Detecting Micro Credit Loan Defaulter Using Machine Learning Model

Submitted by:

Aditya Paruleker

INTRODUCTION

- Business Problem Framing

Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients. We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days.

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.

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).

The sample data is provided to us from our client database. It is hereby given to you for this exercise. 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.

- Motivation for the Problem Undertaken

  Motivation behind this project is to help company detect the loan defaulter using data science knowledge. We achieve this goal by building a machine learning 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. In this case, Label ‘1’ indicates that the loan has been payed i.e. Non- defaulter, while, Label ‘0’ indicates that the loan has not been payed i.e. defaulter.

Analytical Problem Framing

- Data Sources and their formats

  The dataset provided by the company, consists of 2,09,593 rows and 37 columns. There are 4 categorical features while the remaining 33 are numerical features. The target feature is encoded.

- Preprocessing Done

a) Null Value Analysis

```
Unnamed: 0              0
label                   0
msisdn                  0
aon                     0
daily_decr30            0
daily_decr90            0
rental30                0
rental90                0
last_rech_date_ma       0
last_rech_date_da       0
last_rech_amt_ma        0
cnt_ma_rech30           0
fr_ma_rech30            0
sumamnt_ma_rech30       0
medianamnt_ma_rech30    0
medianmarechprebal30    0
cnt_ma_rech90           0
fr_ma_rech90            0
sumamnt_ma_rech90       0
medianamnt_ma_rech90    0
medianmarechprebal90    0
cnt_da_rech30           0
fr_da_rech30            0
cnt_da_rech90           0
fr_da_rech90            0
cnt_loans30             0
amnt_loans30            0
maxamnt_loans30         0
medianamnt_loans30      0
cnt_loans90             0
amnt_loans90            0
maxamnt_loans90         0
medianamnt_loans90      0
payback30               0
payback90               0
pcircle                 0
pdate                   0
dtype: int64
```

No null values are detected in the dataset. It is observed that 8.8% of msisdn feature that contains mobile data have duplicate values. We drop these values. Also the data in pdate column which contains date of loan, is separated into new columns (year, month and day column).

b) Outlier Analysis

Outliers are detected in all the features. Usually removing the outliers from dataset is not always considered to be safe and simply removing all them might affect our models incase there were also outliers in the test dataset.

Since the data loss after removing outliers is more than 20%, we avoid removing outliers altogether.

c) Separating Target and Input features

Before proceeding with further preprocessing, the input features labelled x are separated from target features labelled y. We drop column year, pcircle that contribute single value.

d) Encoding Target Variable

The categorical features are encoded.

e) Skew Analysis

It is observed that all the numeric features are skewed and hence we use quantile transformation to reduce within range of +/-0.5.

```
msisdn                 -2.073036e-19
day                     2.007065e-01
month                   3.512926e-01
maxamnt_loans90         1.650198e+00
fr_ma_rech90            2.250443e+00
cnt_loans30             2.737584e+00
amnt_loans30            3.006644e+00
amnt_loans90            3.165962e+00
cnt_ma_rech30           3.471313e+00
medianamnt_ma_rech30    3.519213e+00
cnt_ma_rech90           3.558616e+00
medianamnt_ma_rech90    3.753115e+00
last_rech_amt_ma        3.830612e+00
daily_decr30            4.003019e+00
daily_decr90            4.301490e+00
medianamnt_loans30      4.470128e+00
rental90                4.530925e+00
rental30                4.676793e+00
medianamnt_loans90      4.774958e+00
sumamnt_ma_rech90       5.231693e+00
payback90               6.763241e+00
sumamnt_ma_rech30       7.134012e+00
payback30               8.193009e+00
aon                     1.036503e+01
medianmarechprebal30    1.467754e+01
fr_da_rech30            1.472861e+01
last_rech_date_da       1.478182e+01
fr_ma_rech30            1.482222e+01
last_rech_date_ma       1.485212e+01
cnt_loans90             1.671719e+01
maxamnt_loans30         1.771807e+01
cnt_da_rech30           1.774948e+01
cnt_da_rech90           2.839629e+01
fr_da_rech90            2.895985e+01
medianmarechprebal90    4.357636e+01
dtype: float64
```

