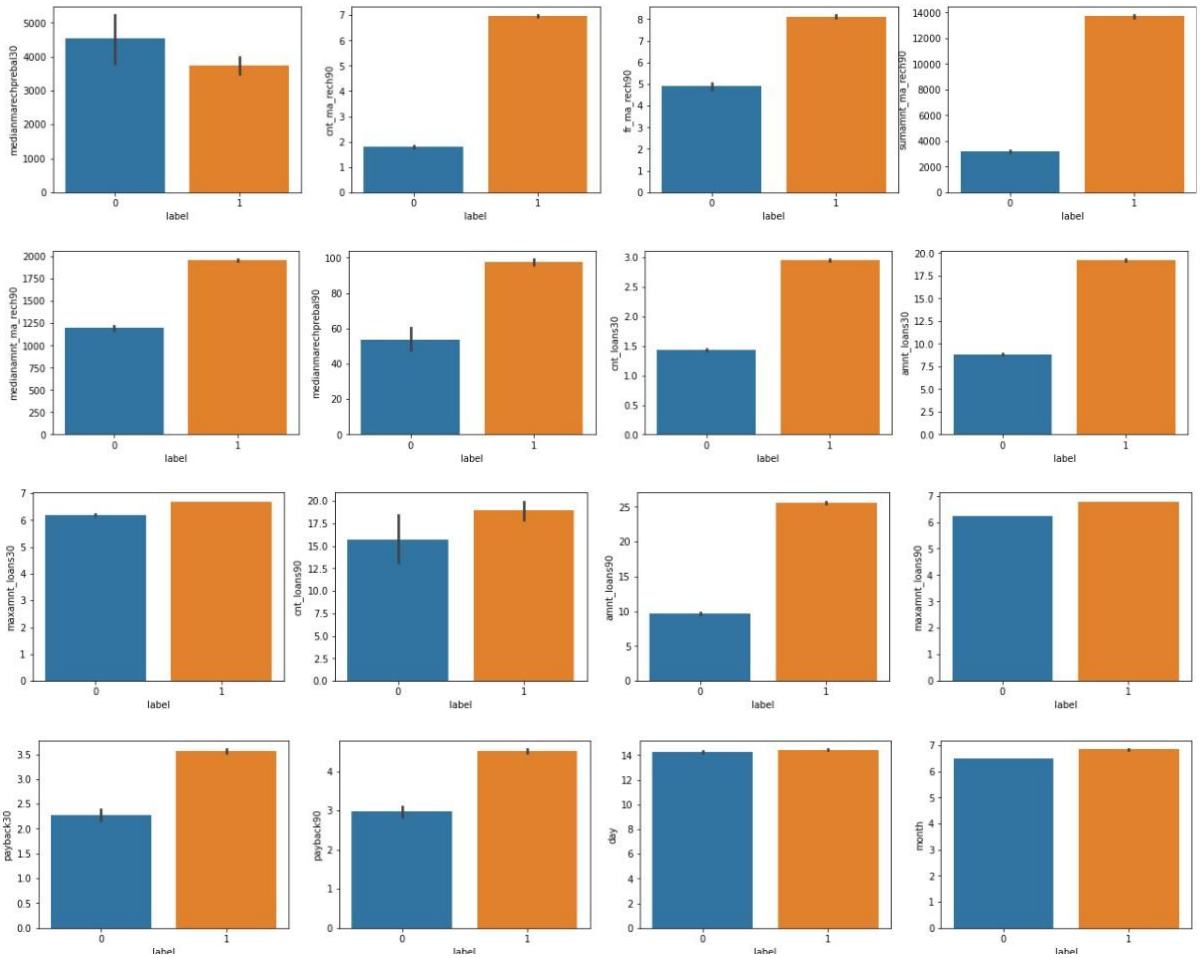
Figure 4:It can be seen from plot that if more number of times account got recharged in last 90 days then less then high number of payback time in days is seen.

Figure 5: It can be seen that of number of loans taken by user in last 30 days is more then its payback time in days is also faster.

Figure 6: There is no general trend seen in this plot.The number of loans that taken by user in past 90 days are mostly paid back in approx. 0-25 days.

