Micro-Credit Defaulter Model

FLIP ROBO

ABHISHEK MISHRA

Micro-Credit Defaulter Model

SUBMITTED BY

ABHISHEK MISHRA

ACKNOWLEDGEMENT

First of all, I extol the Almighty for pouring his blessings on me and giving me potentiality and opportunity to carry the work to its end with a success.

"It's impossible to prepare a project report without the help and fillip of some people and certainly this project report is of no exception."

At the commencement of this project report I would like to evince my deepest sense of gratitude to Ms. Astha Mishra, my honored mentor. Without her guidance, insightful decision, valuable comments and correction it would not have possible to reach up to this mark.

I would like to draw my gratitude to Flip Robo and Data Trained for providing me a suitable environment and guidance to complete my work. Last but not least thanks to the brilliant authors from where I have got the idea to carry out the project.

References were taken from various articles from Medium, KDnuggets, Towards Data Science, Machine Learning Mastery, Analytics Vidya, American Statistical Association, Research Gate and documentations of Python and Sklearn.

Abhishek Mishra

CONTENTS

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.1 BUSINESS PROBLEM

Our client is a Telecom Industry having fixed wireless telecommunication network; they are keen to provide better services at low price range.

They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The consumer will be considered as a defaulter if he doesn't pay back the loaned amount within 5 days. The consumer has to pay 6 Indonesian Rupiah for a loan of 5 Indonesian Rupiah and for a loan of 10 Indonesian Rupiah the payback amount should be 12 Indonesian Rupiah.

1.2 BACKGROUND

As a result of incapacity of development and traditional banks to effectively finance the low-income population of the world, microfinance seems like a continuum between pure capitalism and socialism economies (World Bank, 2008). Access to formal financial services has been limited for many, if not most, of the world's poorest: more than 2.5 billion people do not use formal financial services.

In developing countries, the problems triggered by informational lopsidedness that are distinctive to credit markets are intensified since low-income people lack collateral that can be provided against loans and because of the weak legal system enforcement cannot be possible,

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. In current world scenario microfinance is widely accepted as a poverty-reduction tool, and it represents $70 billion in outstanding loans and a global outreach of 200 million clients.

In current world scenario the importance of communication in a person life is known to all. So here our client having fixed wireless telecommunication network trying to introduce services low price in collaboration with a MFI. The low-income group are the main target of this new launch.

As we know that the telecom sector is one of the most competitive fields so this data is very helpful in understanding the problem for the lower-class people specially by providing them the facility of network and the credit amount provided by the help of MFI and MFS. From this data we get to know that what the criteria to become defaulters and successor are. And the useful information from the data to know how much amount people spend on data recharge or on the main balance recharge.

1.3 MOTIVATION FOR THE PROBLEM UNDERTAKEN

The project was the first provided to me by FlipRobo as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary motivation. Further diving into the dataset, the motive is to help the poor or low-income band to have continuous access to their mobile accounts and to make emergency calls even when they do not have account balance making use of the loan facility. Alternatively, it is also important to ensure that the provider does not incur a loss for providing the facility. Hence, the entire focused is on building a model that can effectively predicts a defaulter by using the historical data which would help in approval process of the loans to end users with a clean sheet.

CHAPTER-2

ANALYTICAL PROBLEM FRAMING

2.1 Analytical Modeling of the Problem

The dataset provided has a shape of (209593, 37). Here the target or the dependent variable named "Label" have two different distinct values 0 and 1. Where 0 represents the defaulter & 1 represents the non-defaulter category of people. As the target is giving binary output so, it is a classification-based problem. Here the dataset has no null values and no duplicate values.

Different values of the dataset like the count, mean, standard deviation, min, 25%,50%,75%, max can be obtained using df.describe() function. The values obtained from this function shows that maximum values have high standard deviation.

```
sns.set(context='talk',style='whitegrid',palette='dark',font='monospace',font_scale=0.8)
plt.figure(figsize=(15,5),dpi=120)
sns.heatmap(micro.isnull(),cmap='brg')
plt.show()
```

