Micro Credit Loan Defaulter Model

Submitted by:

Swati Pandey

ACKNOWLEDGMENT

I have taken efforts in this project. I am highly indebted to (Flip Robo Technologies) for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

A huge thanks to “Datatrained”, because of that I got this opportunity to work with FlipRobo. Lastly, I would like to thank my family, who have always been my support in every step of my life.

CONTENTS

INTRODUCTION

- Business Problem

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

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

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

We are working with one such client that is in 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).

- # Background of the Domain Problem

  To understand the domain problem, first we have to understand what is Micro Credit?

Microcredit is a common form of microfinance that involves an extremely small loan given to an individual to help them become self-employed or grow a small business. These borrowers tend to be low-income individuals, especially from less developed countries (LDCs). Microcredit is also known as "microlending" or "microloan."

We are working with one such client that is in Telecom Industry. For Telecom Industry, Micro Credit, value-added service is designed on the premise of “what the consumer needs”. It is a service which the telecom can offer to subscribers and resellers both. Through this service, telco can provide ease of access of airtime stock credit to their resellers when they run out of stock balance, without disrupting the normal flow of business. On the subscriber side, they can remain connected without having to recharge through a voucher or digital topup. This loan is disbursed in real-time and is calculated based on intelligent reseller/subscriber profiling.

When the subscriber or reseller (user) runs out of airtime credit and is unable to recharge, they can dial a USSD code to request for airtime credit. The system will perform user profiling and disburse loans. Next time the user recharges the account, the system will verify if there is any outstanding loan. The loan with a service fee is deducted from the user’s account.

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).

- # Review of The Literature:

  An attempt has been made in this report to review the available literature in the area of microfinance. Approaches to microfinance, issues related to measuring social impact versus profitability of MFIs, issue of sustainability, variables impacting sustainability, effect of regulations of profitability and impact assessment of MFIs have been summarized in the below report. We hope that the below report of literature will provide a platform for further research and help the industry to combine theory and practice to take microfinance forward and contribute to alleviating the poor from poverty.

- # Motivation for the Problem:

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.

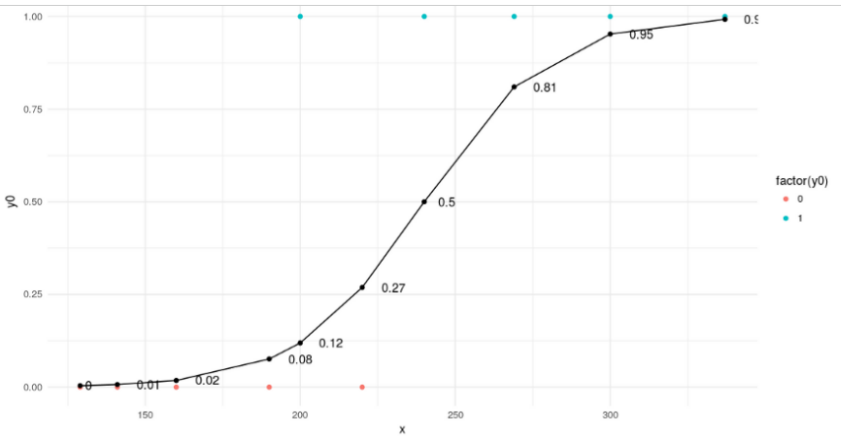
Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

  Since target or output variable for prediction is known, so this problem comes under Supervised learning, since label contains two values 0 and 1, so this a classification analysis problem.
  During data Analysis I have found that there are so many outliers present. Data is very much skewed. So, I have considered Random Forest as my final model to predict defaulter or non-Defaulter because Random Forest is insensitive to outliers.

  I have considered following algorithms for analytical modelling:

Logistic Regression: Logistic Regression is a Classification algorithm used to estimate the outcome of a categorical variable based on the independent variables. It predicts the probability of an event occurring by fitting the data to a logistic function. The coefficients of the independent variables in the logistic function are optimized by maximizing the likelihood function. A decision boundary is optimized such that the cost function is minimal. The cost function can be minimized by using Gradient Descent.



The sigmoid curve of Logistic Regression

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Cost Function of Logistic Regression

$$P = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\beta_3 X_3+\ldots+\beta_n X_n)}}$$

Decision Trees: Decision Trees are mostly used for Classification problems but they can also be used for Regression. In this algorithm, we divide the dataset into two or more homogeneous sets based on the attribute that divides the dataset most effectively. One of the methods to select the attribute that will split the dataset is by calculating Entropy and Information Gain. Entropy captures the amount of impurity in the variable. The Information Gain is the entropy of the parent node minus the sum of entropies of the child nodes. The attribute which provides maximum Information Gain is selected for splitting. We can also use the Gini Index as an impurity criterion to split the dataset. To prevent overfishing, we optimize the hyper-parameters of Decision Trees like max_features, min_samples_split, max_depth, etc.

