Project on Micro-Credit Defaulter Model

Salman Pradhan

Submitted by:

INTRODUCTION

- Business Problem Framing

  We have to build a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan

- Motivation for the Problem Undertaken

  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.

- Mathematical/ Analytical Modeling of the Problem
  - In the month of august Defaulter customer is less compared to june and july.
  - Columns like cnt_ma_rech30,cnt_ma_rech90,sumamnt_ma_rech90 and sumamnt_ma_rech30 are highly positive corelated with our output column.
  - Though the dataset is imbalanced,we have to make dataset balanced by using under sampling or over sampling.After using both both sampling ,we have to see in which sampling my accuracy is good for the datset that is considered as our best model to predict output.

- Data Sources and their formats
  - The dataset is in csv format,so we have to load dataset by using pd.read_csv ()method.

```
df=pd.read_csv("C:/Users/Lenovo/Downloads/Micro Credit Project/Data file.csv")
df
pd.pandas.set_option('display.max_columns',41)
df
```

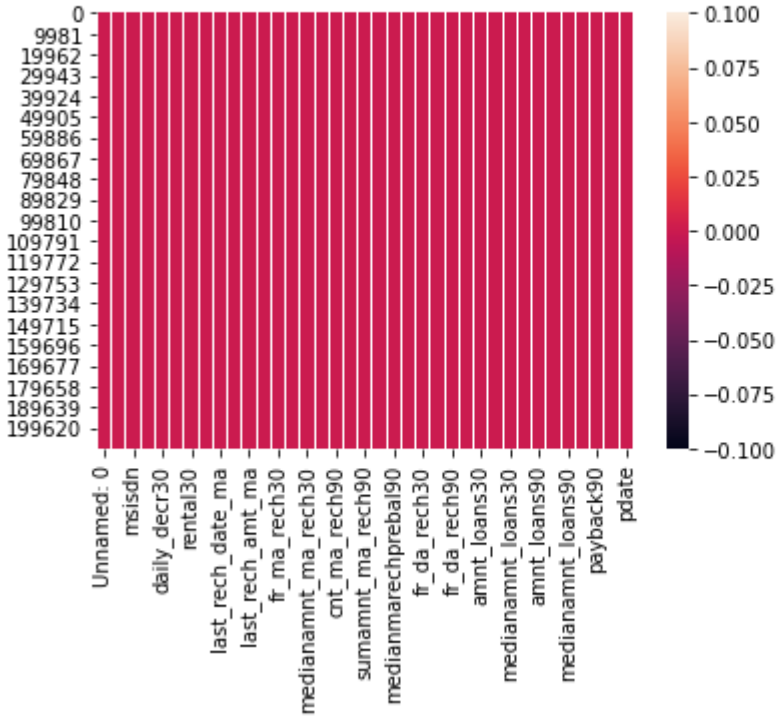
	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	
...	...	...	...	...	...	...	...	...	...	...	...	
209588	209589	1	22758I85348	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	4048	
209589	209590	1	95583I84455	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	773	
209590	209591	1	28556I85350	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	0.0	1539	
209591	209592	1	59712I82733	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	38.0	773	
209592	209593	1	65061I85339	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	0.0	7526	

209593 rows × 37 columns

- This dataset has 209593 rows and 37 features.

```
sns.heatmap(df.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dd90b62fa0>
```



- This dataset has no null values.

- Data Preprocessing Done

- Some columns have standard deviation is more compared to their mean that is nothing but outliers.We have to remove it.

```
#statistical analysis
df.describe()
```

	Unnamed: 0	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_r
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	20
mean	104797.000000	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921	
std	60504.431823	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430	
min	1.000000	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	
25%	52399.000000	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000	
50%	104797.000000	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000	
75%	157195.000000	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000	
max	209593.000000	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	5

More features have standard deviation is more comparing to mean,so they are outliers.We have to remove outlier later.

Some skewness is present in the dataset,we have to remove skewness by power_transform method.