Even after applying multiple transformations, there are few features whose skewness cannot be reduced any further.

```
msisdn                  -2.073036e-19
aon                     -1.571218e-04
daily_decr30             1.565198e-04
daily_decr90            -1.631770e-03
rental30                 1.783008e-03
rental90                 1.911681e-04
last_rech_date_ma        3.308857e-02
last_rech_date_da        5.248084e+00
last_rech_amt_ma        -7.825671e-02
cnt_ma_rech30           -1.150997e-01
fr_ma_rech30            -3.424723e-02
sumamnt_ma_rech30       -1.175243e-01
medianamnt_ma_rech30    -1.230627e-01
medianmarechprebal30     2.133637e-02
cnt_ma_rech90           -8.210942e-02
fr_ma_rech90            -1.328452e-01
sumamnt_ma_rech90       -8.549747e-02
medianamnt_ma_rech90    -8.633389e-02
medianmarechprebal90     5.550692e-01
cnt_da_rech30            6.884901e+00
fr_da_rech30             1.143287e+01
cnt_da_rech90            5.961745e+00
fr_da_rech90             1.566311e+01
cnt_loans30              2.355911e-01
amnt_loans30             1.904437e-01
maxamnt_loans30          1.527042e+00
medianamnt_loans30       3.382142e+00
cnt_loans90              1.846763e-01
amnt_loans90             1.564270e-01
maxamnt_loans90          1.894048e+00
medianamnt_loans90       3.684994e+00
payback30                3.304652e-01
payback90                1.757430e-01
month                    3.512926e-01
day                      2.007065e-01
dtype: float64
```

f) Collinearity Analysis

In order to avoid two or more input features from feeding same information into the model, it is necessary to detect correlation between independent variables using Variance Inflation factor or VIF method. Below is the screenshot of dataset and its VIF value.

	variables	VIF
0	msisdn	3.981527
1	aon	4.141602
2	daily_decr30	1191.098404
3	daily_decr90	1282.152663
4	rental30	100.868612
5	rental90	114.090377
6	last_rech_date_ma	6.336924
7	last_rech_date_da	115.387052
8	last_rech_amt_ma	16.747997
9	cnt_ma_rech30	108.909037
10	fr_ma_rech30	5.398373
11	sumamnt_ma_rech30	132.534631
12	medianamnt_ma_rech30	32.297651
13	medianmarechprebal30	12.341845
14	cnt_ma_rech90	130.884116
15	fr_ma_rech90	5.379706

16	sumamnt_ma_rech90	145.236869
17	medianamnt_ma_rech90	36.629323
18	medianmarechprebal90	8.900125
19	cnt_da_rech30	1.867800
20	fr_da_rech30	1.240571
21	cnt_da_rech90	4.727121
22	fr_da_rech90	1.473900
23	cnt_loans30	309.939215
24	amnt_loans30	366.727742
25	maxamnt_loans30	96.772315
26	medianamnt_loans30	5.229985
27	cnt_loans90	178.455254
28	amnt_loans90	249.712992
29	maxamnt_loans90	101.280043
30	medianamnt_loans90	5.188455
31	payback30	11.363891
32	payback90	12.054601
33	month	146.037413
34	day	4.190146

In above case, we decided to drop only those features ('daily_decr30','rental30','amnt_loans30','cnt_loans90') that are found to have a very high VIF value and also contribute less towards target variable.

g) **Oversampling**

The dataset is imbalanced. Label '1' has approximately 87.5% records, while, label '0' has approximately 12.5% records. The unbalanced dataset is balanced using SMOTE function.

h) Feature Scaling

Feature scaling is a technique frequent used in machine learning to generalize the data points so that the distance between them is smaller.

In this project, since the data doesn’t have Gaussian distribution, we use normalization technique to rescale the data.
The **Min-Max Normalization** uses distribution value between 0 and 1 to re-scales all the feature values.

- Data Inputs- Logic- Output Relationships

Correlation is a measure of how strongly or weakly connected two features are in a dataset. Below table show the percentage of correlation each input feature has with the output column. Higher the percentage stronger is the bond, and vice versa. Positive value indicates that the relationship between the two variables move in the same direction whereas negative value indicates that the relationship between two variables move in opposite direction.

Below is the percentage of correlation of each feature in dataset with target column named label.

```
Percent correlation of following

label                   100.0
aon                      -0.0
daily_decr30             17.0
daily_decr90             17.0
rental30                  6.0
rental90                  8.0
last_rech_date_ma         0.0
last_rech_date_da         0.0
last_rech_amt_ma         13.0
cnt_ma_rech30            24.0
fr_ma_rech30              0.0
sumamnt_ma_rech30        20.0
medianamnt_ma_rech30     14.0
medianmarechprebal30     -0.0
cnt_ma_rech90            24.0
fr_ma_rech90              8.0
sumamnt_ma_rech90        21.0
medianamnt_ma_rech90     12.0
medianmarechprebal90      4.0
cnt_da_rech30             0.0
fr_da_rech30             -0.0
cnt_da_rech90             0.0
fr_da_rech90             -1.0
cnt_loans30              20.0
amnt_loans30             20.0
maxamnt_loans30           0.0
medianamnt_loans30        4.0
cnt_loans90               0.0
amnt_loans90             20.0
maxamnt_loans90           8.0
medianamnt_loans90        4.0
payback30                 5.0
payback90                 5.0
Name: label, dtype: float64
```

- Hardware and Software Requirements and Tools Used

  Following libraries were used in this project,

  Pandas
  Created by Wes McKinney in 2008, this python library is widely used for data manipulation, analyzing and cleaning of data. Apart from this, it also helps in finding correlation between columns and in deleting rows.