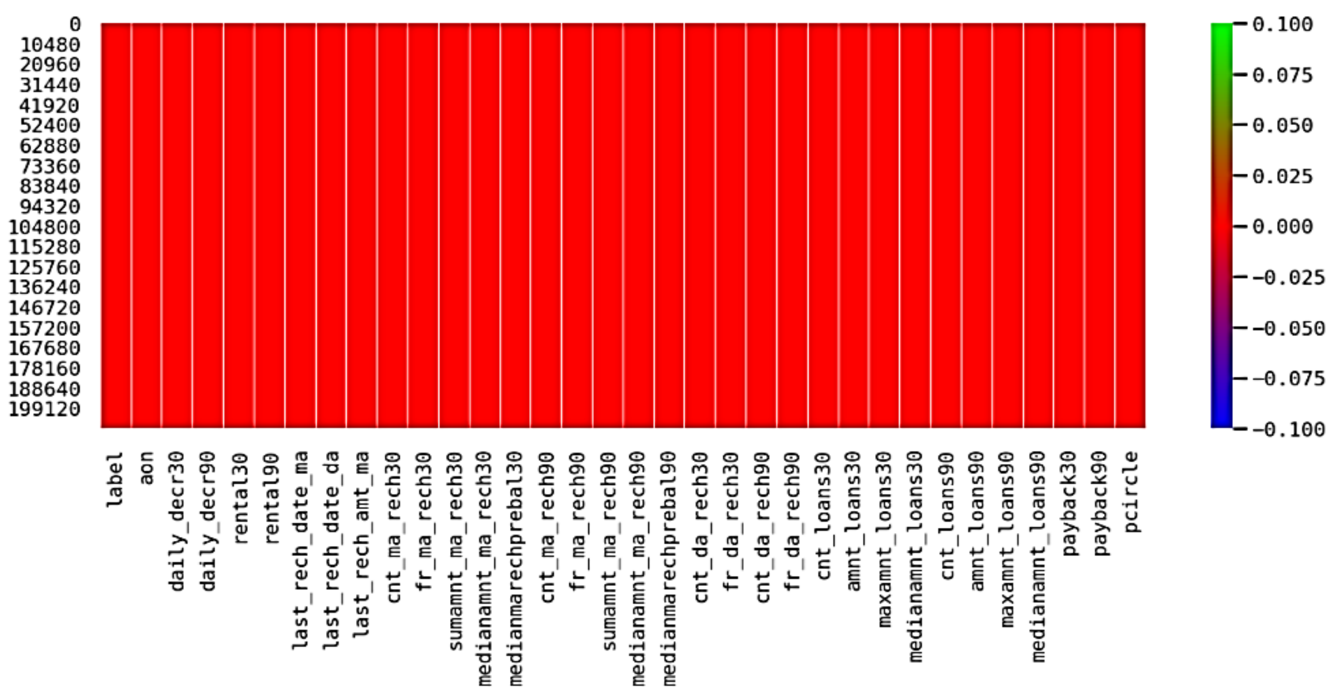

Fig.1: NULL VALUES IN THE DATASET

The describe function showed that few columns have values less than zero which is absurd and the of boxplot have shown the outliers in the dataset but we can’t remove all of them using Zscore method as this will lead to loss of more than 10% of data so all the outliers were removed using proper condition.

```
mfi.drop(mfi.index[mfi['aon']<0], inplace = True)
mfi.drop(mfi.index[mfi['daily_decr30']<0], inplace = True)
mfi.drop(mfi.index[mfi['daily_decr90']<0], inplace = True)
mfi.drop(mfi.index[mfi['rental30']<0], inplace = True)
mfi.drop(mfi.index[mfi['rental90']<0], inplace = True)
mfi.drop(mfi.index[mfi['last_rech_date_ma']<0], inplace = True)
mfi.drop(mfi.index[mfi['last_rech_date_da']<0], inplace = True)
mfi.drop(mfi.index[mfi['medianmarechprebal30']<0], inplace = True)
mfi.drop(mfi.index[mfi['cnt_ma_rech90']<0], inplace = True)
mfi.drop(mfi.index[mfi['medianmarechprebal90']<0], inplace = True)
```

999860.75 days or 2739.34 year on cellular network is impossible so removing any data where aon is greater than 7300 days or 20 years.

```
mfi.drop(mfi.index[mfi['aon']>7301], inplace = True)
```

Fig.2: NEGATIVE VALUES IN DATASET

2.2 DATA SOURCES AND THEIR FORMATS

The data source obtained was of CSV form and have 209593 rows and 37 columns. The df.info() function shows the data types of the columns.

```
1 micro.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 34 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   label                 209593 non-null  int64
 1   aon                   209593 non-null  float64
 2   daily_decr30          209593 non-null  float64
 3   daily_decr90          209593 non-null  float64
 4   rental30              209593 non-null  float64
 5   rental90              209593 non-null  float64
 6   last_rech_date_ma     209593 non-null  float64
 7   last_rech_date_da     209593 non-null  float64
 8   last_rech_amt_ma      209593 non-null  int64
 9   cnt_ma_rech30         209593 non-null  int64
 10  fr_ma_rech30          209593 non-null  float64
 11  sumamnt_ma_rech30     209593 non-null  float64
 12  medianamnt_ma_rech30  209593 non-null  float64
 13  medianmarechprebal30  209593 non-null  float64
 14  cnt_ma_rech90         209593 non-null  int64
 15  fr_ma_rech90          209593 non-null  int64
 16  sumamnt_ma_rech90     209593 non-null  int64
 17  medianamnt_ma_rech90  209593 non-null  float64
 18  medianmarechprebal90  209593 non-null  float64
 19  cnt_da_rech30         209593 non-null  float64
 20  fr_da_rech30          209593 non-null  float64
 21  cnt_da_rech90         209593 non-null  int64
 22  fr_da_rech90          209593 non-null  int64
 23  cnt_loans30           209593 non-null  int64
 24  amnt_loans30          209593 non-null  int64
 25  maxamnt_loans30       209593 non-null  float64
 26  medianamnt_loans30    209593 non-null  float64
 27  cnt_loans90           209593 non-null  float64
 28  amnt_loans90          209593 non-null  int64
 29  maxamnt_loans90       209593 non-null  int64
 30  medianamnt_loans90    209593 non-null  float64
 31  payback30             209593 non-null  float64
 32  payback90             209593 non-null  float64
 33  pcircle               209593 non-null  object
dtypes: float64(21), int64(12), object(1)
memory usage: 54.4+ MB
```

Fig.3: INFO OF THE DATASET

Label	**Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan {1: success, 0: failure}**
Msisdn	**mobile number of users**
Aon	**age on cellular network in days**
daily_decr30	**Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)**
daily_decr90	**Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)**
rental30	**Average main account balance over last 30 days**
rental90	**Average main account balance over last 90 days**
last_rech_date_ma	**Number of days till last recharge of main account**
last_rech_date_da	**Number of days till last recharge of data account**
last_rech_amt_ma	**Amount of last recharge of main account (in Indonesian Rupiah)**
cnt_ma_rech30	**Number of times main account got recharged in last 30 days**
fr_ma_rech30	**Frequency of main account recharged in last 30 days**
sumamnt_ma_rech30	**Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)**
medianamnt_ma_rech30	**Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)**

medianmarechprebal30	**Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)**
cnt_ma_rech90	**Number of times main account got recharged in last 90 days**
fr_ma_rech90	**Frequency of main account recharged in last 90 days**
sumamnt_ma_rech90	**Total amount of recharge in main account over last 90 days (in Indonesian Rupee)**
medianamnt_ma_rech90	**Median of amount of recharges done in main account over last 90 days at user level (in Indonesian Rupee)**
medianmarechprebal90	**Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupee)**
cnt_da_rech30	**Number of times data account got recharged in last 30 days**
fr_da_rech30	**Frequency of data account recharged in last 30 days**
cnt_da_rech90	**Number of times data account got recharged in last 90 days**
fr_da_rech90	**Frequency of data account recharged in last 90 days**
cnt_loans30	**Number of loans taken by user in last 30 days**
amnt_loans30	**Total amount of loans taken by user in last 30 days**
maxamnt_loans30	**maximum amount of loan taken by the user in last 30 days**
medianamnt_loans30	**Median of amounts of loan taken by the user in last 30 days**
cnt_loans90	**Number of loans taken by user in last 90 days**
amnt_loans90	**Total amount of loans taken by user in last 90 days**
maxamnt_loans90	**maximum amount of loan taken by the user in last 90 days**
medianamnt_loans90	**Median of amounts of loan taken by the user in last 90 days**
payback30	**Average payback time in days over last 30 days**
payback90	**Average payback time in days over last 90 days**
Pcircle	**telecom circle**
Pdate	**Date**