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

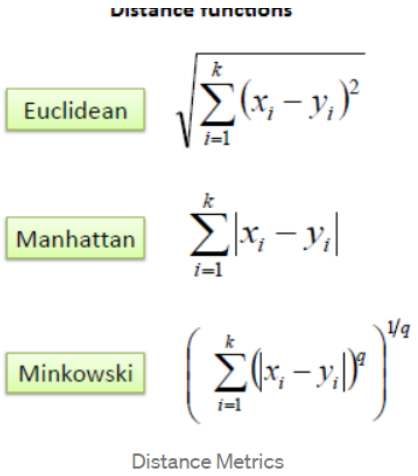
Entropy, c → No. of classes

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Gini

Random Forests: Random Forests consists of several Decision Trees which operate as an ensemble. An Ensemble consists of a group of models that are used to predict the outcome, rather than an individual model. In random forests, each decision tree predicts a class outcome and the class outcome with the number of votes becomes the prediction of random forests. For accurate predictions, the Decision Trees should be least correlated with each other. A couple of ways to ensure this is by Bagging and Feature Selection. Bagging is a technique for choosing random samples of observations from a dataset. Feature Selection allows the Decision Trees to model on a random subset of features only. This prevents individual trees to predict using the same features

k-NN (k- Nearest Neighbors) : This algorithm can also be used for both, Regression as well as Classification. The algorithm finds the k nearest neighbors of data points by computing its distance from all the data points. *The data point is assigned to the class with the highest number of points among the k neighbors (voting process). In the case of Regression, it calculates the mean of the k nearest neighbors.* Different distance metrics that can be used are Euclidean distance, Manhattan distance, Minkowski distance, etc. To eliminate the probability of a tie, the value of k must be an odd number. This algorithm is expensive computationally as the distance of each data point is calculated with every other data point.



Distance Metrics

AdaBoost Classifier : AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called **Decision Stumps.**

What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received.

$$Performance\ of\ the\ stump\ =\ \frac{1}{2}\log_e\left(\frac{1-Total\ Error}{Total\ Error}\right)$$

Data Sources and their formats:

Data is provided by client. Data Descriptions is as follows:

	A	B
1	Variable	Definition
2	label	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}
3	msisdn	mobile number of user
4	aon	age on cellular network in days
5	daily_decr30	Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
6	daily_decr90	Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
7	rental30	Average main account balance over last 30 days
8	rental90	Average main account balance over last 90 days
9	last_rech_date_ma	Number of days till last recharge of main account
10	last_rech_date_da	Number of days till last recharge of data account
11	last_rech_amt_ma	Amount of last recharge of main account (in Indonesian Rupiah)
12	cnt_ma_rech30	Number of times main account got recharged in last 30 days
13	fr_ma_rech30	Frequency of main account recharged in last 30 days
14	sumamnt_ma_rech30	Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
15	medianamnt_ma_rech30	Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
16	medianmarechprebal30	Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
17	cnt_ma_rech90	Number of times main account got recharged in last 90 days
18	fr_ma_rech90	Frequency of main account recharged in last 90 days
19	sumamnt_ma_rech90	Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)
20	medianamnt_ma_rech90	Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)
21	medianmarechprebal90	Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)
22	cnt_da_rech30	Number of times data account got recharged in last 30 days
23	fr_da_rech30	Frequency of data account recharged in last 30 days
24	cnt_da_rech90	Number of times data account got recharged in last 90 days
25	fr_da_rech90	Frequency of data account recharged in last 90 days
26	cnt_loans30	Number of loans taken by user in last 30 days
27	amnt_loans30	Total amount of loans taken by user in last 30 days
28	maxamnt_loans30	maximum amount of loan taken by the user in last 30 days
29	medianamnt_loans30	Median of amounts of loan taken by the user in last 30 days
30	cnt_loans90	Number of loans taken by user in last 90 days
31	amnt_loans90	Total amount of loans taken by user in last 90 days
32	maxamnt_loans90	maximum amount of loan taken by the user in last 90 days
33	medianamnt_loans90	Median of amounts of loan taken by the user in last 90 days
34	payback30	Average payback time in days over last 30 days
35	payback90	Average payback time in days over last 90 days
36	pcircle	telecom circle
37	pdate	date

- # Data Pre-processing

  Steps:

  a) I have imported all the necessary libraries.

  b) Loaded and read the csv file using pandas DataFrame.

  c) I have checked dataset size, columns datatype, unique values of each column datatype.

  d) I have checked for null values. There are not any null values present in dataset.

  e) During analysis I found that pcircle column has only one value, i.e., UPW, so dropped that column.

  f) ‘msisdn’ column, that is mobile number of the user has inappropriate value, found ‘I’ at 6th place, made it proper also changed data type to float.

```
df['msisdn'] =[ i.replace('I','') for i in df['msisdn']]
```

```
df['msisdn']
```

```
0         2140870789
1         7646270374
2         1794370372
3         5577370781
4         0381382730
             ...
209588    2275885348
209589    9558384455
209590    2855685350
209591    5971282733
209592    6506185339
Name: msisdn, Length: 209593, dtype: object
```

Mobile number is in string format,converting it into numerical format.

```
## Changing data type from string to float:-
df['msisdn'] = df['msisdn'].astype(float)
```

g) Extract day, month and year from pdate feature.

```
df['recharge_day'] =pd.DatetimeIndex(df['pdate']).day
```

```
df['recharge_month']=pd.DatetimeIndex(df['pdate']).month
```

```
df['recharge_year']=pd.DatetimeIndex(df['pdate']).year
```

h) Checked correlation using heatmap.