  Numpy
  Created by Travis Oliphant in 2005, this python library provides an array object called ndarray i.e. up to 50x faster than traditional python lists. It was has functions can be used in linear algebra, matrices etc

  Seaborn
  Seaborn is a high level interface based data visualization library that uses matplotlib library underneath the working.

  Matplotlib
  Unlike saeborn, matplotlib is a low level data visualization python library. Majority of its function lies in the submodule named pyplot

  Scikit-Learn
  It provides tools for classification, regression, clustering and dimensionality reduction through its interface in python.

  Pickle
  Used for serializing (pickling) and de-serializing (unpickling) python object structure so that the data can be easily transferred from system to system and then stored in a file.

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

  In this study several machine learning models were tested and their results were compared before selecting the model that gave best result among all.

  Logistic Regression
  Using logistic approach for modelling, this supervised machine learning model aims to solve classification problems by predicting categorical outcomes, unlike linear regression that predicts a continuous outcome.

  KNeighbor Classifier
  The K-NN algorithm used in this classification stores all the available data and classifies a new data point based on the similarity. It is non parametric since it does not make any underlying assumption.

  Support Vector Classifier
  The SVC approach finds a hyperplane that creates a boundary between two classes of data to classify them.

  Multinomial Naive Bayes Classifier
  The algorithm first guesses the tag of a text using the Bayes theorem and then calculates each tag's likelihood for a given sample and lastly outputs the tag with the greatest chance.

  Ensemble Methods
  Ensemble technique uses several base estimators of machine learning model to predict the output. This results are then combined to improve the robustness of model over single estimator.

Randomforest Classifier

Using multiple decision trees as base, the model randomly performs the dataset sampling over the rows and features such that it form sample datasets for every model.

GradientBoost Classifier

The model after calculating the residual i.e. the difference between actual value and predicted target value; trains the weak model for mapping the features to that residual.

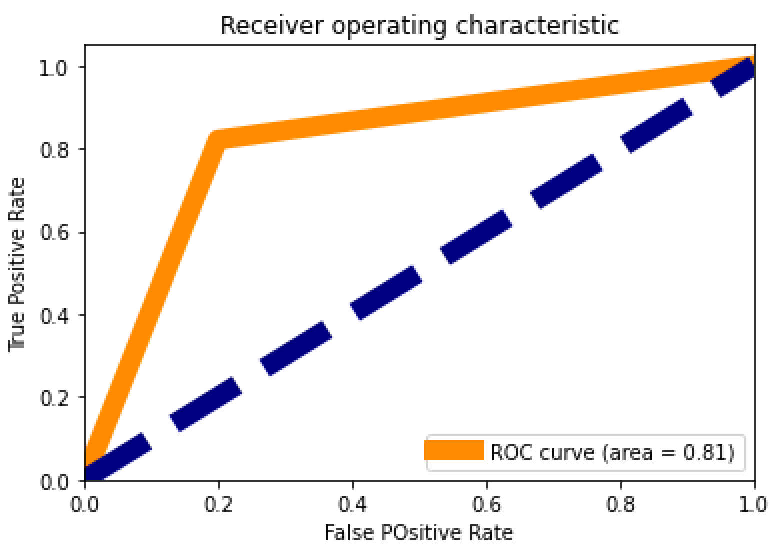
- Run and Evaluate models

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import r2_score,accuracy_score,classificatio
n_report,auc,roc_curve
from sklearn.model_selection import train_test_split,cross_val_sc
ore
from sklearn.metrics import log_loss


lg = LogisticRegression()

x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,
random_state=42)
lg.fit(X,y)
pred_train=lg.predict(x_train)
pred_test=lg.predict(x_test)
print("Train Accuracy : ",round(accuracy_score(y_train,pred_train
)*100,2))
print("Test Accuracy : ",round(accuracy_score(y_test,pred_test)*1
00,2))
print("cv score : ", round(cv_score.mean()*100,2))
logloss = log_loss(y_test, lg.predict_proba(x_test))
logloss
print("Classification Report:\n",classification_report(y_test,pre
d_test))
```

Output