Barplots with respect to label





Observation for above barplots:

Plot of label vs aon suggests that the loan defaulters age on cellular network in days is high compared to non defaulters age

Plot of label vs daily_decr30 and daily_decr90 suggests that daily amount spent from main account averaged over 30 and 90 days shows that most of the non-defaulters paid their amount within loan limit time while less number of defaulters i.e. around approx.>1000 did not pay loan amount within stipulated time.

Plot of label vs rental30 and rental90 suggests that main average account balance for 30 days incase of defaulters is almost 2000 while that of non-defaulters is >2500. Incase of average account balance for 90 days, defaulters is approx.2400 while that of non defaulters is >3500.

Plot of label vs last_rech_date_ma suggests that number of days till last recharge of main account is higher incase of non-defaulters while less in case of defaulters.

Plot of label vs last_rech_amt_ma suggests that Amount of last recharge of main account is higher incase of non-defaulters i.e.>2000 while incase of defaulters the amount is approx. 1250.

Plot of label vs cnt_ma_rech30 suggests that >1 or single time main account of defaulters got recharged in last 30 days while incase of non defaulters the main account was recharged >4 times. The low rate of account recharge may suggest no payment of loans within 5 days.

Plot of label vs fr_ma_rech30 suggests that Almost equal number of people have the frequency of main account being recharged in last 30 days.

Plot of label vs sumamnt_ma_rech30 suggests that Total amount of recharge in main account over last 30 days is most for non-defaulters i.e. >8000 than >2000 for defaulters.

Plot of label vs medianamnt_ma_rech30 suggests that median amount of recharges done in past 30 days in indonesian rupiah is higher for non-defaulters i.e. approx.1800 while that for defaulters is lower i.e. approx. 1000 rupiah.

Plot of label vs medianmarechprebal30 suggests that median of main account balance just before recharge for defaulters is higher i.e. approx. 4500 while that for non-defaulters is just over 3500 in indinesian rupiah.

Plot of label vs cnt_ma_rech90 suggests that >1 or 2 times main account of defaulters got recharged in last 90 days while incase of non defaulters the main account was recharged >6 times. The low rate of account recharge may suggest no payment of loans within 5 days.

Plot of label vs fr_ma_rech90 suggests that incase of defaulters the frequency of main account being recharged in last 90 days was 5 times while that incase of non-defaulters the frequency was 8 times.

Plot of label vs sumamnt_ma_rech90 suggests that Total amount of recharge in main account over last 90 days is most for non-defaulters i.e. >13000 than 3000 for defaultersin form of indonesian rupiah.

Plot of label vs medianamnt_ma_rech90 suggests that median of main account balance just before recharge for defaulters is higher i.e. approx. 4500 while that for non-defaulters is just over 3500 in indinesian rupiah.

Plot of label vs medianmarechprebal90 suggests that median amount of recharges done in past 90 days in indonesian rupiah is higher for non-defaulters i.e. approx.2000 while that for defaulters is lower i.e. approx. 1250 rupiah.

Plot of label vs cnt_loans30suggests that defaulters took average of 1.5 loans in past 30 days while non- defaulters took average of 3 loans in past 30 days.

Plot of label vs amnt_loans30 suggests the total number of loans taken by defaulters is 8 time while that of non-defaulters is 18 times in past 30 days.

Plot of label vs maxamnt_loans30 suggests the maximun amount of loans taken by defaulter is 6 times while that of non-defaulters is 7 times in past 30 days

Plot of label vs cnt_loans90 suggests that defaulters took average of >15 loans in past 90 days while non- defaulters took average of 18 loans in past 90 days.