TABLE 1: METADATA

2.3 DATA PREPROCESSING

After loading the dataset, the null and duplicate values were checked, the absurd values and skewed values were removed with proper conditions. The columns named ‘msisdn’, ‘pcircle’ and ‘date’ were dropped as they served no purpose in modelling. After preprocessing the dataset shape has been reduced to 189339,33.

```
print('Shape of the dataset - ',micro.shape)
print('\nColumns in the dataset-\n\n',micro.columns.values)
```

```
Shape of the dataset -  (209593, 37)

Columns in the dataset-

 ['Unnamed: 0' 'label' 'msisdn' 'aon' 'daily_decr30' 'daily_decr90'
 'rental30' 'rental90' 'last_rech_date_ma' 'last_rech_date_da'
 'last_rech_amt_ma' 'cnt_ma_rech30' 'fr_ma_rech30' 'sumamnt_ma_rech30'
 'medianamnt_ma_rech30' 'medianmarechprebal30' 'cnt_ma_rech90'
 'fr_ma_rech90' 'sumamnt_ma_rech90' 'medianamnt_ma_rech90'
 'medianmarechprebal90' 'cnt_da_rech30' 'fr_da_rech30' 'cnt_da_rech90'
 'fr_da_rech90' 'cnt_loans30' 'amnt_loans30' 'maxamnt_loans30'
 'medianamnt_loans30' 'cnt_loans90' 'amnt_loans90' 'maxamnt_loans90'
 'medianamnt_loans90' 'payback30' 'payback90' 'pcircle' 'pdate']
```

Fig.5: DATASET

```
print(' Earlier the shape of dataset with outliers was:',micro.shape,'\n The shape of the dataset after outlier removal is:',mfi.shape,
     '\n Percentage of data_loss:', data_loss)
```

```
Earlier the shape of dataset with outliers was: (209593, 33)
The shape of the dataset after outlier removal is: (189339, 33)
Percentage of data_loss: 9.663
```

Fig.6: PROCESSED DATASET

2.4 HARDWARE & TOOL USED

In this project the below mentioned machine, IDE and packages were used;

HARDWARE	LAPTOP: ASUS TUF A17 OS: WIN 10 HOME BASIC PROCESSOR: AMD RYZEN 7 4800H RAM: 16GB VRAM: 6GB NVIDIA GTX 1660Ti
LANGUAGE	Python 3.8
IDE	JUPYTER NOTEBOOK 6.0.3
PACKAGES	PANDAS, NUMPY, SCIPY, SKLEARN, MATPLOTLIB, SEABORN

TABLE 2: DEVICE AND TOOLS

CHAPTER-3

DEVELOPMENT AND EVALUATION

3.1 IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES

After loading the dataset, the absurd values, unwanted columns and the skewness were removed. Which leads to removal of approximately 9% of data. After this the dataset is split into two-part, one part named ‘x’ contains all the independent columns and one part ‘y’ which contains the target column.

The x part then normalized using standard scaler and y part is converted into numpy array and reshaped so that the shape of x and y remain same. Once this part is completed, they can be sent for modelling.

```
x=mfi.drop(['label'],axis=1)
y=mfi.label
```

```
ss=StandardScaler()
x=ss.fit_transform(x)
print(x)
```

```
[[-0.63497471  0.53116441  0.50874106 ... -0.23671909  2.6639062
   2.35395517]
 [ 0.46850625  1.00573954  0.97600258 ... -0.23671909 -0.85094496
  -0.92930934]
 [ 0.14051447  0.26206303  0.24207332 ... -0.23671909 -0.85094496
  -0.92930934]
 ...
 [ 0.87331506  0.99772413  0.96976859 ... -0.23671909  0.81227363
   0.40891675]
 [ 1.48936289  1.01598956  0.9883761  ... -0.23671909 -0.85094496
   1.38544149]
 [ 1.38457446  0.66369256  0.64182028 ... -0.23671909 -0.85094496
  -0.92930934]]
```

```
y=np.array(y)
y=y.reshape(-1,1)
```

```
print('shape of x:',x.shape,'\nshape of y:',y.shape)
```

```
shape of x: (189339, 32)
shape of y: (189339, 1)
```

Fig.7: DATA PREPARATION

3.2 TESTING OF IDENTIFIED APPROACHES

Once the normalized data were obtained, they can be sent for modelling. Here the output column 'label' is generating binary output so it's a classification-based problem and we can use the following algorithms for modelling and the highest performing algorithms to get our final model;

- Logistic Regression
- Decision Tree Classifier
- Gaussian NB
- Random Forest Classifier
- XgBoost Classifier

With the help of RandomizedSearchCV hyper parameter tuning will be done and the best parameters for each model will be found.

During modelling various metrices like f1 score, confusion matrix, accuracy score, classification report, roc curve, auc, roc auc score, mean squared error, precision score, recall score will be used to determine the performance of the model.

To check whether the model suffering from over fitting or underfitting cross val score will be used. To view the best performing model AUC Curve will be used. At the end the best model will be saved using Joblib library.

3.3 RUNNING AND EVALUATION

Here I have generated a function which will find the best random score for the model in a range of 25 to 180. It'll also show the confusion matrix, accuracy score, classification report, roc curve, auc, roc auc score, mean squared error, precision score, recall score, tpr, fpr at that random score.

The values so obtained will be then append to their respective lists. This function will also draw the AUC Curve and heat map of the confusion matrix. Below are the few images of modelling, heatmap and auc curve.

LOGISTIC REGRESSION

```
logi=LogisticRegression()
acusr(logi,x,y)
```

```
In this LogisticRegression()

The best suited RANDOM SCORE= 123

ACCURACY SCORE: 0.8929791204535051

CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.61      0.18      0.28      6785
           1       0.90      0.98      0.94     50017

    accuracy                           0.89     56802
   macro avg       0.75      0.58      0.61     56802
weighted avg       0.86      0.89      0.86     56802


CONFUSION MATRIX:
 [[ 1235  5550]
 [  794 49223]]

PRECISION:
 0.8986727037043799

RECALL:
 0.9841253973648959

MEAN SQUARED ERROR:
 0.1116862082321045

ROOT MEAN SQ. ERROR:
 0.334194865657904

AUC_ROC Score:
 0.5830722786382327

TPR: [0.         0.81798084 1.        ]
FPR: [0.        0.9841254 1.        ]
```

Fig.8: MODELLING LOGISTIC REGRESSION

```
#using cross_val_score to check for over/under fitting of logistic regressor model
logi_accuracy=cvs(logi,x,y,scoring='accuracy',cv=80)
print('THE ACCURACY SCORE AT LOGISTIC MODEL IS=', logi_accuracy.mean())
CV_ACC.append(logi_accuracy.mean())
```

```
THE ACCURACY SCORE AT LOGISTIC MODEL IS= 0.8891987036638251
```

Fig.9: CROSS VAL SCORE LOGISTIC REGRESSION

RANDOM FOREST CLASSIFIER

```
rfc=RandomForestClassifier()
rfc_para={"n_estimators":[300,350,400],'max_depth':[3,6,9],'criterion':['gini','entropy']}
rfc_rsv=rsv(rfc,rfc_para,cv=30,n_jobs=-1)
rfc_rsv.fit(x,y)
print(rfc_rsv)
print('\nbest score=',rfc_rsv.best_score_)
print('\nbest parameters for RFC=\n',rfc_rsv.best_params_)
```

```
RandomizedSearchCV(cv=30, estimator=RandomForestClassifier(), n_jobs=-1,
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_depth': [3, 6, 9],
                                        'n_estimators': [300, 350, 400]})

best score= 0.909126004969696

best parameters for RFC=
 {'n_estimators': 300, 'max_depth': 9, 'criterion': 'gini'}
```