```
Accuracy of training model : 80.96
Accuracy of test data : 81.02
cv score :  80.82
Log loss:  0.4341528764197971

Classification report for test data
              precision    recall  f1-score   support

           0       0.80      0.83      0.81     32066
           1       0.82      0.79      0.81     32088

    accuracy                           0.81     64154
   macro avg       0.81      0.81      0.81     64154
weighted avg       0.81      0.81      0.81     64154

Classification report for training data
              precision    recall  f1-score   support

           0       0.80      0.83      0.81    128317
           1       0.82      0.79      0.81    128295

    accuracy                           0.81    256612
   macro avg       0.81      0.81      0.81    256612
weighted avg       0.81      0.81      0.81    256612

Confusion Matrix
 [[26473  5593]
 [ 6581 25507]]
```



Model Selection User Defined Function

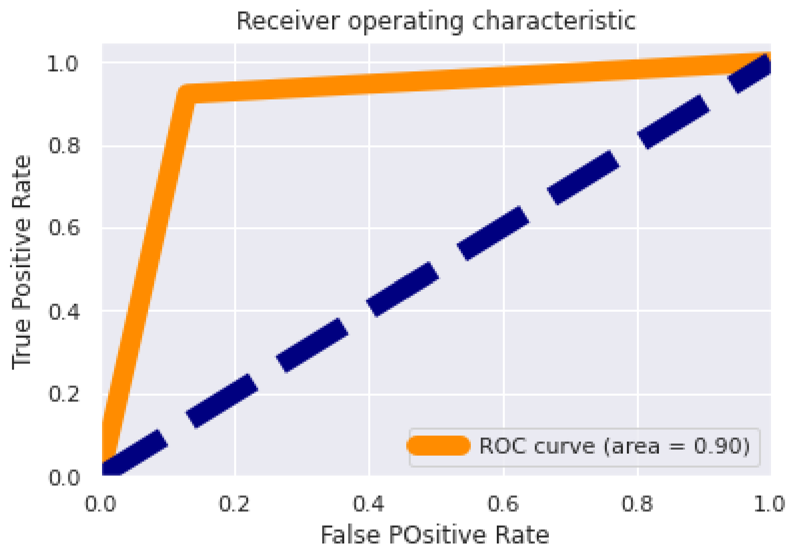
```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve,auc

def model_selection(algorithm_instance,x_train,y_train,x_test,y_t
est):
algorithm_instance.fit(x_train,y_train)
model_pred_train=algorithm_instance.predict(x_train)
 model_pred_test=algorithm_instance.predict(x_test)
 print("Accuracy of training model :",round(accuracy_score(y_train
 ,model_pred_train)*100,2))
 print("Accuracy of test data :",round(accuracy_score(y_test,model
 _pred_test)*100,2))

 cv_score=cross_val_score(algorithm_instance,X,y,cv=5)
 print("cv score : ", round(cv_score.mean()*100,2))
print("\nClassification report for test data\n",classification_repo
 rt(y_test,model_pred_test))
print("Classification report for training data\n",classification_re
 port(y_train,model_pred_train))
print("Confusion Matrix\n",confusion_matrix(y_test,model_pred_test))
print("\n")

fpr, tpr, thresholds = roc_curve(model_pred_test,y_test)
roc_auc= auc(fpr,tpr)
plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label='ROC curve (area =
%0.2f)' % roc_auc)
plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0,1.05])
plt.xlabel("False POsitive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic")
plt.legend(loc='lower right')
plt.show()
```

K Neighbour Classifier

```
from sklearn.neighbors import KNeighborsClassifier
k=KNeighborsClassifier()
model_selection(k,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 93.18
Accuracy of test data : 89.47
cv score is  89.14

Classification report for test data
              precision    recall  f1-score   support

           0       0.87      0.93      0.90     32066
           1       0.92      0.86      0.89     32088

    accuracy                           0.89     64154
   macro avg       0.90      0.89      0.89     64154
weighted avg       0.90      0.89      0.89     64154

Classification report for training data
              precision    recall  f1-score   support

           0       0.91      0.96      0.93    128317
           1       0.96      0.90      0.93    128295

    accuracy                           0.93    256612
   macro avg       0.93      0.93      0.93    256612
weighted avg       0.93      0.93      0.93    256612