Correlation

```
plt.figure(figsize =(30,30))
sns.heatmap(df.corr(),annot = True,cmap ='RdYlGn')
plt.show()
```

i) Dropped unnecessary columns, which are very weakly correlated with label.

j) Detected outliers using **distplot** and **boxplot.**

k) Removed skewness using PowerTransformer, method is ‘yeo-johnson’.

l) Removed outlier using zscore.

```
from sklearn.preprocessing import PowerTransformer
skewed_features = ["daily_decr30","daily_decr90","rental30","rental90","last_rech_amt_ma","cnt_ma_rech30","sumamnt_ma_rech30
                   "medianamnt_ma_rech30","cnt_ma_rech90","fr_ma_rech90","sumamnt_ma_rech90","medianamnt_ma_rech90","medianma
                   "cnt_loans30","amnt_loans30","medianamnt_loans30","amnt_loans90","maxamnt_loans90","medianamnt_loans90","m
                   "payback30","payback90"]
scaler = PowerTransformer(method='yeo-johnson')
```

```
df[skewed_features] = scaler.fit_transform(df[skewed_features].values)
df[skewed_features].head()
```

	daily_decr30	daily_decr90	rental30	rental90	last_rech_amt_ma	cnt_ma_rech30	sumamnt_ma_rech30	medianamnt_ma_rech30	cnt_ma_rech90	fr_ma_rech
0	0.332156	0.299181	-0.580786	-0.568411	0.083417	-0.275617	-0.177906	0.208594	-0.577409	1.335
1	1.116021	1.045084	0.339736	0.148082	1.501277	-0.799355	0.219811	1.632404	-1.027906	-1.207
2	-0.007508	-0.028203	-0.366965	-0.409047	0.083417	-0.799355	-0.535362	0.208594	-1.027906	-1.207
3	-1.031486	-1.022743	-0.603059	-0.597731	-0.291052	-1.694157	-1.902189	-1.961729	-1.027906	-1.207
4	-0.682337	-0.680928	-0.310473	-0.364212	0.449550	0.992270	1.262660	0.575164	0.657800	-0.060

```
columns=df[['daily_decr30', 'daily_decr90',
       'medianmarechprebal90','medianamnt_loans90']]
from scipy.stats import zscore
z = np.abs(zscore(columns))
z
```

```
array([[0.332156  , 0.29918138, 0.32584055, 0.24827185],
       [1.11602076, 1.04508396, 0.00272174, 0.24827185],
       [0.00750823, 0.02820273, 0.02376962, 0.24827185],
       ...,
       [1.10045475, 1.03342362, 0.0753427 , 0.24827185],
       [1.13605673, 1.06838078, 0.18258877, 0.24827185],
       [0.52502128, 0.48715657, 1.26249783, 0.24827185]])
```

```
threshold = 3
np.where(z>3)
```

```
(array([    30,     35,     37, ..., 209538, 209565, 209587], dtype=int64),
 array([3, 3, 3, ..., 3, 3, 3], dtype=int64))
```

```
df_new =df[(z<3).all(axis=1)]
df_new
```

	label	daily_decr30	daily_decr90	rental30	rental90	last_rech_amt_ma	cnt_ma_rech30	sumamnt_ma_rech30	medianamnt_ma_rech30	cnt_ma_rech90
0	0	0.332156	0.299181	-0.580786	-0.568411	0.083417	-0.275617	-0.177906	0.208594	-0.577409
1	1	1.116021	1.045084	0.339736	0.148082	1.501277	-0.799355	0.219811	1.632404	-1.027906
2	1	-0.007508	-0.028203	-0.366965	-0.409047	0.083417	-0.799355	-0.535362	0.208594	-1.027906
3	1	-1.031486	-1.022743	-0.603059	-0.597731	-0.291052	-1.694157	-1.902189	-1.961729	-1.027906
4	1	-0.682337	-0.680928	-0.310473	-0.364212	0.449550	0.992270	1.262660	0.575164	0.657800

m) Checked multicollinearity using VIF (Variance Inflation Factor) and dropped columns, whose values are greater than 10.

n) Split data between features and label.

Separating Features and Target for Classification Problem:

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

o) Data is not balanced in target column, so using SMOTE, I oversampled data to balance.

```
from imblearn.over_sampling import SMOTE
SM = SMOTE()
x, y = SM.fit_resample(x,y)
```

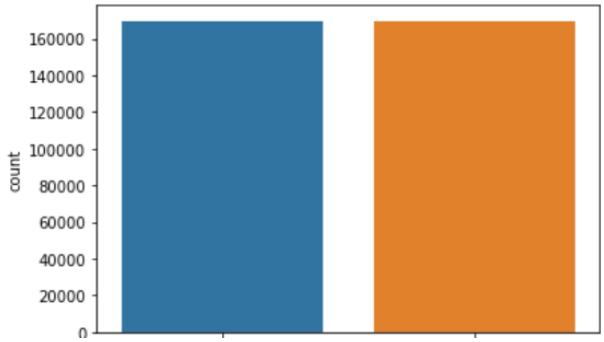
```
y.value_counts()
```

```
0    169667
1    169667
Name: label, dtype: int64
```

Now data are balanced in target column.