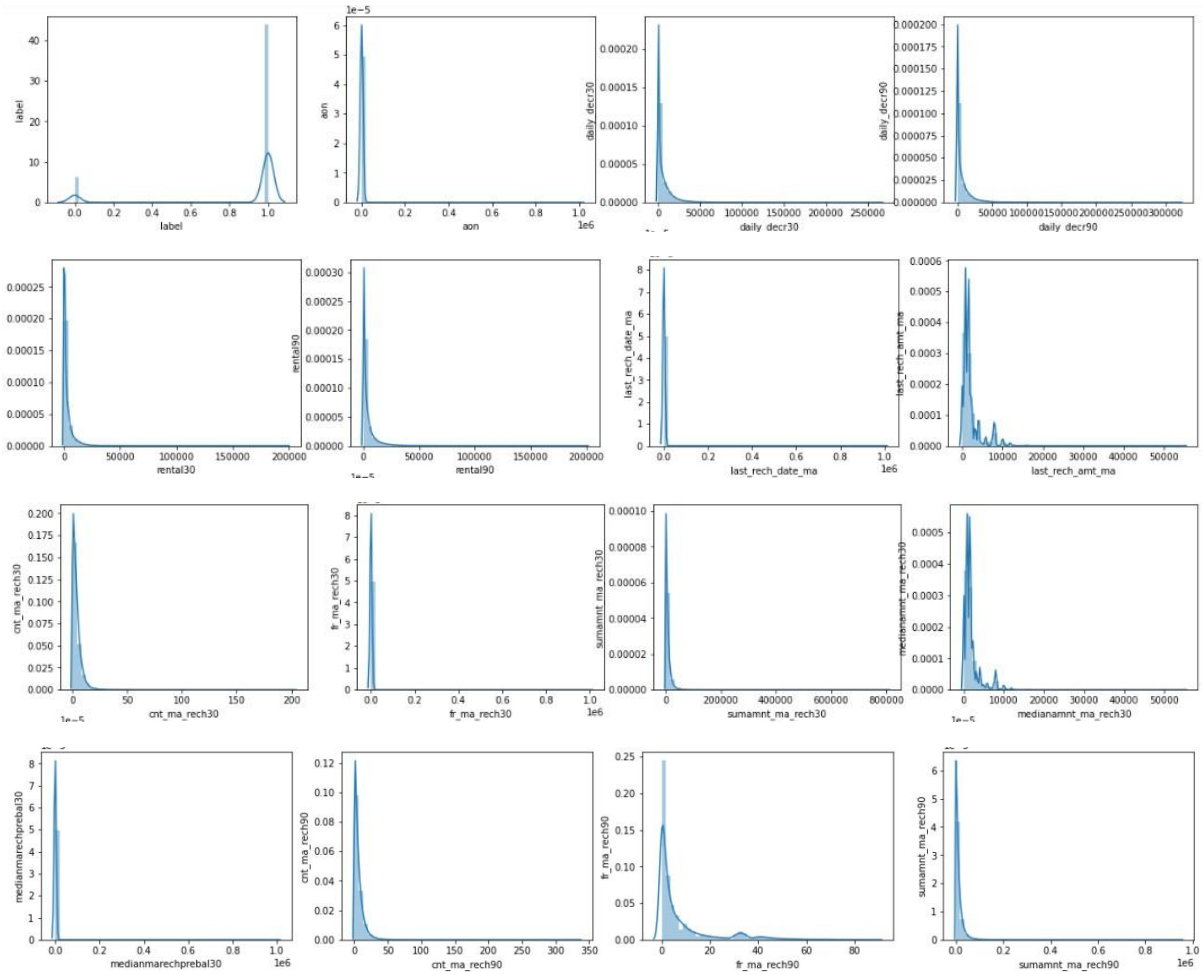
Plot of label vs amnt_loans90 suggests the total number of loans taken by defaulters is 10 times while that of non-defaulters is 28 times in past 90 days.