Fig.10: RANDOMIZED SEARCH CV RFC

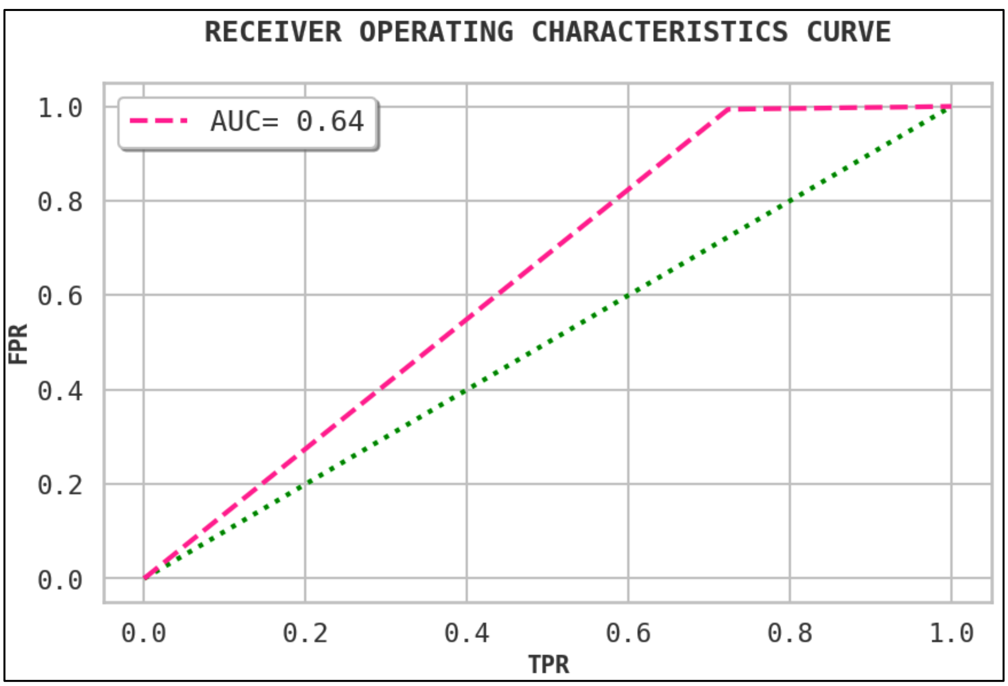

Fig.11: AUC OF RANDOM FOREST CLASSIFIER

XGBOOST CLASSIFIER

```
1  XGB = XGBClassifier()
2  acusr(XGB,x,y)
```

```
In this XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)

The best suited RANDOM SCORE= 123

ACCURACY SCORE: 0.9191753811485511

CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.76      0.43      0.55      6785
           1       0.93      0.98      0.95     50017

    accuracy                           0.92     56802
   macro avg       0.84      0.70      0.75     56802
weighted avg       0.91      0.92      0.90     56802


CONFUSION MATRIX:
 [[ 2886  3899]
 [  905 49112]]

PRECISION:
 0.9264492275188169

RECALL:
 0.9819061519083512

MEAN SQUARED ERROR:
 0.08457448681384458

ROOT MEAN SQ. ERROR:
 0.2908169300674302

AUC_ROC Score:
 0.7036280943771676

TPR: [0.         0.57464996 1.        ]
FPR: [0.         0.98190615 1.        ]
```

Fig.12: MODELLING XGBOOST

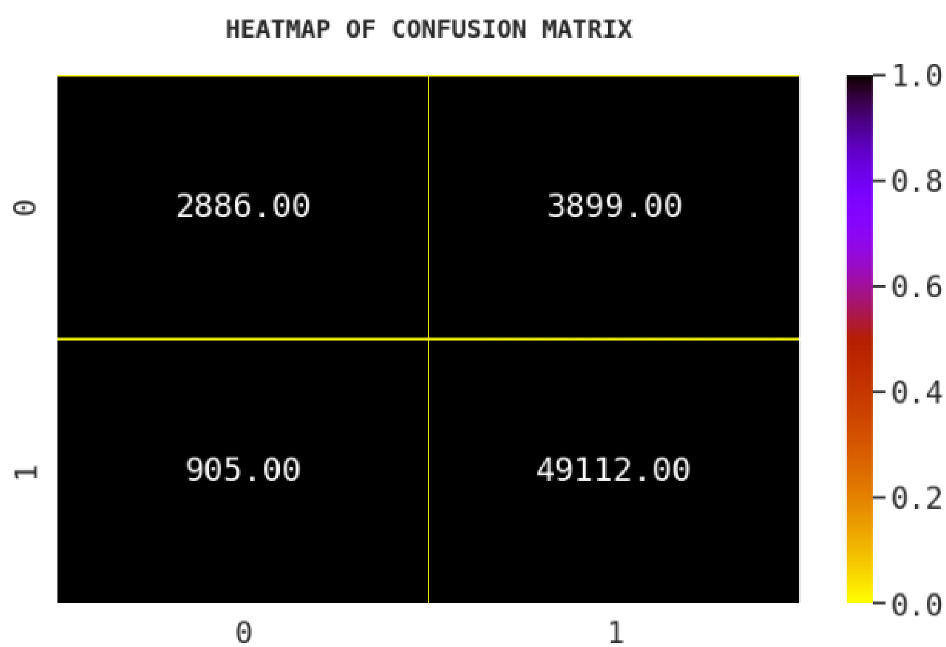

Fig.13: HEATMAP OF CONFUSION MATRIX OF XGBOOST

3.3 METRICE OF EVALUATION

In the modeling I have chosen metrices like Precision, Recall, Mean Squared Error, Root Mean Square Error, Classification Report, Accuracy score, Confusion Matrix, AUC, tpr, fpr and Cross val Score as my evaluation criteria. All the values were stored in a list and later they were saved in form of a DataFrame for proper evaluation and visualization of the values. Basing on the values the best model has been selected.

```
model=["LOGISTIC",'DTC','GAUSSIAN-NB','RANDOM FOREST','XGBOOST']
results = pd.DataFrame({"MODEL":model,'Acuracy':ACCURACY,'Precision': PRECESION ,'Recall': RECALL,
                        'RMSE':RMSE,'MSE':MSE,'AUC':AUC,'TPR':TPR,'FPR':FPR,"CV_ACCURACY":CV_ACC})



results.style.set_properties(**{'background-color':'midnightblue','color': 'gold','border-color': 'darkorange'})
```