Confusion Matrix
 [[29740  2326]
 [ 4431 27657]]
```

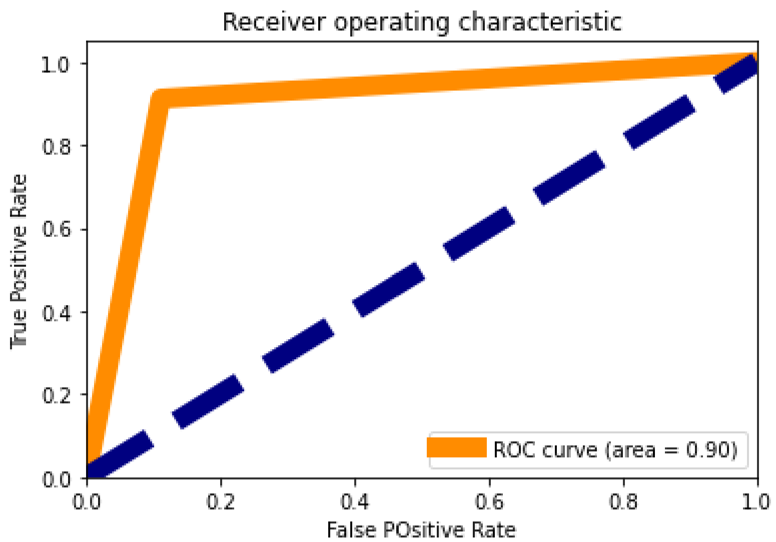


Support Vector Classifier

```
from sklearn import svm
s=svm.SVC()
model_selection(s,x_train,y_train,x_test,y_test)
```

Output

```
 Accuracy of training model : 89.02
 Accuracy of test data : 88.93

Classification report for test data
              precision    recall  f1-score   support

           0       0.89      0.91      0.90     32066
           1       0.91      0.89      0.90     32088

    accuracy                           0.90     64154
   macro avg       0.90      0.90      0.90     64154
weighted avg       0.90      0.90      0.90     64154

Classification report for training data
              precision    recall  f1-score   support

           0       0.89      0.91      0.90    128317
           1       0.91      0.89      0.90    128295

    accuracy                           0.90    256612
   macro avg       0.90      0.90      0.90    256612
weighted avg       0.90      0.90      0.90    256612

Confusion Matrix
 [[29308  2758]
 [ 3668 28420]]
```

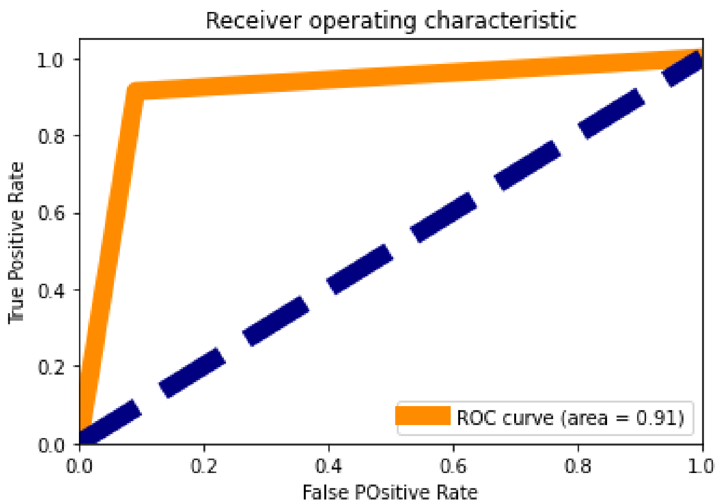


Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
model_selection(dtc,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 100.0
Accuracy of test data : 91.1
cv score :  88.71
Log loss:  3.0746477537072496

Classification report for test data
              precision    recall  f1-score   support

           0       0.91      0.91      0.91     32066
           1       0.91      0.91      0.91     32088

    accuracy                           0.91     64154
   macro avg       0.91      0.91      0.91     64154
weighted avg       0.91      0.91      0.91     64154

Classification report for training data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    128317
           1       1.00      1.00      1.00    128295

    accuracy                           1.00    256612
   macro avg       1.00      1.00      1.00    256612
weighted avg       1.00      1.00      1.00    256612

Confusion Matrix
 [[29309  2757]
 [ 2954 29134]]
```

