FLIP ROBO

Micro Credit Loan Use Case

Submitted by:

Ram kumar

ACKNOWLEDGMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend my sincere thanks to SME. Khushboo Garg.

We are highly indebted to Flip Robo technology for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

Thanks all.

Ram kumar

INTRODUCTION

- 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 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.

- 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.

- 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

Analytical Problem Framing

- Import library and load the dataset.

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

import warnings
warnings.filterwarnings('ignore')
```

```
import pandas as pd
df = pd.read_csv('Data file.csv')
df
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	mec
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
209588	209589	1	22758I85348	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	...	6.0	
209589	209590	1	95583I84455	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	...	6.0	
209590	209591	1	28556I85350	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	0.0	...	12.0	
209591	209592	1	59712I82733	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	38.0	...	12.0	
209592	209593	1	65061I85339	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	0.0	...	12.0	

209593 rows × 37 columns

```
#checking data dimension
df.shape
```

```
(209593, 37)
```

```
#display columns of dataframe
df.columns
```

```
Index(['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'],
      dtype='object')
```

```
#drop unnamed column
df.drop('Unnamed: 0',axis=1,inplace=True)
df
```

	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	...	maxamnt_loans
0	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	...	6
1	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	...	12
2	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	...	6
3	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	...	6
4	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	...	6
...	...	...	...	...	...	...	...	...	...	...	...	
209588	1	22758I85348	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	4048	...	6
209589	1	95583I84455	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	773	...	6

- Display all column name of dataset.

```
#display datatypes of columns
df.dtypes
```

```
label                     int64
msisdn                   object
aon                     float64
daily_decr30            float64
daily_decr90            float64
rental30                float64
rental90                float64
last_rech_date_ma       float64
last_rech_date_da       float64
last_rech_amt_ma          int64
cnt_ma_rech30             int64
fr_ma_rech30            float64
sumamnt_ma_rech30       float64
medianamnt_ma_rech30    float64
medianmarechprebal30    float64
cnt_ma_rech90             int64
fr_ma_rech90              int64
sumamnt_ma_rech90         int64
medianamnt_ma_rech90    float64
medianmarechprebal90    float64
cnt_da_rech30           float64
fr_da_rech30            float64
cnt_da_rech90             int64
fr_da_rech90              int64
cnt_loans30               int64
amnt_loans30              int64
maxamnt_loans30         float64
medianamnt_loans30      float64
cnt_loans90             float64
amnt_loans90              int64
maxamnt_loans90           int64
medianamnt_loans90      float64
payback30               float64
payback90               float64
pcircle                  object
pdate                    object
dtype: object
```

```
#display information of columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 36 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   label                 209593 non-null  int64  
 1   msisdn                209593 non-null  object 
 2   aon                   209593 non-null  float64
 3   daily_decr30          209593 non-null  float64
 4   daily_decr90          209593 non-null  float64
 5   rental30              209593 non-null  float64
 6   rental90              209593 non-null  float64
 7   last_rech_date_ma     209593 non-null  float64
 8   last_rech_date_da     209593 non-null  float64
 9   last_rech_amt_ma      209593 non-null  int64  
 10  cnt_ma_rech30         209593 non-null  int64  
 11  fr_ma_rech30          209593 non-null  float64
 12  sumamnt_ma_rech30     209593 non-null  float64
 13  medianamnt_ma_rech30  209593 non-null  float64
 14  medianmarechprebal30  209593 non-null  float64
 15  cnt_ma_rech90         209593 non-null  int64  
 16  fr_ma_rech90          209593 non-null  int64  
 17  sumamnt_ma_rech90     209593 non-null  int64  
 18  medianamnt_ma_rech90  209593 non-null  float64
 19  medianmarechprebal90  209593 non-null  float64
 20  cnt_da_rech30         209593 non-null  float64
 21  fr_da_rech30          209593 non-null  float64
 22  cnt_da_rech90         209593 non-null  int64  
 23  fr_da_rech90          209593 non-null  int64  
 24  cnt_loans30           209593 non-null  int64  
 25  amnt_loans30          209593 non-null  int64  
 26  maxamnt_loans30       209593 non-null  float64
 27  medianamnt_loans30    209593 non-null  float64
 28  cnt_loans90           209593 non-null  float64
 29  amnt_loans90          209593 non-null  int64  
 30  maxamnt_loans90       209593 non-null  int64  
 31  medianamnt_loans90    209593 non-null  float64
 32  payback30             209593 non-null  float64
 33  payback90             209593 non-null  float64
 34  pcircle               209593 non-null  object 
 35  pdate                 209593 non-null  object 
dtypes: float64(21), int64(12), object(3)
memory usage: 57.6+ MB
```

- Display datatypes and sum of null values.