	MODEL	Acuracy	Precision	Recall	RMSE	MSE	AUC	TPR	FPR	CV_ACCURACY
0	LOGISTIC	0.892979	0.898673	0.984125	0.334195	0.111686	0.583072	[0. 0.81798084 1.]	[0. 0.9841254 1.]	0.889199
1	DTC	0.911975	0.922459	0.977548	0.303524	0.092127	0.685900	[0. 0.60574797 1.]	[0. 0.97754763 1.]	0.910911
2	GAUSSIAN-NB	0.797278	0.947967	0.807166	0.456961	0.208813	0.740281	[0. 0.3266028 1.]	[0. 0.80716556 1.]	0.792890
3	RANDOM FOREST	0.912855	0.910144	0.993922	0.302915	0.091757	0.635281	[0. 0.72336035 1.]	[0. 0.99392207 1.]	0.908909
4	XGBOOST	0.919175	0.926449	0.981906	0.290817	0.084574	0.703628	[0. 0.57464996 1.]	[0. 0.98190615 1.]	0.916293

Fig.14: RESULTS

3.4 VISUALIZATION

Visualization is a part of EDA which helps to understand the data better. Here to understand the data various plots like Heatmap, Countplot, Bar graph were plotted.

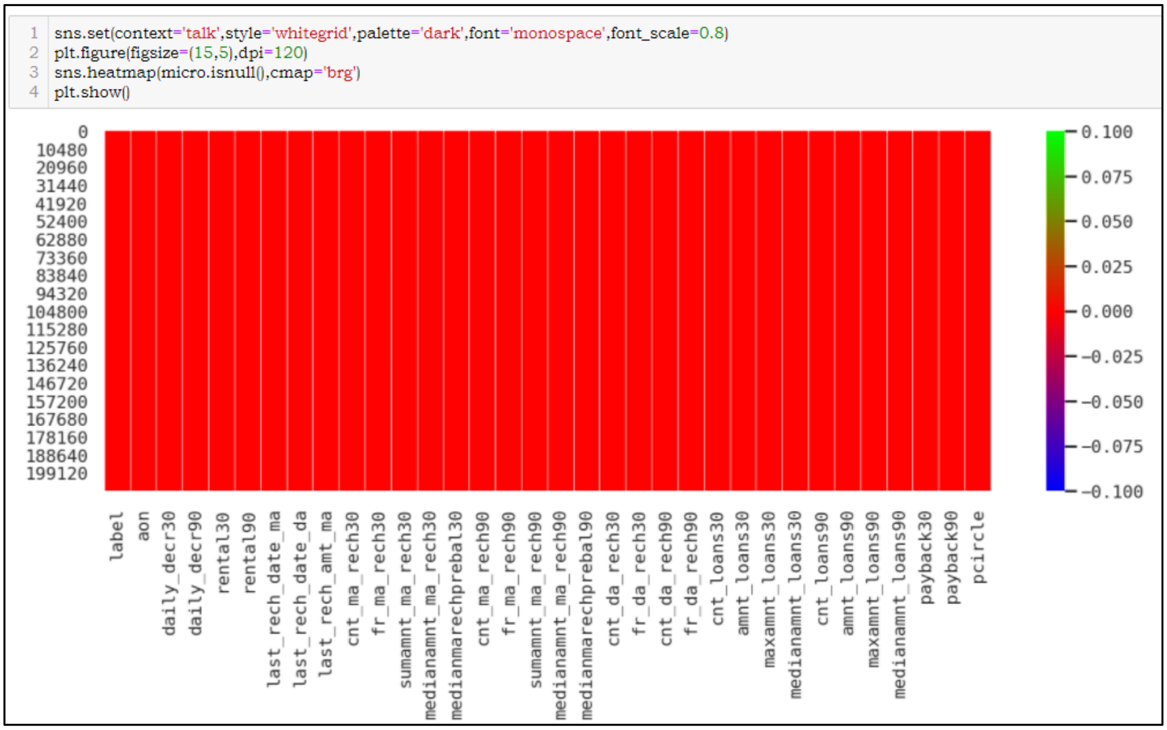

Fig.15: NULL

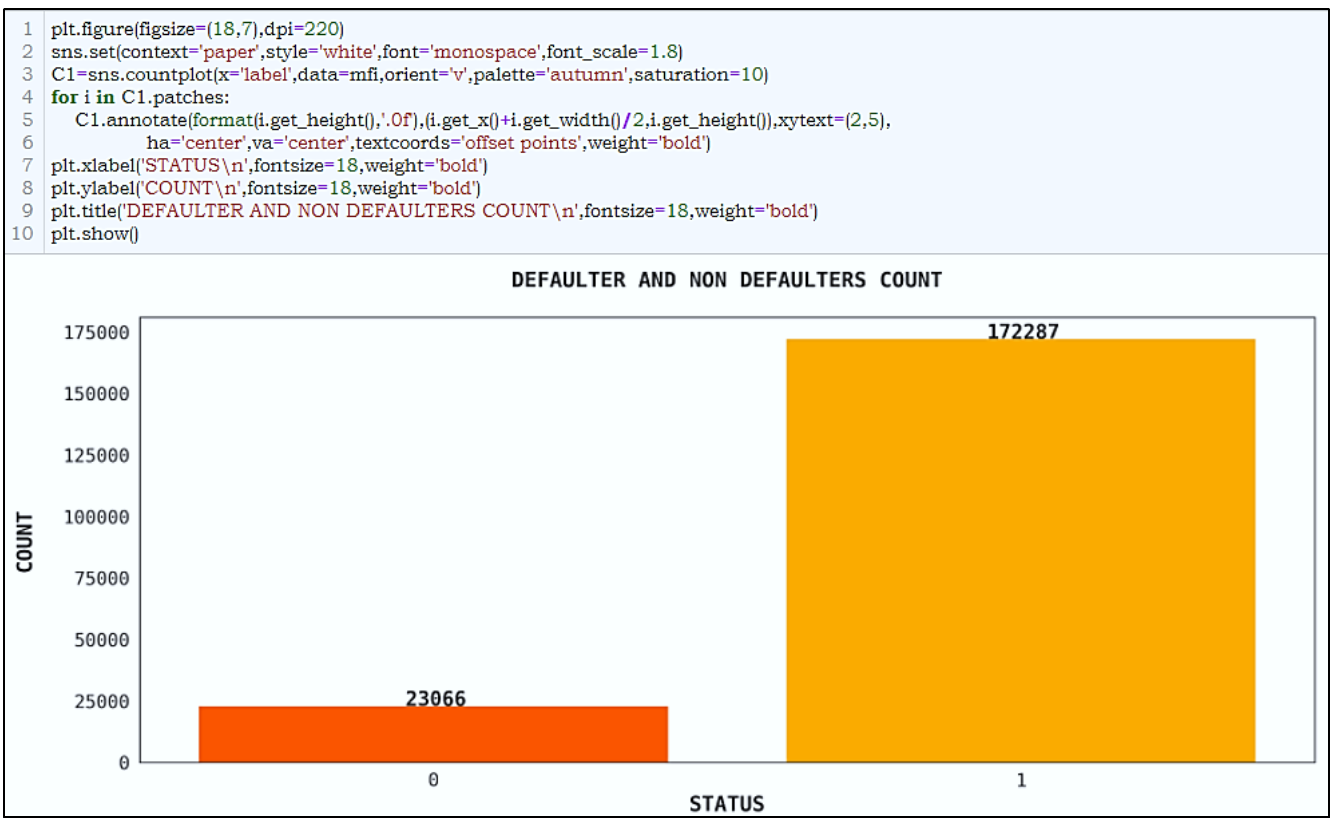