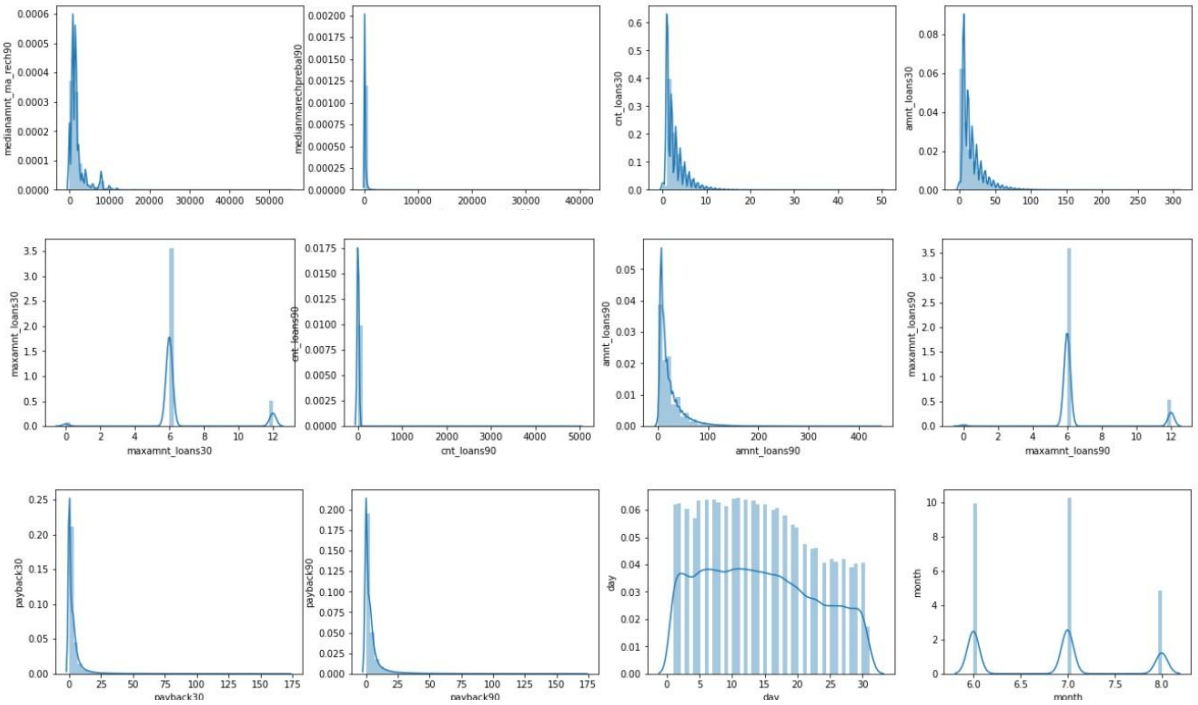
Plot of label vs maxamnt_loans90 suggests the maximun amount of loans taken by defaulter is >6 times while that of non-defaulters is 7 times in past 90 days.

Plot of label vs payback30 suggest that average payback time over days for 30 days is within 2-2.5 days for defaulters while that for non-defaulters is 3.5 times.

Plot of label vs payback90 suggest that average payback time over days for 30 days is within 3 days for defaulters while that for non-defaulters is >4 times.