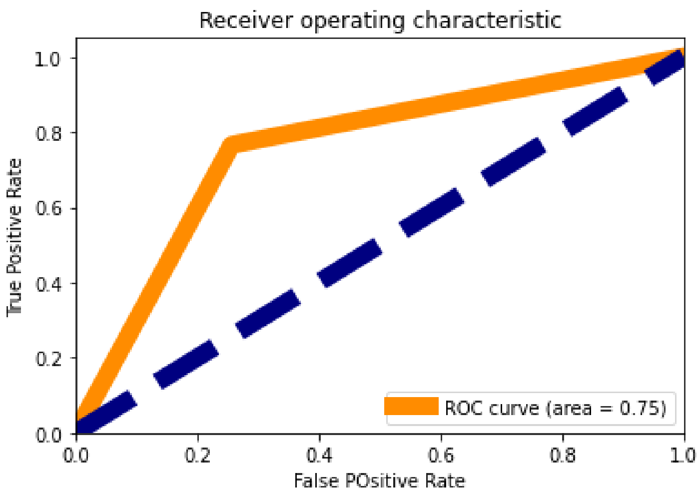


Multinomial Naïve Bayes Classifier

```
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB()
model_selection(mnb,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 75.35
Accuracy of test data : 75.4
cv score :  75.35
Log loss:  0.5229032403278038

Classification report for test data
              precision    recall  f1-score   support

           0       0.74      0.78      0.76     32066
           1       0.77      0.73      0.75     32088

    accuracy                           0.75     64154
   macro avg       0.75      0.75      0.75     64154
weighted avg       0.75      0.75      0.75     64154

Classification report for training data
              precision    recall  f1-score   support

           0       0.74      0.78      0.76    128317
           1       0.77      0.73      0.75    128295

    accuracy                           0.75    256612
   macro avg       0.75      0.75      0.75    256612
weighted avg       0.75      0.75      0.75    256612

Confusion Matrix
 [[24872  7194]
 [ 8588 23500]]
```



Randomforest Classifier

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

parameter={'criterion':['gini', 'entropy', 'log_loss'],
     'max_features' : [None,"sqrt","log2"],
      'class_weight':['balanced',' balanced_subsample'],
      'n_estimators':range(0,100,50)}
rf=RandomForestClassifier()
clf=GridSearchCV(rf,parameter)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

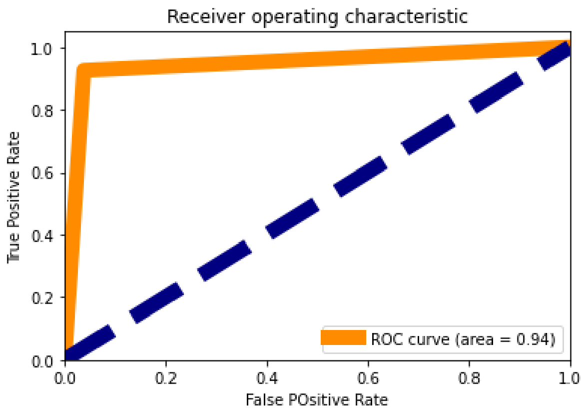
```
{'class_weight': 'balanced', 'criterion': 'gini', 'max_features': 'sqrt', 'n_estimators': 50}
```

```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n_estimators=50, class_weight= 'balanced
',criterion='entropy', max_features= None)
      model_selection(rf,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 99.99
Accuracy of test data : 94.26
cv score :  91.73
Log loss:  0.17015618121924608

Classification report for test data
              precision    recall  f1-score   support

           0       0.96      0.92      0.94     32066
           1       0.93      0.96      0.94     32088

    accuracy                           0.94     64154
   macro avg       0.94      0.94      0.94     64154
weighted avg       0.94      0.94      0.94     64154

Classification report for training data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    128317
           1       1.00      1.00      1.00    128295

    accuracy                           1.00    256612
   macro avg       1.00      1.00      1.00    256612
weighted avg       1.00      1.00      1.00    256612
```

```
Confusion Matrix
 [[29566  2500]
 [ 1182 30906]]
```



GradientBoost Regressor

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier

parameter={'loss':['log_loss', 'deviance', 'exponential'],
      'criterion':['friedman_mse', 'squared_error', 'mse'],
      'max_features':['auto', 'sqrt', 'log2'],
      'n_estimators':range(0,100,50)}

rf3=GradientBoostingClassifier()
clf=GridSearchCV(rf3,parameter)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