Fig.16: DEFAULTERS AND NON-DEFAULTERS

```
plt.figure(figsize=(18,7),dpi=220)
sns.set(context='paper',style='white',font='monospace',font_scale=1.8)
sns.barplot(x='label',y='aon',data=mfi,ci=None,palette="bone",saturation=35)
plt.xlabel('STATUS\n',fontsize=18,weight='bold')
plt.ylabel('AGE ON NETWORK\n',fontsize=18,weight='bold')
plt.title('AGE ON NETWORKS \n',fontsize=18,weight='bold')
plt.show()
```

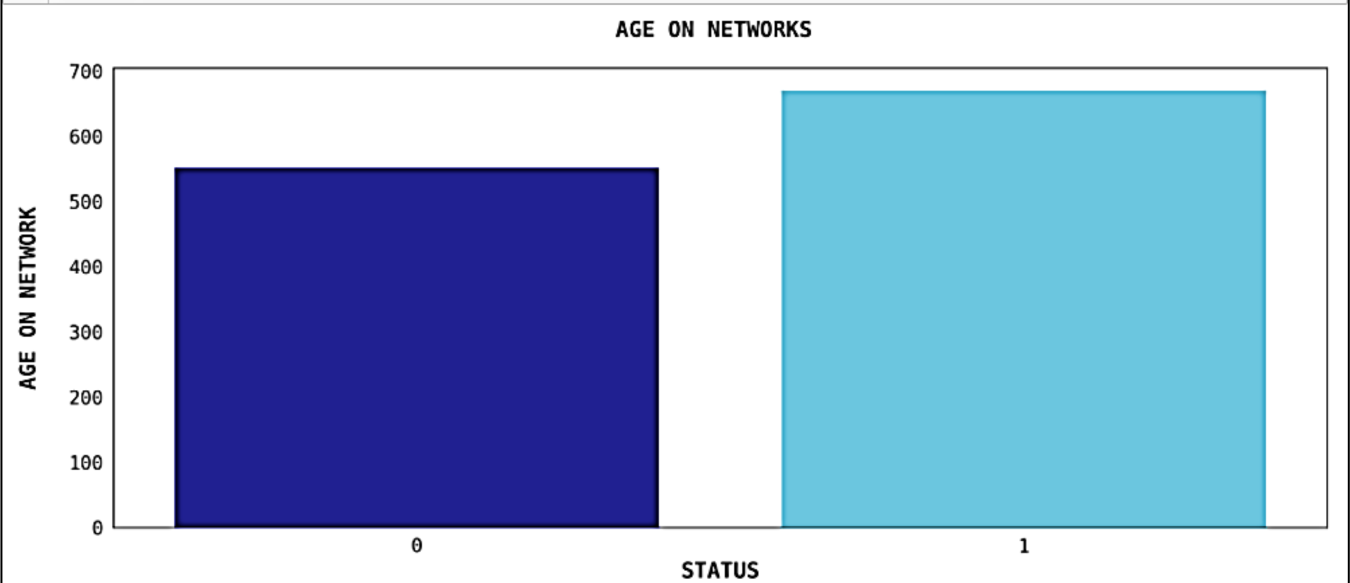

Fig.17: AGE ON NETWORK

```
plt.figure(figsize=(18,7),dpi=220)
sns.set(context='paper',style='white',font='monospace',font_scale=1.8)
sns.barplot(x='label',y='rental30',data=mfi,ci=None,palette="ocean_r",saturation=35)
plt.xlabel('STATUS\n',fontsize=18,weight='bold')
plt.ylabel('RENTAL30\n',fontsize=18,weight='bold')
plt.title('AVERAGE MAIN ACCOUNT BALANCE OF LAST 30 DAYS FOR DEFAULTER AND NON DEFAULTERS \n',fontsize=18,weight='bold')
plt.show()
```