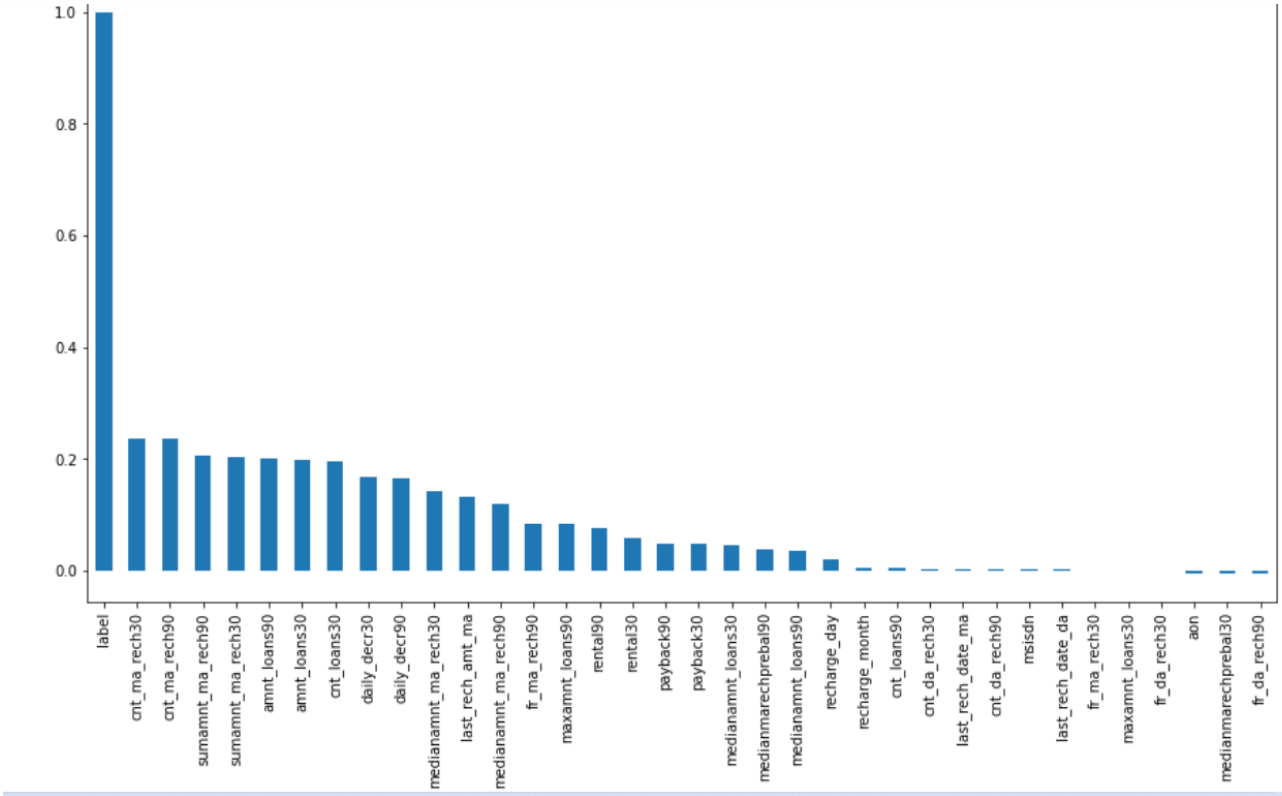
```
sns.countplot(y)
```

```
<AxesSubplot:xlabel='label', ylabel='count'>
```



p) Standardize data using StandardScaler.
q) Checked multicollinearity using Variance Inflation Factor, Dropped those columns, whose vif values are greater than 10.

Correlation of features with label:



Hardware and Software Requirements and Tools Used

Hardware:

- Process: Intel core i5 and above.
- RAM:16GB and above.
- SSD:250GB and above.

Software & Tools used:

- Python3.0,
- Jupyter Notebook,

Libraries:

- **Pandas**: pandas is a popular Python-based data analysis toolkit which can be imported using import pandas as pd. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a numpy matrix array. This makes pandas a trusted ally in data science and machine learning.

- **Numpy:** NumPy is the fundamental package for package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation,

sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

• **Seaborn**: Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas' data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

• **Matplotlibt**: matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some linesin a plotting area, decorates the plot with labels, etc.

- from sklearn.preprocessing import StandardScaler
- from sklearn.preprocessing import PowerTransformer
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.ensemble import RandomForestClassifier
- from sklearn.linear_model import LogisticRegression
- from sklearn.ensemble import AdaBoostClassifier
- from sklearn.metrics import classification_report
- from sklearn.metrics import accuracy_score
- from sklearn.model_selection import cross_val_score

Models/ Development and Evaluation

- Imported all necessary classifier models from desired sklearn package.
- To perform training and testing of given dataset, done Train Test Spilt.

Building Models

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
# Train Test Spilit
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=151)
```

```
x_train.shape,y_train.shape
```

```
((237533, 13), (237533,))
```

```
x_test.shape,y_test.shape
```

```
((101801, 13), (101801,))
```

Following are the models and classification report, which I have chosen to predict the label:

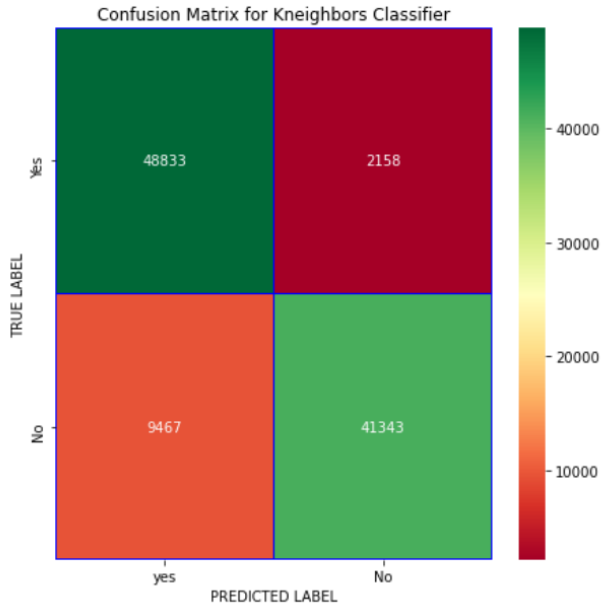
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, roc_auc_score, accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import plot_roc_curve