Some Skewness is present in columns,so we have to remove it's skewness by using power_transform methhod

```
from sklearn.preprocessing import power_transform
for i in x_res[coltouse].columns:
    x_res[i]=power_transform(x_res[[i]])
```

- State the set of assumptions (if any) related to the problem under consideration
  - Though dataset has output approximately 88% is 1 and 12% is 0.So
  - We have to use either under_sampling or over_sampling to make
  - the dataset balanced.
  - I use both to see in which sampling technique my accuracy is good.

- Hardware and Software Requirements and Tools Used

Libraries required to solve micro credit loan use case are:

1.Numpy-

- It is used to create an array,matrix.
- It is used to perform operation on matrices and linear algebra operations..

2.Pandas

- It is used to read dataset.
- It is used to manipulate in the dataset

3.Scikit_learn

- It is used for model building to predict output for the dataset.
- It is used to divide dataset in training and test training

4.Matplotlib

- It is a visualization library for data analysis.

5.Seaborn

- It is used for visualization of dataset .

6.imblearn

- Though the dataset is imbalanced,imblearn library is used to balancing the output feature's unique value.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
  - I am dividing dataset in 80% training and 20% for testing purposes.Though the dataset is imbalanced I am using both undersampling and oversampling method to check the accuracy and auc_roc score.But oversampling method is giving good accuracy.In oversampling all information is not

going to loss but we have to add more data to make dataset balanced.So I choose oversampling method.

- Though output is either ‘0’ or ‘1’ so it is binary classification problem.

- Testing of Identified Approaches (Algorithms)
  - Algorithms used for the training and testing
  - LogisticRegression
  - DecisionTreeClassifier
  - GaussianNB
  - XGBClassifier
  - RandomForestClassifier
  - AdaBoostClassifier
  - GradientBoosting Classifier
- Run and Evaluate selected models

```
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
y_pred=dt.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
0.9584183251557418
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     36340
           1       1.00      0.92      0.96     36216

    accuracy                           0.96     72556
   macro avg       0.96      0.96      0.96     72556
weighted avg       0.96      0.96      0.96     72556

[[36285    55]
 [ 2962 33254]]
```

```
lr=LogisticRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
C:\Users\Lenovo\anacon\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (s
tatus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
0.7659738684602239
              precision    recall  f1-score   support

           0       0.76      0.77      0.77     36340
           1       0.77      0.76      0.76     36216

    accuracy                           0.77     72556
   macro avg       0.77      0.77      0.77     72556
weighted avg       0.77      0.77      0.77     72556

[[28116  8224]
 [ 8756 27460]]
```

```
gn=GaussianNB()
gn.fit(x_train,y_train)
y_pred=gn.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
0.7475605049892496
              precision    recall  f1-score   support

           0       0.73      0.78      0.76     36340
           1       0.76      0.71      0.74     36216

    accuracy                           0.75     72556
   macro avg       0.75      0.75      0.75     72556
weighted avg       0.75      0.75      0.75     72556

[[28372  7968]
 [10348 25868]]
```

```
xg=XGBClassifier()
xg.fit(x_train,y_train)
y_pred=xg.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
C:\Users\Lenovo\anacon\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[12:49:47] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
0.8668614587353217
              precision    recall  f1-score   support

           0       0.86      0.88      0.87     36340
           1       0.87      0.86      0.87     36216

    accuracy                           0.87     72556
   macro avg       0.87      0.87      0.87     72556
weighted avg       0.87      0.87      0.87     72556

[[31866  4474]
 [ 5186 31030]]
```

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
y_pred=rf.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
0.9778929378686807
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     36340
           1       1.00      0.96      0.98     36216

    accuracy                           0.98     72556
   macro avg       0.98      0.98      0.98     72556
weighted avg       0.98      0.98      0.98     72556

[[36288    52]
 [ 1552 34664]]
```

```
ad=AdaBoostClassifier()
ad.fit(x_train,y_train)
y_pred=ad.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
0.8145156844368487
              precision    recall  f1-score   support

           0       0.80      0.84      0.82     36340
           1       0.83      0.79      0.81     36216

    accuracy                           0.81     72556
   macro avg       0.82      0.81      0.81     72556
weighted avg       0.82      0.81      0.81     72556