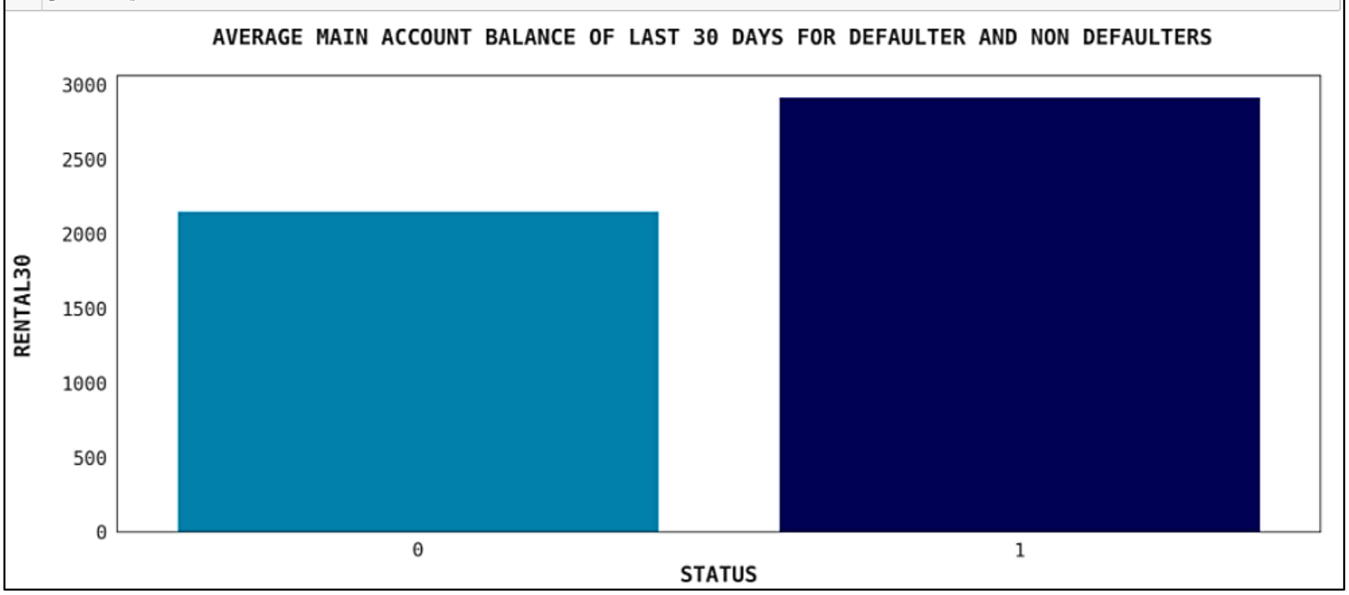

Fig.18: MAIN ACCOUNT BALANCE OF LAST 30 DAYS

```python
plt.figure(figsize=(18,7),dpi=220)
sns.set(context='paper',style='white',font='monospace',font_scale=1.8)
sns.barplot(x='label',y='amnt_loans30',data=mfi,ci=None,palette="plasma",saturation=35)
plt.xlabel('STATUS\n',fontsize=18,weight='bold')
plt.ylabel('AVG LOAN TAKEN IN LAST 30 DAYS\n',fontsize=18,weight='bold')
plt.title('LOAN TAKEN IN LAST 30 DAYS BY DEFAULTER AND NON DEFAULTERS \n',fontsize=18,weight='bold')
plt.show()
```

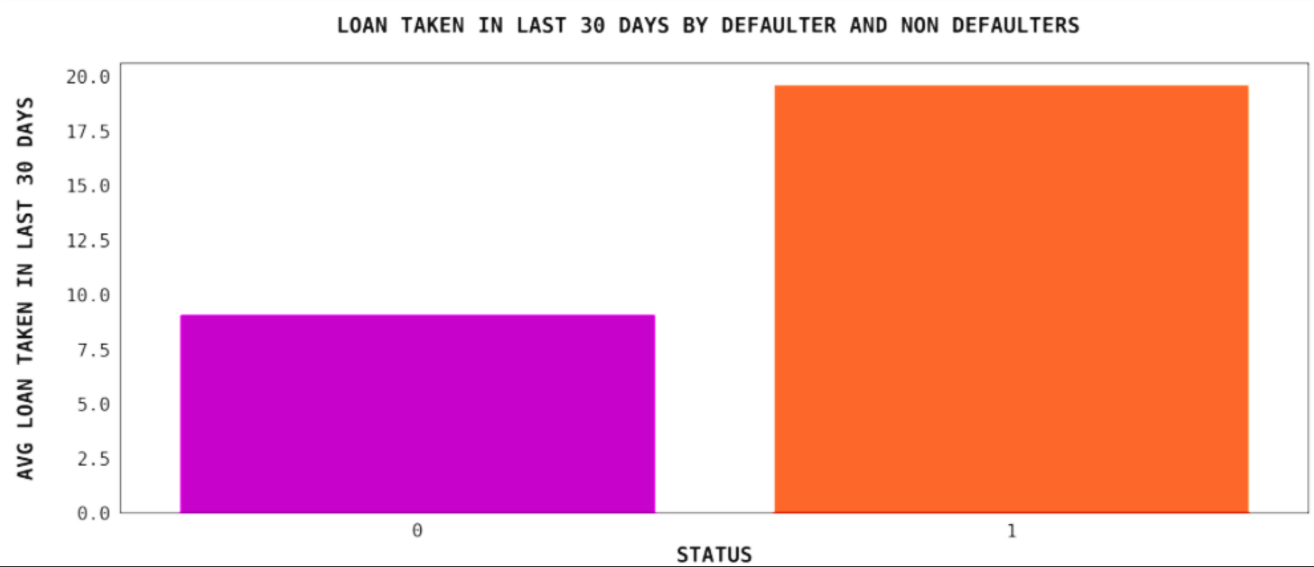

Fig.19: LOAN TAKEN IN 30 DAYS

```python
plt.figure(figsize=(18,7),dpi=220)
sns.set(context='paper',style='white',font='monospace',font_scale=1.8)
sns.barplot(x='label',y='fr_ma_rech90',data=mfi,ci=None,palette="Oranges",saturation=35)
plt.xlabel('STATUS\n',fontsize=18,weight='bold')
plt.ylabel('FREQUENCY OF RECHARGE IN LAST 90 DAYS\n',fontsize=18,weight='bold')
plt.title('FREQUENCY OF RECHARGE IN LAST 90 DAYS BY DEFAULTER AND NON DEFAULTERS \n',fontsize=18,weight='bold')
plt.show()
```

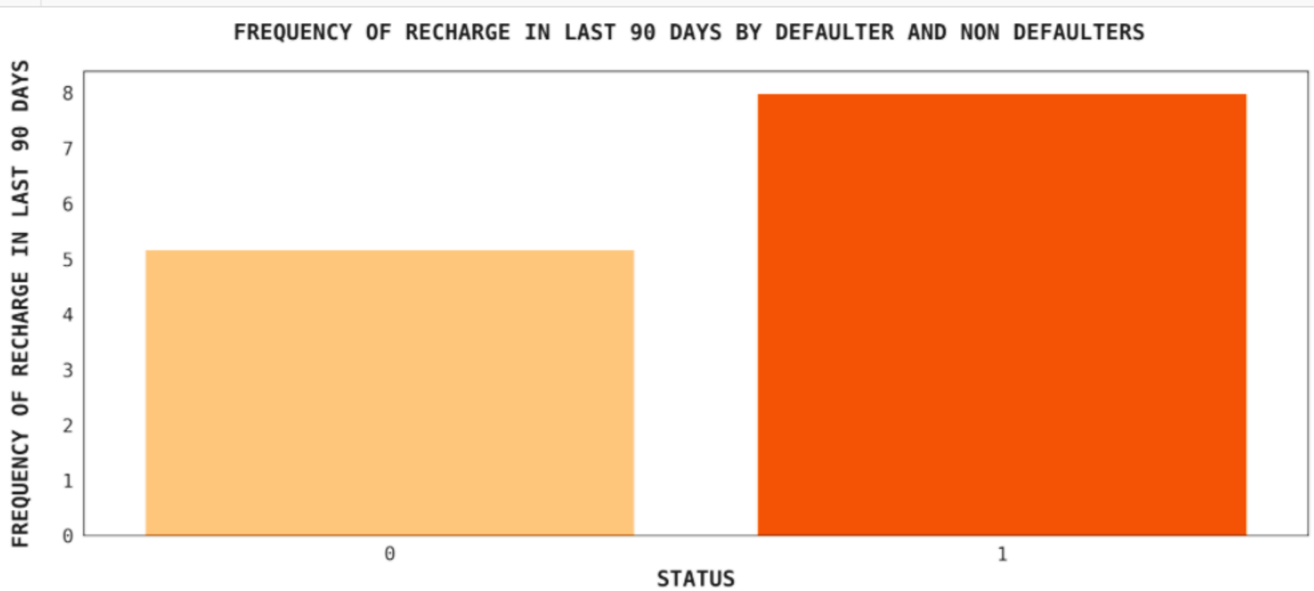

Fig.20: RECHARGE FREQUENCY IN 90 DAYS

```
sns.set_context('talk',font_scale=1.9)
sns.set(context='notebook',style='whitegrid',palette='dark',font='monospace',font_scale=1.3)
plt.figure(figsize=(12,3),dpi=120)
mfi.hist(figsize=(80,80),grid=True)
plt.show()
```