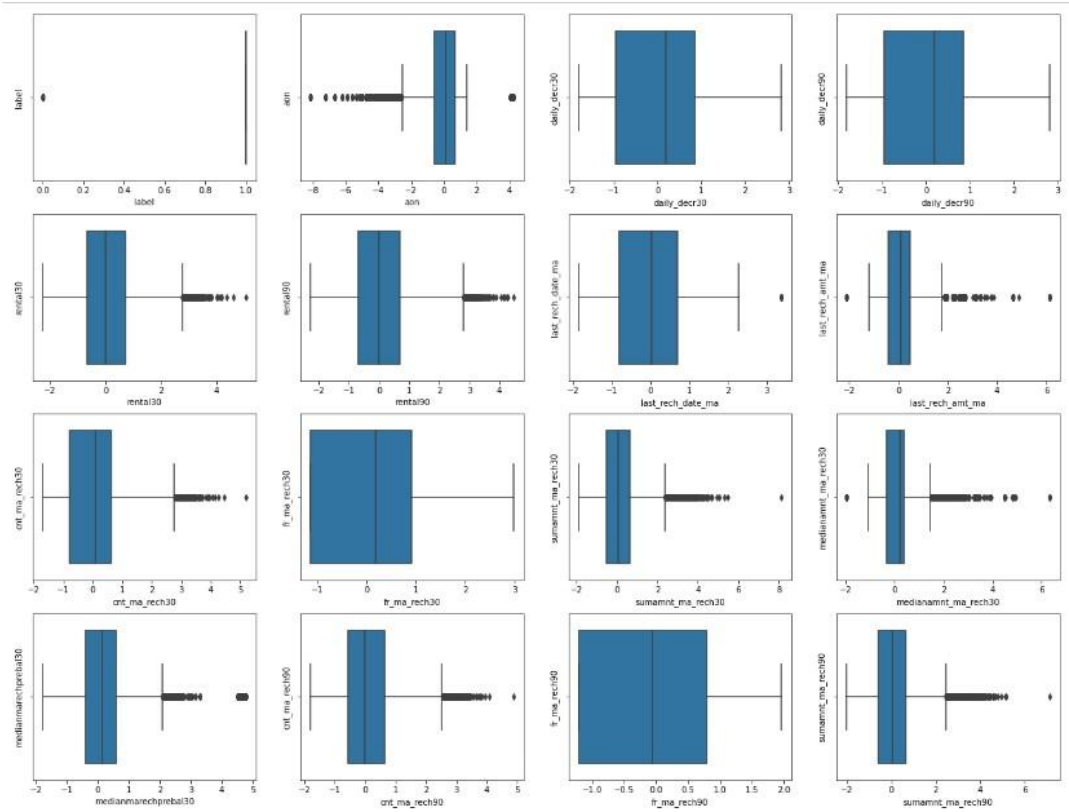
Distplots to check for skewness

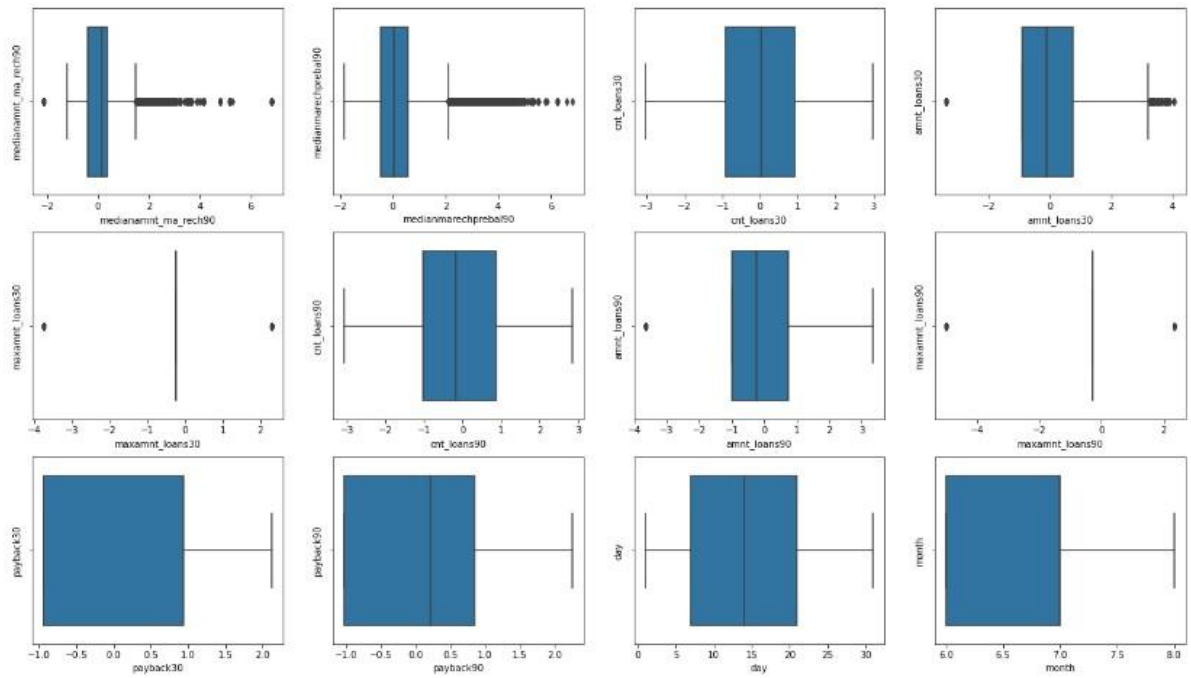




- From above plots it is seen that the data is highly skewed, hence data was further treated using power transformer method to remove/reduce skewness.