[[30413  5927]
 [ 7531 28685]]
```

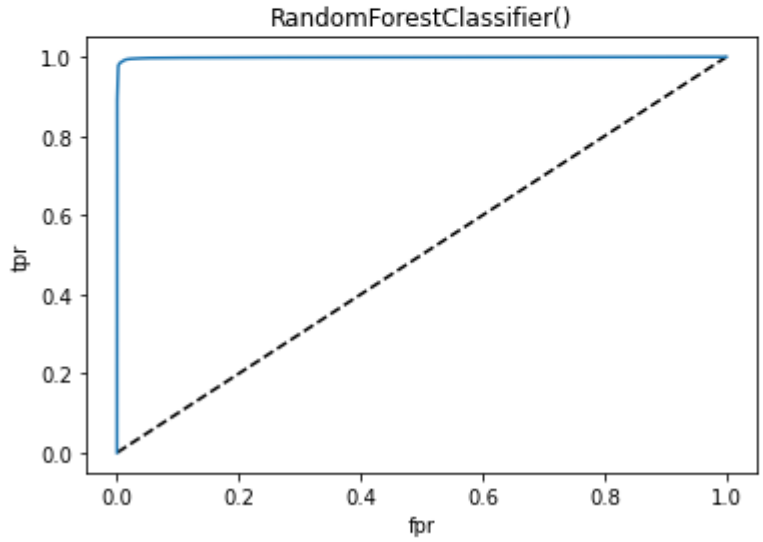
```
gb=GradientBoostingClassifier()
gb.fit(x_train,y_train)
y_pred=gb.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
0.8258862120293291
              precision    recall  f1-score   support

           0       0.82      0.83      0.83     36340
           1       0.83      0.82      0.82     36216

    accuracy                           0.83     72556
   macro avg       0.83      0.83      0.83     72556
weighted avg       0.83      0.83      0.83     72556

[[30328  6012]
 [ 6621 29595]]
```

- After using all above algorithm we conclude that RandomForestClassifier is the best model for the dataset as accuracy score,f1score,recall and precision is above 96%

- Using roc_auc _curve we also get same finding that RandomForestClassifier is best algorithm compared to other algorithm as roc_auc_score of RandomForestClassifier is 99.87% which is higher compared to others.

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
y_pred_prob=rf.predict_proba(x_test)[:,1]
fpr,tpr,thresholds=roc_curve(y_test,y_pred_prob)
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr)
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title(rf)
plt.show()
print(roc_auc_score(y_test,y_pred_prob))
```