```

KNeighbors Classifier

```
knc = KNeighborsClassifier()
knc.fit(x_train,y_train)
predknc = knc.predict(x_test)

print(accuracy_score(y_test, predknc))
print(confusion_matrix(y_test, predknc))
print(classification_report(y_test,predknc))

```

```
0.8858066227247277
[[48833  2158]
 [ 9467 41343]]
              precision    recall  f1-score   support

           0       0.84      0.96      0.89     50991
           1       0.95      0.81      0.88     50810

    accuracy                           0.89    101801
   macro avg       0.89      0.89      0.89    101801
```

```
Text(0.5, 1.0, 'Confusion Matrix for Kneighbors Classifier')
```



True Positive-48833,True Negative-41343,False Positive-2158,False Negative-9467

DecisionTreeClassifier

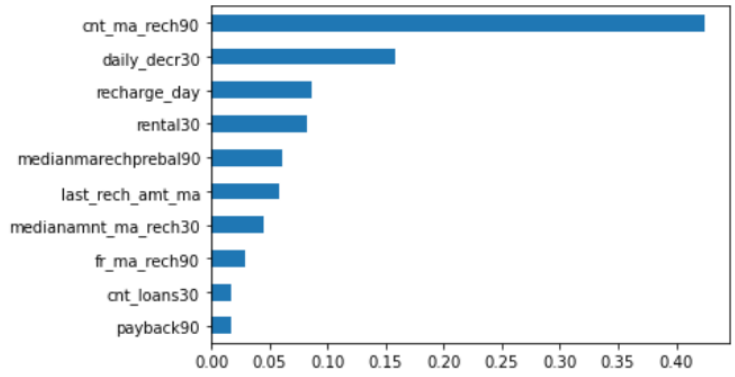
```
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)
preddtc = dtc.predict(x_test)

print(accuracy_score(y_test, preddtc))
print(confusion_matrix(y_test, preddtc))
print(classification_report(y_test,preddtc))

```

```
0.8972308719953636
[[46204  4787]
 [ 5675 45135]]
              precision    recall  f1-score   support

           0       0.89      0.91      0.90     50991
           1       0.90      0.89      0.90     50810

    accuracy                           0.90    101801
   macro avg       0.90      0.90      0.90    101801
weighted avg       0.90      0.90      0.90    101801
```

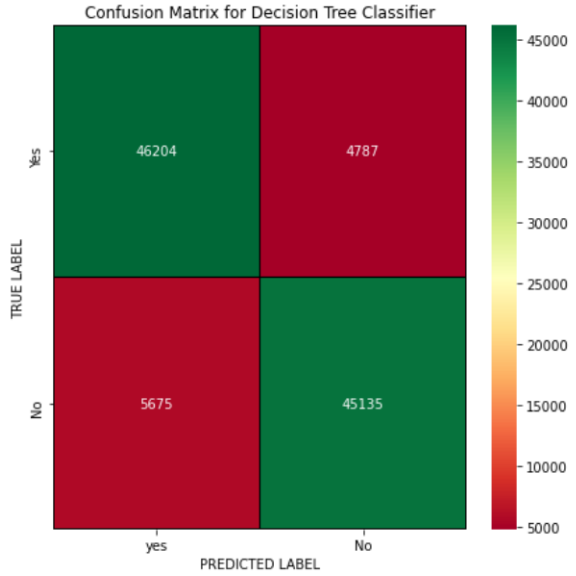
```
importances = dtc.feature_importances_
weights = pd.Series(importances,
                    index=x.columns.values)
weights.sort_values()[-10:].plot(kind = 'barh')

```

```
importances = dtc.feature_importances_
weights = pd.Series(importances,
                    index=x.columns.values)
weights.sort_values()[-10:].plot(kind = 'barh')

```

```
<AxesSubplot:>
```




```
: Text(0.5, 1.0, 'Confusion Matrix for Decision Tree Classifier')
```



True Positive-46204,True Negative-45135,False Positive-4787,False Negative-5675

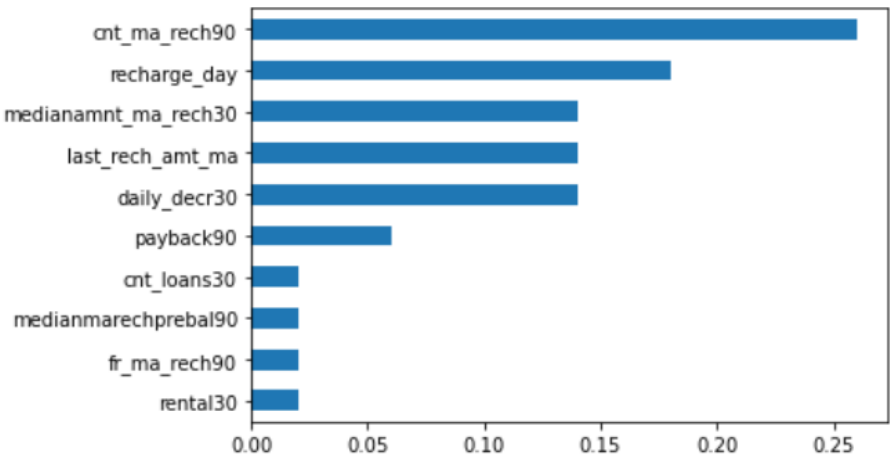
AdaBoostClassifier

```
adb = AdaBoostClassifier()
adb.fit(x_train,y_train)
predadb = adb.predict(x_test)

print(accuracy_score(y_test, predadb))
print(confusion_matrix(y_test, predadb))
print(classification_report(y_test,predadb))