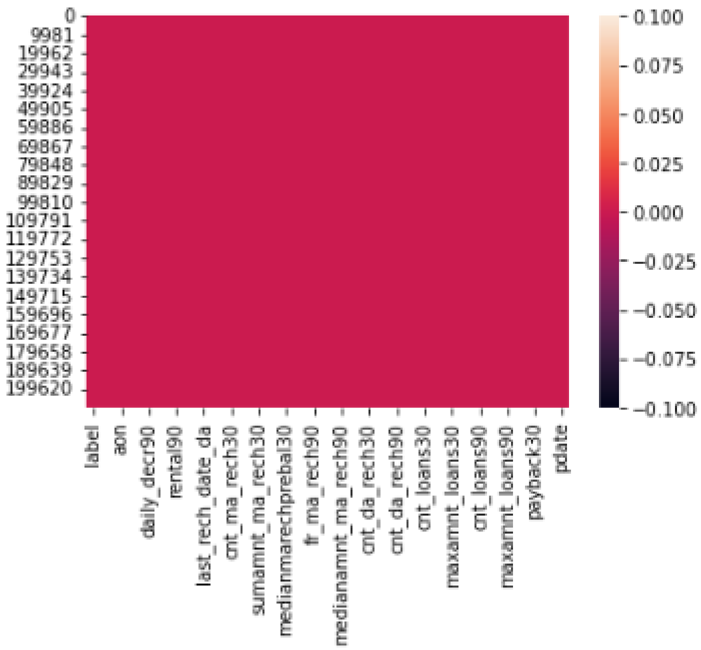
```
#display sum of null values in columns
df.isnull().sum()
```

```
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
```

- Display null values of columns using heatmap.

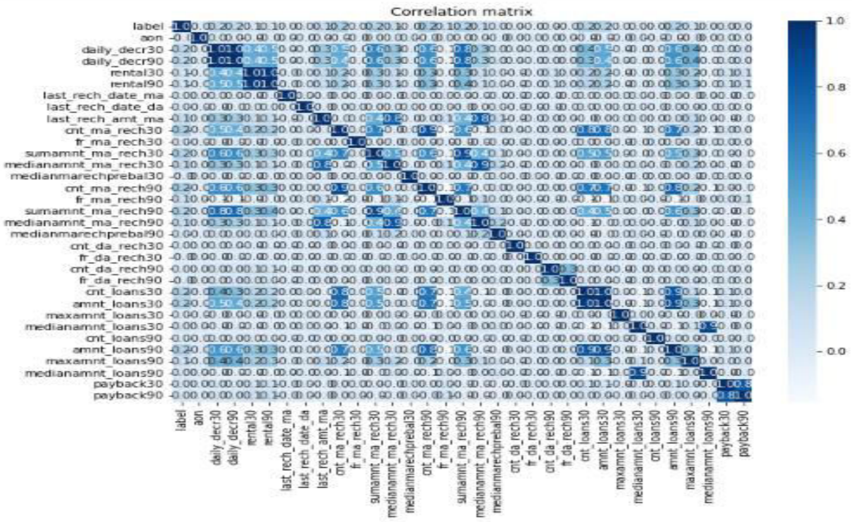
```
#display heatmap of null values in columns
sns.heatmap(df.isnull())
```

<AxesSubplot:>



- Display correlation of columns using heatmap.

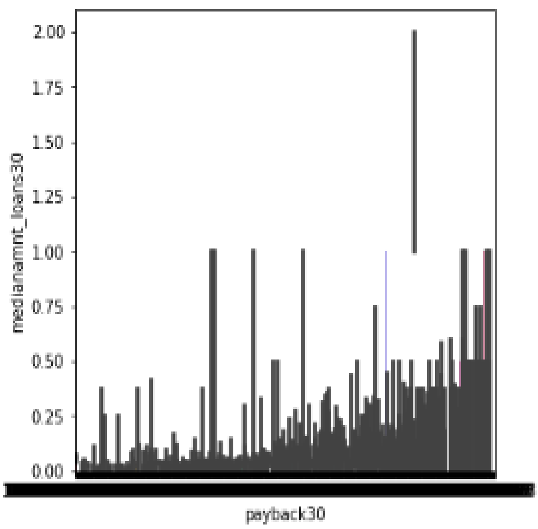
```
corr_mat = df.corr()
plt.figure(figsize=(10,8))
sns.heatmap(corr_mat, annot=True, fmt = '.1f',cmap = 'Blues')
plt.title('Correlation matrix')
plt.show()
```



- Display barplot of all columns.

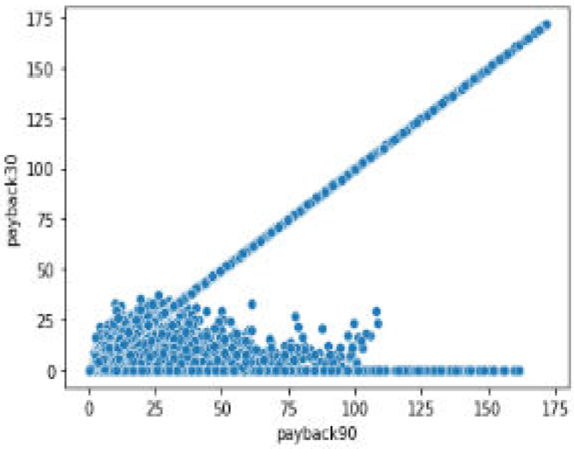
```
#citric acid vs Quality
plot = plt.figure(figsize=(5,5))
sns.barplot(x='payback30',y='medianamnt_loans30', data=df)
```

```
<AxesSubplot:xlabel='payback30', ylabel='medianamnt_loans30'>
```