Fig.21: FREQUENCY DISTRIBUTION

```python
plt.figure(figsize=(70,60))
sns.set(style='darkgrid')
sns.set_context('talk',font_scale=1.1)
for i in range(len(mficn)):
    plt.subplot(7,5,i+1)
    sns.distplot(mfi[mficn[i]], color='dodgerblue', bins=80,kde=True,rug=True)
    plt.tight_layout()
```

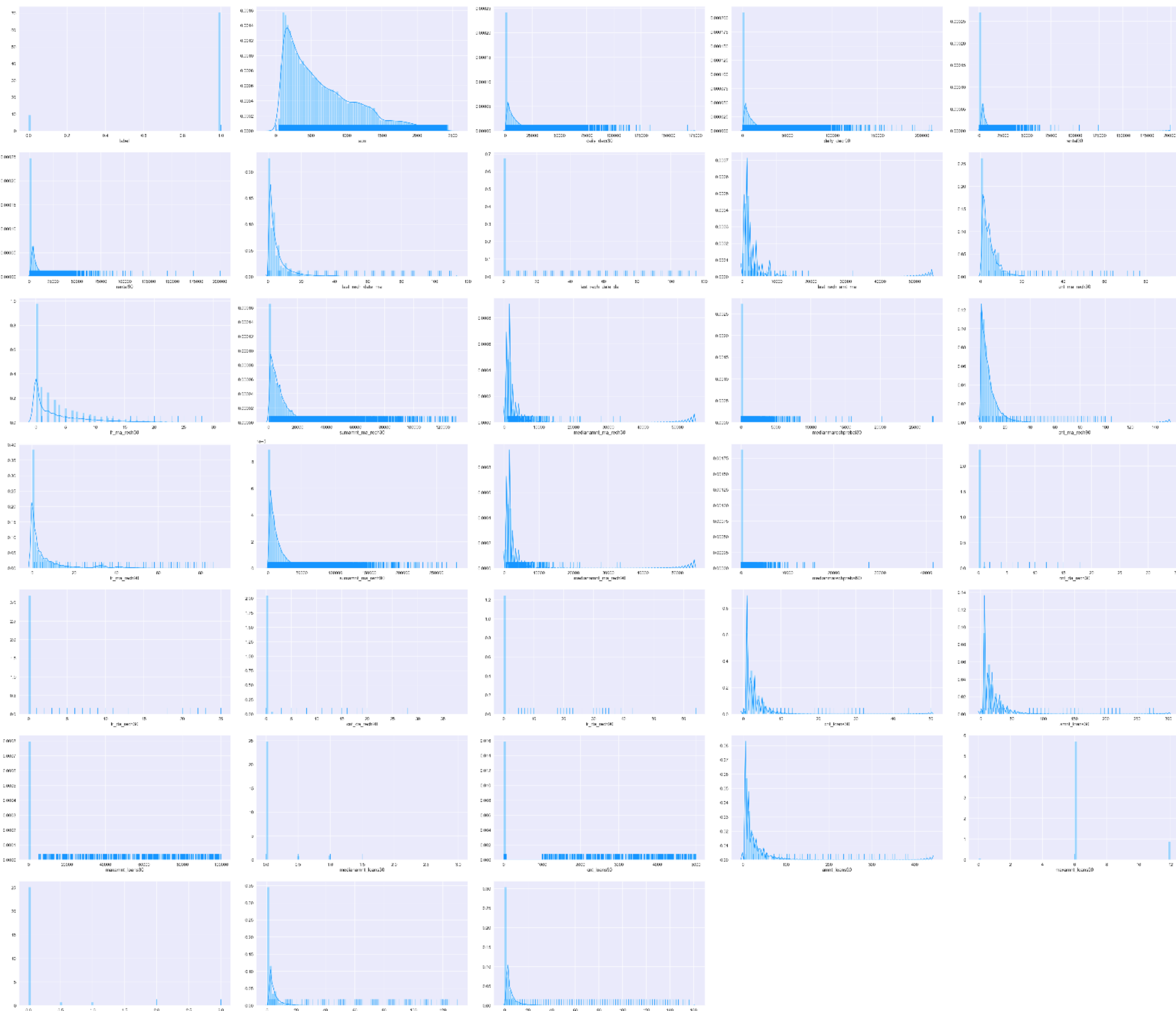

Fig.22: PROBABILITY DENSITY

3.5 INTERPRETATION

Basing on the result obtained XgBoost Classifier have performed well and has given better result as compared to other models. Here the accuracy score is 91.91%, Precision is 92.64%, AUC value is 70% and after cross validation the accuracy becomes 91.62%. As this model have all the desired characters and performance so XgBoost has been selected as final model and it will be saved using joblib library.

```
joblib.dump(XGB,'MFI.obj')
```

```
['MFI.obj']
```

Fig.23: MODEL SAVING

CHAPTER-4

CONCLUSION

4.1 KEY FINDINGS

From the above analysis the below mentioned results were achieved which depicts the chances and conditions of a user being a defaulter

- Users who are new to the network have a chance of being a defaulter.
- Users maintaining a low average of main account balance for last 30 days have a chance of being a defaulter.
- Users with low frequency of recharge in last 90 days have a chance of being a defaulter.
- Users who have taken low amount of loan in last 30 days have a chance of being a defaulter.

4.2 LEARNING OUTCOMES OF THE STUDY

The dataset obtained was imbalanced and have values which are absurd. Few values were negative while few are too high to believe. So those value were treated using proper conditions.

The count plots, bar plot, heatmap gave a vivid idea to understand the behavioral patterns which differentiate the defaulters and non-defaulters.

In this classification problem the model created using XgBoost classifier worked efficiently as compared to other models.

4.3 LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

There were certain limitations found in this dataset.

- The metadata provided could have been more precise.

- The ‘pcircle’ column have shown a single network circle which put a limitation to the analysis.
- Dataset have a large number of absurd values which must be taken into consideration.