```

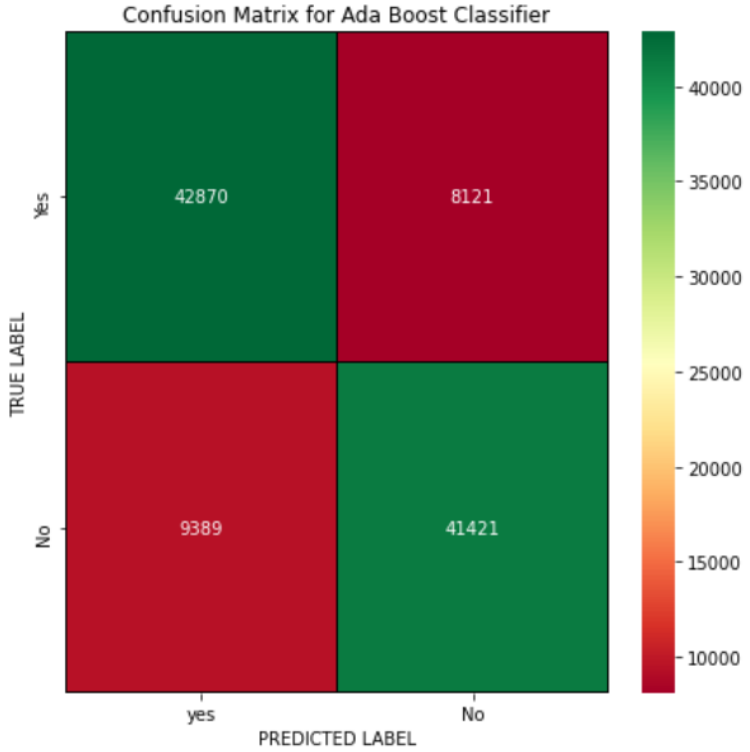
```
0.8279977603363424
[[42870  8121]
 [ 9389 41421]]
              precision    recall  f1-score   support

           0       0.82      0.84      0.83     50991
           1       0.84      0.82      0.83     50810

    accuracy                           0.83    101801
   macro avg       0.83      0.83      0.83    101801
weighted avg       0.83      0.83      0.83    101801
```

```
importances = adb.feature_importances_
weights = pd.Series(importances,
                    index=x.columns.values)
weights.sort_values()[-10:].plot(kind = 'barh')

```

```
<AxesSubplot:>
```



```
Text(0.5, 1.0, 'Confusion Matrix for Ada Boost Classifier')
```



True Positive-42870,True Negative-41421,False Positive-8121,False Negative-9389

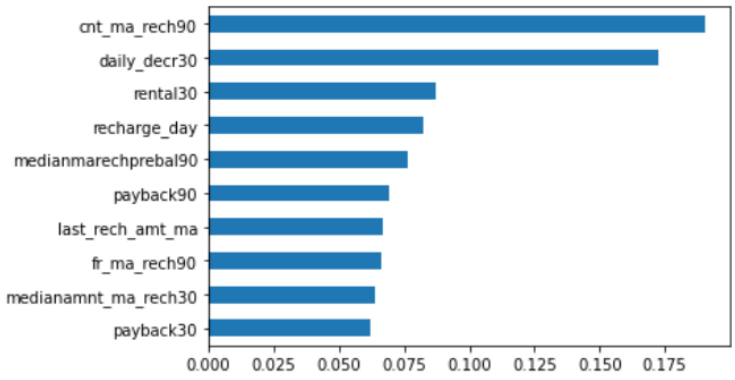
RandomForestClassifier

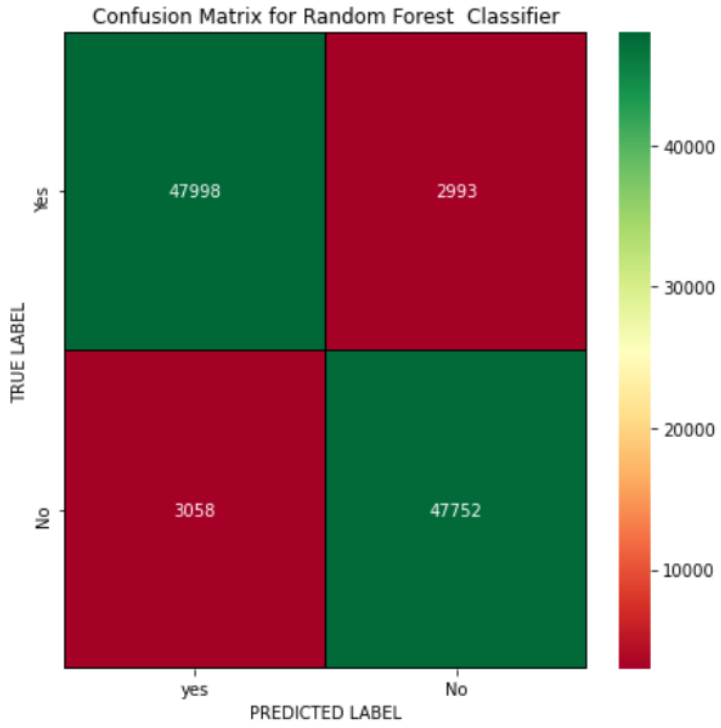
```
1  rfc = RandomForestClassifier()
2  rfc.fit(x_train,y_train)
3  predrfc = rfc.predict(x_test)
4
5  print(accuracy_score(y_test, predrfc))
6  print(confusion_matrix(y_test, predrfc))
7  print(classification_report(y_test,predrfc))
8
```

```
0.940560505299555
[[47998  2993]
 [ 3058 47752]]
              precision    recall  f1-score   support

           0       0.94      0.94      0.94     50991
           1       0.94      0.94      0.94     50810

    accuracy                           0.94    101801
   macro avg       0.94      0.94      0.94    101801
weighted avg       0.94      0.94      0.94    101801
```

```
importances = rfc.feature_importances_
weights = pd.Series(importances,
                    index=x.columns.values)
weights.sort_values()[-10:].plot(kind = 'barh')