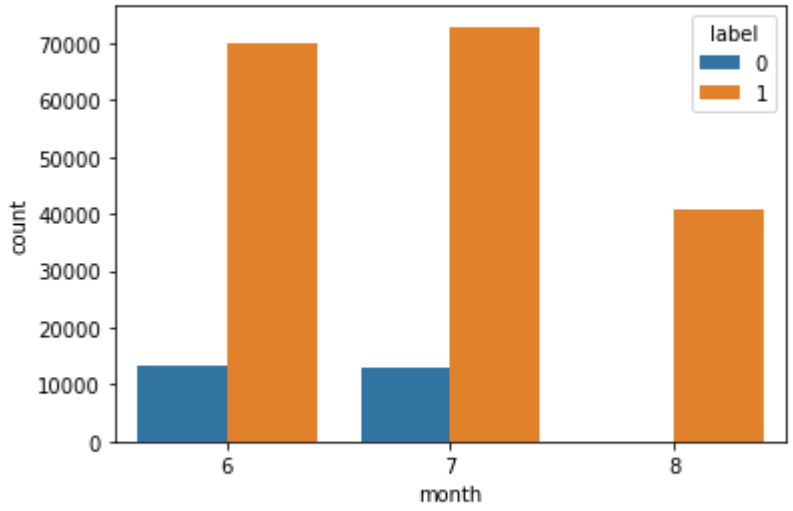



```
0.9987068025559115
```

- Visualizations

```
sns.countplot(x='month',hue='label',data=df)
```

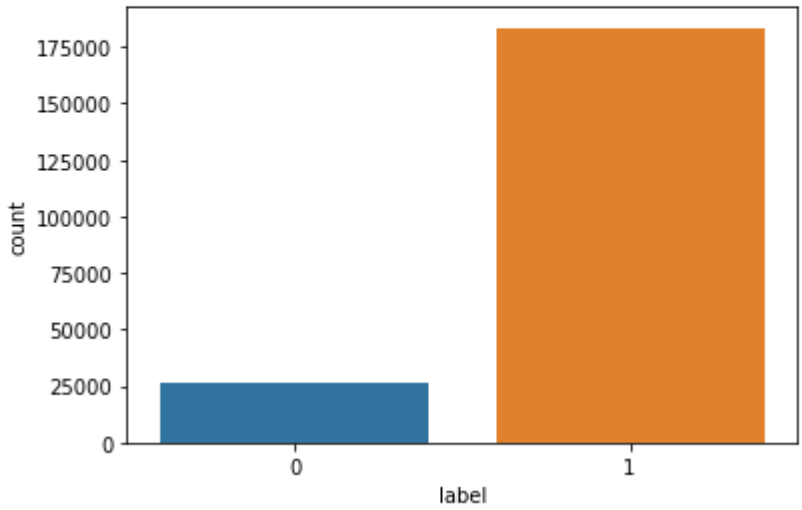
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dd9184c9a0>
```



In August month defaulter customer is less compared to june and july.

```
sns.countplot(x='label',data=df)
print(df['label'].value_counts())
```

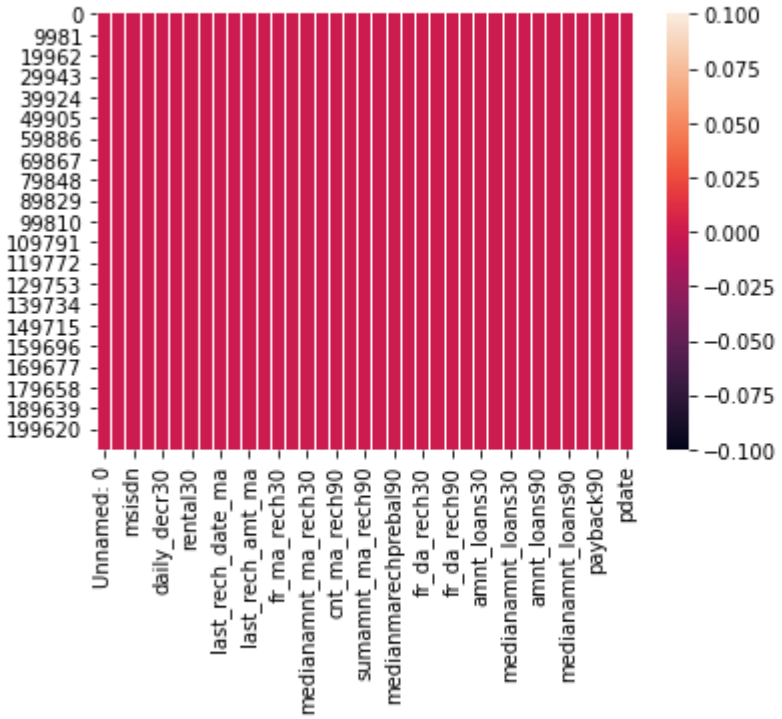
```
1    183431
0     26162
Name: label, dtype: int64
```



- It clearly shows that dataset is imbalanced.