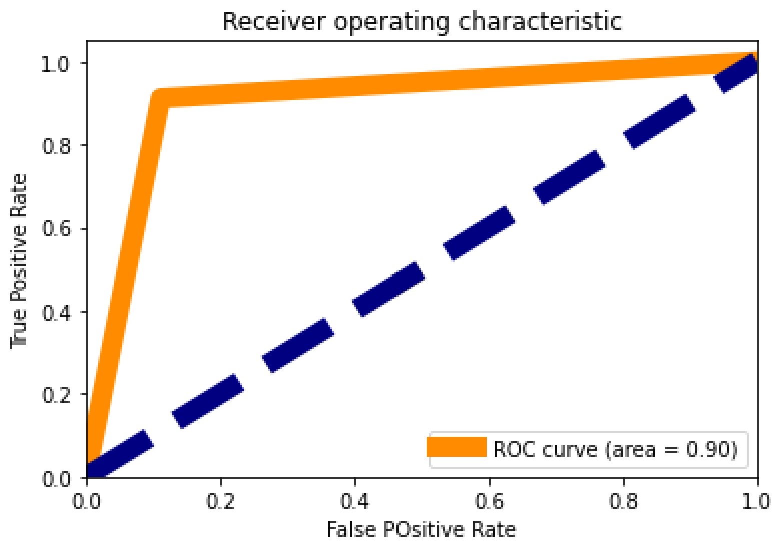
```
{'criterion': 'friedman_mse', 'loss': 'deviance', 'max_features': 'auto', 'n_estimators': 50}
```

```
rf3=GradientBoostingClassifier(criterion='friedman_mse', loss='deviance
', max_features= 'auto', n_estimators= 50)
model_selection(rf3,x_train,y_train,x_test,y_test)
```

Output

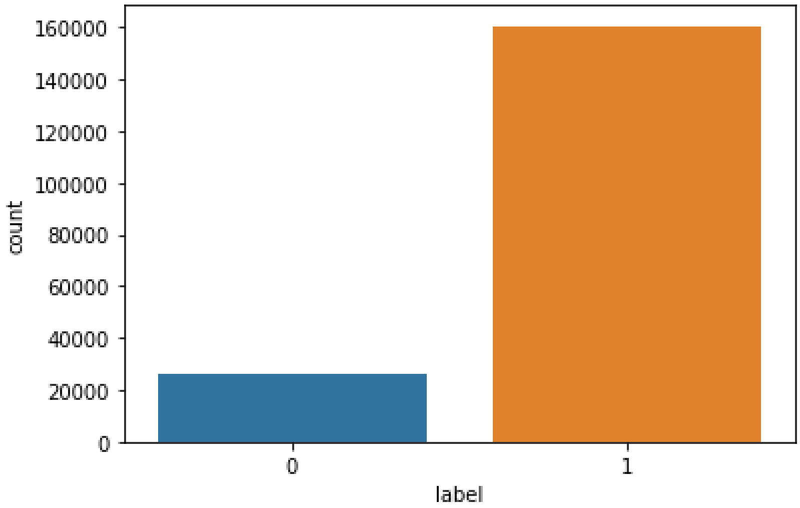
```
Accuracy of training model : 89.98
Accuracy of test data : 89.98
cv score :  89.01
Log loss:  0.27177032037491716
```

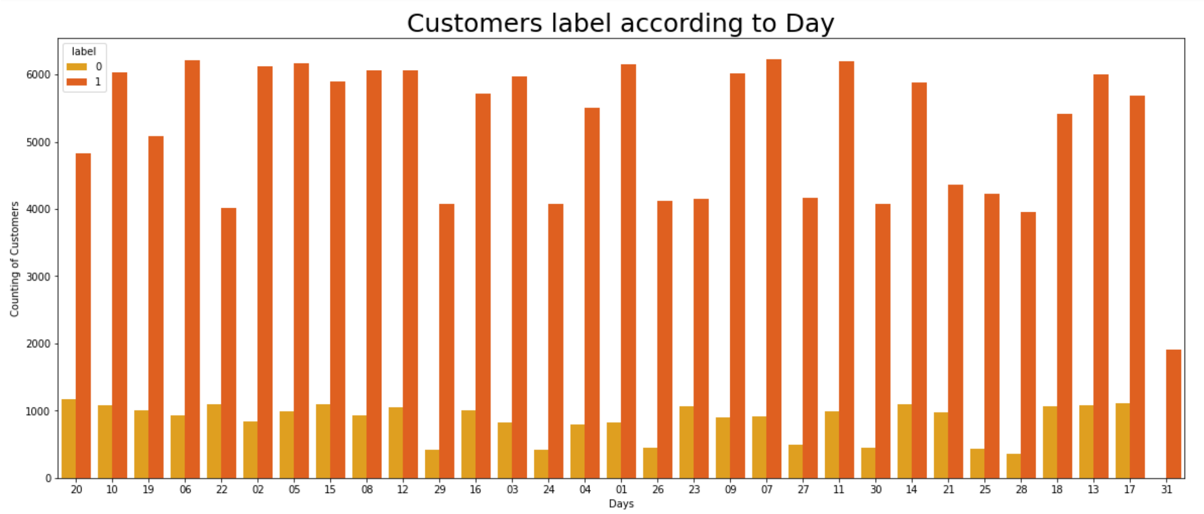
```
Classification report for test data
              precision    recall  f1-score   support

           0       0.89      0.91      0.90     32066
           1       0.91      0.89      0.90     32088

    accuracy                           0.90     64154
   macro avg       0.90      0.90      0.90     64154
weighted avg       0.90      0.90      0.90     64154

Classification report for training data
              precision    recall  f1-score   support

           0       0.89      0.91      0.90    128317
           1       0.91      0.89      0.90    128295

    accuracy                           0.90    256612
   macro avg       0.90      0.90      0.90    256612
weighted avg       0.90      0.90      0.90    256612

Confusion Matrix
 [[29308  2758]
 [ 3668 28420]]
```