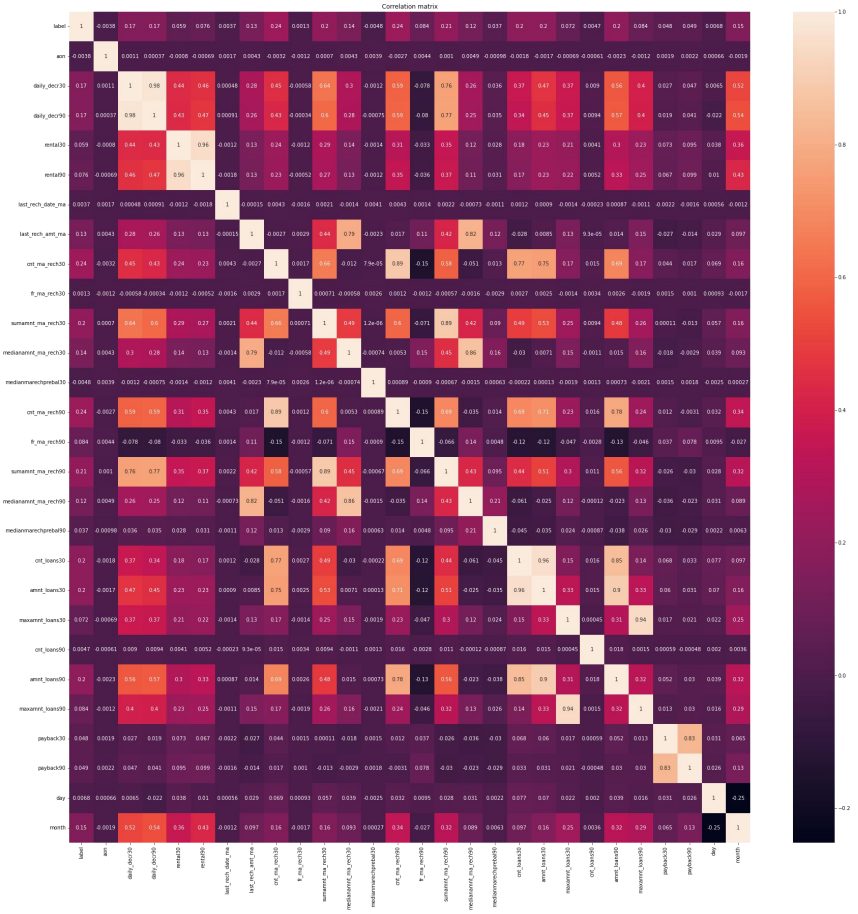
Boxplots to check for outliers





- Most of the columns in above plots has outliers which were further removed using z-score method.

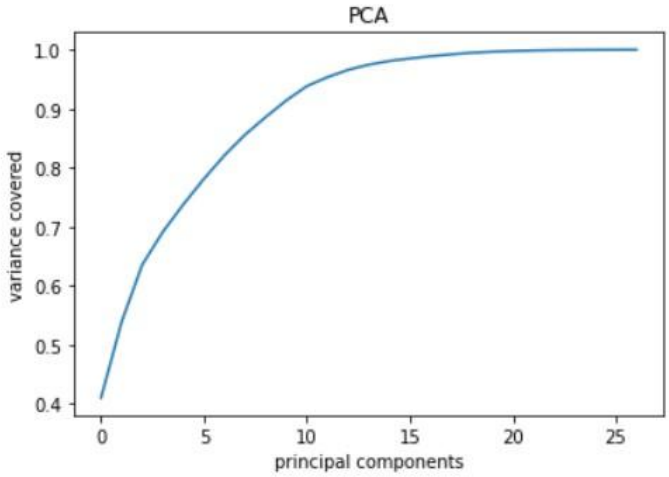
Multivariate Analysis:



Observation:

As seen in plot it showed that most of the columns showed multicollinearity, hence this was handled using Principal Component Analysis.

Principal Component Analysis



The dataset was divided into features and label and features were then standardized. This feature data was then subjected to Principal Component Analysis (PCA) which took care of the multicollinearity problem that was in dataset.

From plot it can be seen that taking 20 features covered almost all the data. Hence 20 features separate dataframe was made which was then subjected to model building.

[4] CONCLUSION

{4.1} Key Findings and Conclusions of the Study

The dataset was uploaded, cleaned and analysed using using plots. Further the dataset was subjected to removal or reduction of skewness followed by removal of outliers. Further the dataset was subjected to PCA analysis which provided total of 20 features that are responsible for model building. The label column was imbalanced hence it was balanced using SMOTE method. The data was trained on total of 6 algorithms which provided 6 ML models. The best models was selected using metrics like accuracy score, Cross validation score, F1 score and AUC score.

The following findings were seen from dataset:

- From the whole study I found that the MFIs have provided loan to the user who have no recharge or balance in their account which needs to be stopped.
- Also, the frequency of main account recharged in last 30 days & 90 days we have seen the users with low frequency are causing huge losses, company should implement some kind of strategies to reduce like sending SMS alerts for notification.
- We found the defaulting rate is higher in old customers list.

{4.2} Limitations of this work and Scope for Future Work

- The data provided was only for year 2016. If we get data for other years then it will be quite more interesting to handle and predict model.
- There were some cases wherein data was not inputted propely I.e. incase where 6,12 or 0 values should have been, there were some other values. This kind of things should be minimized.
- Due to the class imbalance we had to balance the class defaulter (0). This might also have some effect on model.