```
sns.heatmap(df.isnull())
```

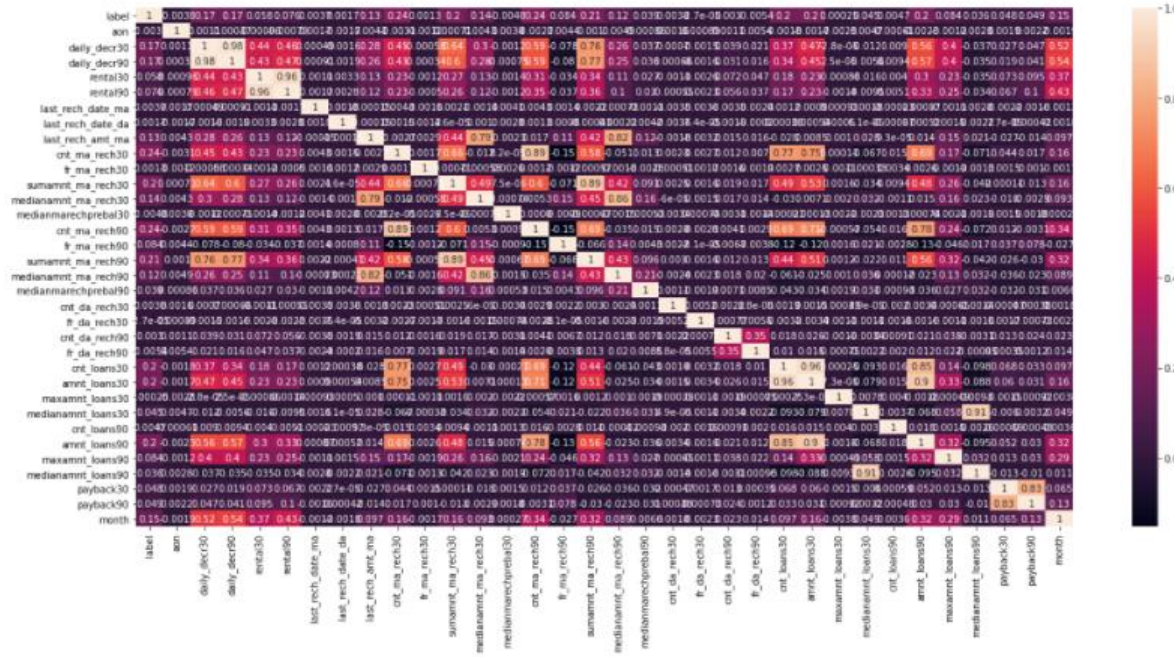
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dd90b62fa0>
```



➢ No Null values are present in the dataset

- Interpretation of the Results

```
plt.figure(figsize=(22,10))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In [54]:
```
df.corr()['label'].sort_values(ascending=False)
```

Out[54]:
```
label                    1.000000
cnt_ma_rech30            0.237331
cnt_ma_rech90            0.236392
sumamnt_ma_rech90        0.205793
sumamnt_ma_rech30        0.202828
amnt_loans90             0.199788
amnt_loans30             0.197272
cnt_loans30              0.196283
daily_decr30             0.168298
daily_decr90             0.166150
month                    0.154949
medianamnt_ma_rech30     0.141490
last_rech_amt_ma         0.131804
medianamnt_ma_rech90     0.120855
fr_ma_rech90             0.084385
maxamnt_loans90          0.084144
rental90                 0.075521
rental30                 0.058085
payback90                0.049183
payback30                0.048336
medianamnt_loans30       0.044589
medianmarechprebal90     0.039300
medianamnt_loans90       0.035747
cnt_loans90              0.004733
cnt_da_rech30            0.003827
last_rech_date_ma        0.003728
cnt_da_rech90            0.002999
last_rech_date_da        0.001711
fr_ma_rech30             0.001330
maxamnt_loans30          0.000248
fr_da_rech30            -0.000027
aon                     -0.003785
medianmarechprebal30    -0.004829
fr_da_rech90            -0.005418
Name: label, dtype: float64
```

- In above graph we found that there are which are approximately 0% correlated with output variable,so we may drop that columns.

- **CONCLUSION**
- Key Findings and Conclusions of the Study
  - In august month defaulter customers are less compared to june and july.
  - I remove approximately 1.1% outlier from the dataset which is pretty good not much of information is lost.
  - I have used 80% of dataset for traing and remaining 20% of data for testing purposes.

- Learning Outcomes of the Study in respect of Data Science
  - As its output value is 2,it is binary classification problem statement.I have used various various classification algorithm to predict output such as LogisticRegression,GaussianNB,XGBClassifier,DecisionTreeClassifier,and ensemble technique like RandomForestClassifier,AdaBoostClassifier,GradientBoosting Classifier.After using all algorithm I found that RandomForestClassifier is best model for this dataset as it's recall,precision,f1_score and accuracy_score is above 96% which is high compared to other algorithms.