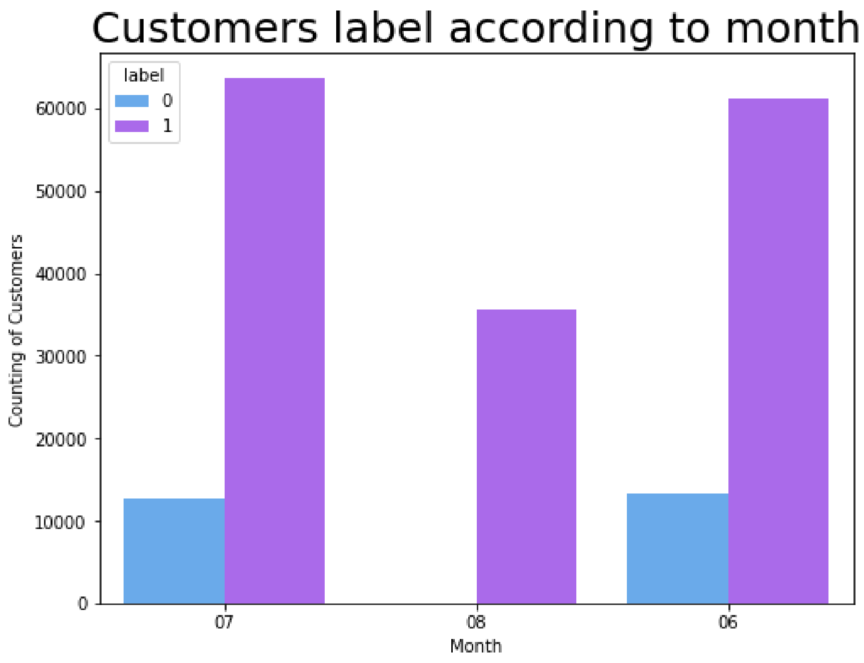


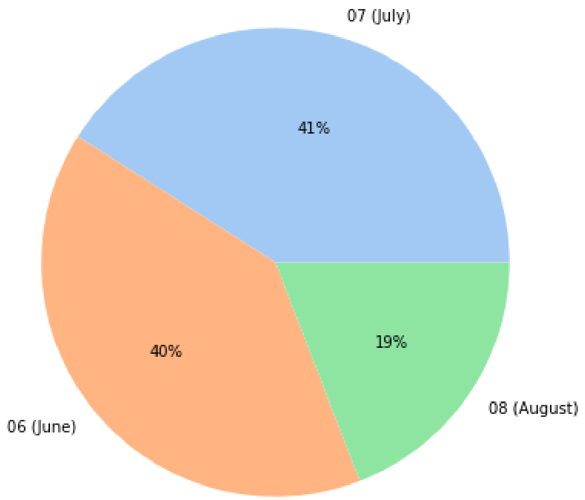
- Dataset Visualizations

```
1    160383
0     25860
Name: label, dtype: int64
```









- 183431 customers have paid loan while 26162 customers have not paid.
- In August, 2016 there are 0 customers who are in defaulters.
- June, July and August have more than +30k customer who have paid the loan
- Percentage of customers who have taken micro credit loan from June to August 2016 are as follows, 41% customer in July, 40% in June and 19% in August.

CONCLUSION

- Key Findings and Conclusions of the Study

  - In this study we found that random forest classifier performs slightly better than rest of the algorithm tested.
  - cnt_ma_rech30, cnt_ma_rech90, sumamnt_ma_rech30, sumamnt_ma_rech90, amnt_loans_30, amnt_loans_90, cnt_loans30, daily_decr30, daily_decr90 are the top features of the micro credit loan that highly impact the defaulter list among all the features in the dataset.
  - August is the only month with 0 defaulters.

Thus we were successfully able to detect the loan defaulter with high accuracy (94.26 %) and minimum error (log loss of 0.17).