```

```
<AxesSubplot:>
```





True Positive-47998,True Negative-47752,False Positive-2993,False Negative-3058

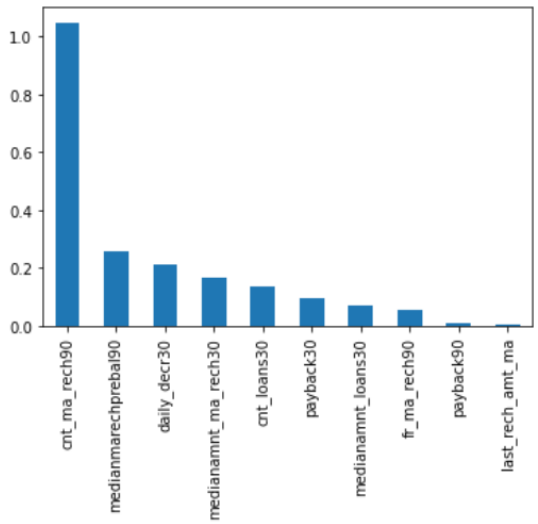
LogisticRegression

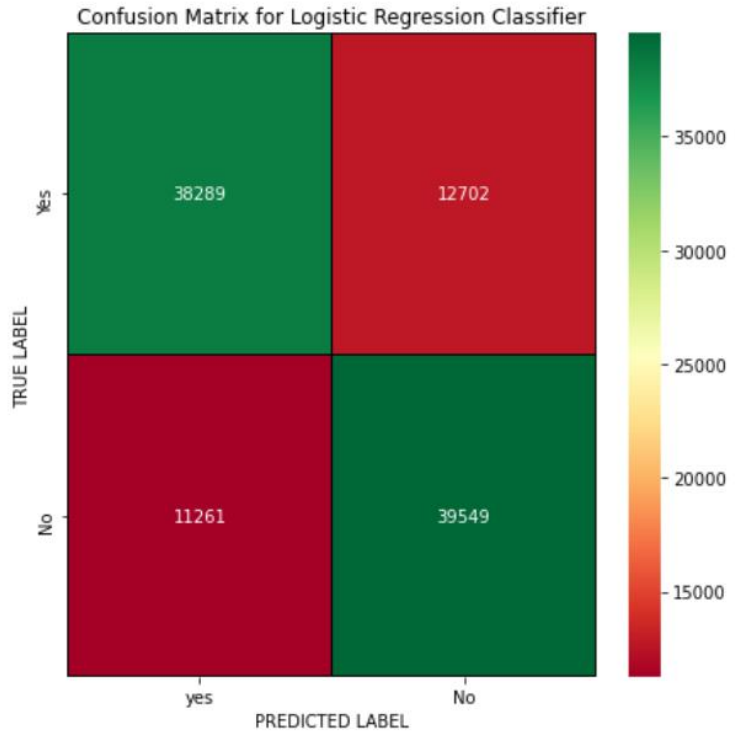
```
lgr = LogisticRegression()
lgr.fit(x_train,y_train)
predlgr = lgr.predict(x_test)

print(accuracy_score(y_test, predlgr))
print(confusion_matrix(y_test, predlgr))
print(classification_report(y_test,predlgr))

```

```
0.7646093849765719
[[38289 12702]
 [11261 39549]]
              precision    recall  f1-score   support

           0       0.77      0.75      0.76     50991
           1       0.76      0.78      0.77     50810

    accuracy                           0.76    101801
   macro avg       0.76      0.76      0.76    101801
weighted avg       0.76      0.76      0.76    101801
```

```
weights = pd.Series(lgr.coef_[0],
                 index=x.columns.values)
print (weights.sort_values(ascending = False)[:10].plot(kind='bar'))

```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```





True Positive-38289,True Negative-39549,False Positive-12702,False Negative-11261

Cross Validation of Models:

Cross Validation

```
from sklearn.model_selection import ShuffleSplit
n_samples = x.shape[0]
cv = ShuffleSplit(n_splits=5, test_size=0.30, random_state=151)
cvs = cross_val_score(lgr, x, y, cv=cv)
print("Cross Validation of LogisticRegression model ",cvs.mean())

```

```
Cross Validation of LogisticRegression model  0.7632007544130215
```

```
cv = ShuffleSplit(n_splits=5, test_size=0.30, random_state=151)
cvs = cross_val_score(knc, x, y, cv=cv)
print("Cross Validation of KNeighbors Classifier model ",cvs.mean())

```

```
Cross Validation of KNeighbors Classifier model  0.8858930658834392
```

```
cv = ShuffleSplit(n_splits=5, test_size=0.30, random_state=151)
cvs = cross_val_score(dtc, x, y, cv=cv)
print("Cross Validation of DecisionTree Classifier model ",cvs.mean())

```

```
Cross Validation of DecisionTree Classifier model  0.8974803783852809
```

```
cv = ShuffleSplit(n_splits=5, test_size=0.30, random_state=101)
cvs = cross_val_score(rfc, x, y, cv=cv)
print("Cross Validation of Random Forest Classifier model ",cvs.mean())

```

```
Cross Validation of Random Forest Classifier model  0.939405310360409
```

```
cv = ShuffleSplit(n_splits=5, test_size=0.30, random_state=101)
cvs = cross_val_score(adb, x, y, cv=cv)
print("Cross Validation of Ada Boost Classifier model ",cvs.mean())

```

```
Cross Validation of Ada Boost Classifier model  0.8289270242924921
```

So we can see that our model is performing excellent for unseen data also.Random forest has given highest score,so will do hyper parameter tuning for that.

Accuracy score is highest for Random Forest Model, Cross Validation Score is also highest for Random Forest. So, I have done hyperparameter tuning for Random Forest model.

```
from sklearn.model_selection import GridSearchCV

x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 151,test_size = 0.30)
rf = RandomForestClassifier()
parameters={'n_estimators' : [100,200],
            'criterion':['gini','entropy'],
           # 'max_depth':range(2,10),
            'max_features':["auto","sqrt","log2"],
            }



gridsearch=GridSearchCV(rf,parameters)
gridsearch.fit(x_train,y_train)
gridsearch.best_params_

```

```
{'criterion': 'entropy', 'max_features': 'sqrt', 'n_estimators': 200}
```

```
## Providing Best parameter to model:-
rf = RandomForestClassifier(criterion='entropy',n_estimators=200,max_features='sqrt')
rf.fit(x_train,y_train)
predrf = rf.predict(x_test)
score = accuracy_score(y_test,predrf)
print("Accuracy Score :",score*100)
print(confusion_matrix(y_test, predrf))
print(classification_report(y_test,predrf))

```

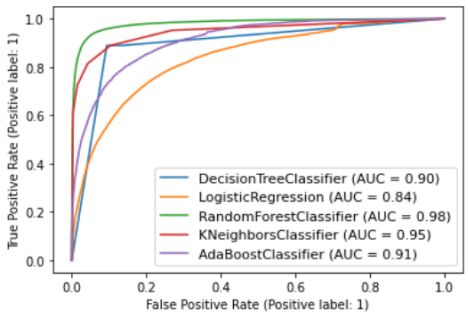
```
Accuracy Score : 94.04819206098172
[[47983  3008]
 [ 3051 47759]]
```

I didn’t get increased accuracy after parameters which I have given for hyperparameter tuning. We can increase score by trying different parameters, but since this process is very much time consuming, So,I have continued with my model with default parameters.

ROC AUC Curve:

Plotting roc_auc_curve

```
disp =plot_roc_curve(dtc,x_test,y_test)
plot_roc_curve(lgr,x_test,y_test,ax = disp.ax_)
plot_roc_curve(rfc,x_test,y_test,ax = disp.ax_)
plot_roc_curve(knc,x_test,y_test,ax = disp.ax_)
plot_roc_curve(adb,x_test,y_test,ax = disp.ax_)
plt.legend(prop={'size': 11},loc = 'lower right')
plt.show()

```



We can see Random Forest Model, Area Under Curve is 98 percent.Our model is performing extremely well.

Area under curve for Random Forest is 98 percent.

CONCLUSION

- Key Findings and Conclusions

  So here, I can conclude that customers who will pay back loan within 5 days, they fall under some of following observations:

  1.) Their daily amount spent from main account is continuous and more.
  2.) Their average main account balance (over last 30 days) is always maintained.
  3.) Their count and frequency of main account got recharged is continuous and more.
  4.) Their count of number of loans, amount of loans, maximum amount of loans, Median of amounts of loan is always more than a defaulter person.
  5.) They always payback their loan with in given limit.

  **Non-Defaulter users show their trust and regularly use services provided by Telecom service provider.

Conclusion

```
a = np.array(y_test)
predicted = np.array(rfc.predict(x_test))
new_df = pd.DataFrame({"Original":a,"Predicted":predicted},index= range(len(a)))
new_df

```

	Original	Predicted
0	1	1
1	0	0
2	1	0
3	1	0
4	1	1
...	...	...
101796	1	1
101797	1	1
101798	0	0
101799	1	1
101800	0	0

101801 rows × 2 columns

- # Learning Outcomes

- Applied quantitative modeling and data analysis techniques to the solution of real-world business problems, communicate findings, and effectively present results using data visualization techniques.
- Demonstrate knowledge of statistical data analysis techniques utilized in business decision making.
- Apply principles of Data Science to the analysis of business problems.
- Use data mining software to solve real-world problems.
- Apply algorithms to build machine intelligence.

  I have found Random Forest Model is giving highest accuracy score, among the models, which I have used for predicting the label.

- # Limitations

  Limitation of this solution is that my model is giving accuracy of 94%, We can improve it to 100% by trying different hyperparameter tuning parameters and trying different classification algorithms.