- Display Scatterplot of payback30column.

```
sns.scatterplot(x='payback90',y='payback30',data=df)
plt.show()
```



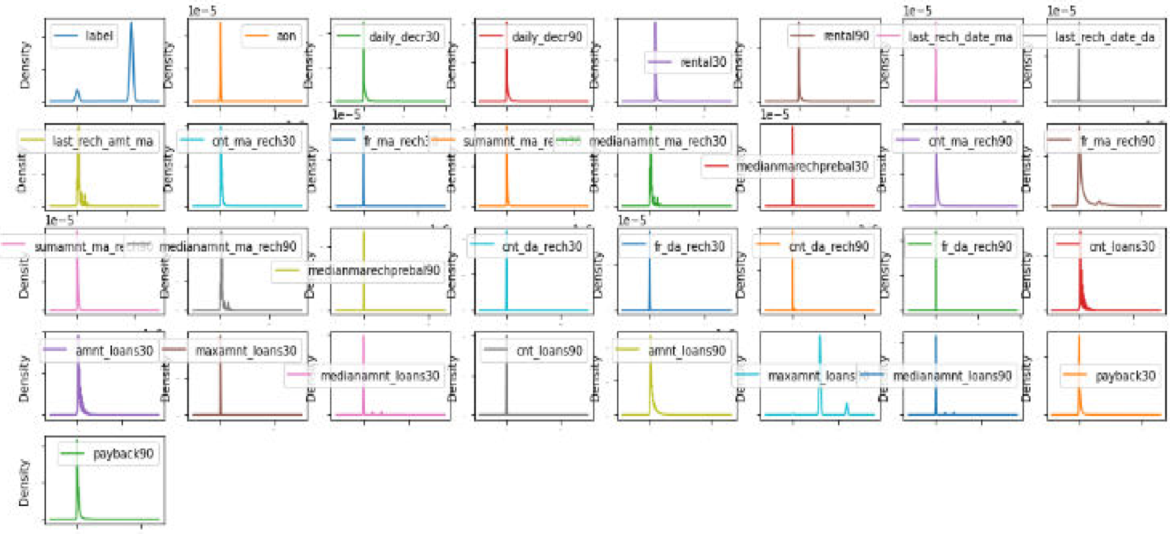
- Display statistical summary.

```
#statistical summary.
df.describe().style.background_gradient()
```

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cr
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	2
mean	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921	2064.452797	
std	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430	2370.786034	
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	0.000000	
25%	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000	770.000000	
50%	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000	1539.000000	
75%	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000	2309.000000	
max	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	55000.000000	

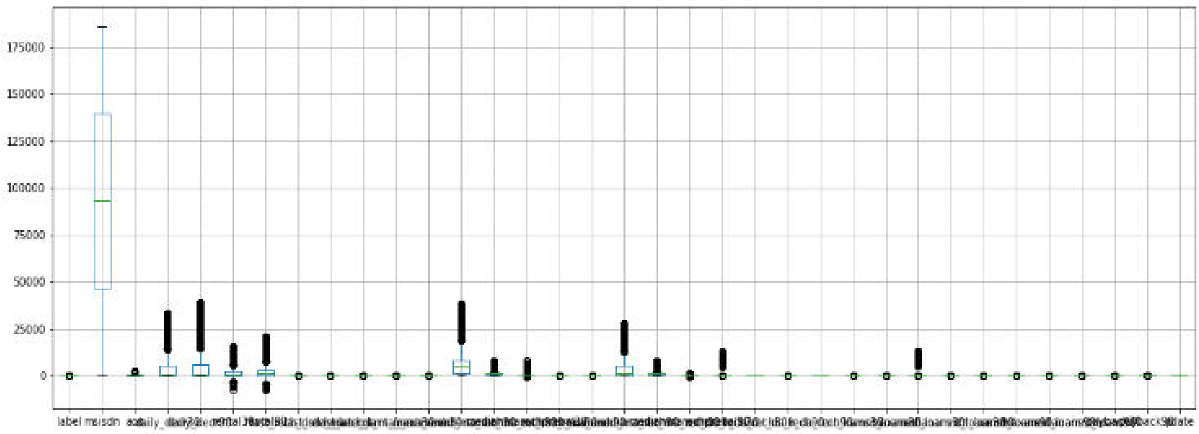
- Check the data distribution among all the columns

```
# Let's check the data distribution among all the columns
df.plot(kind='density', subplots=True, layout=(8,8), sharex=False, legend=True, fontsize=1, figsize=(18,12))
plt.show()
```



- **Checking outliers with boxplot.**

```
#checking outliers with boxplot
df.iloc[:,:].boxplot(figsize=[20,8])
plt.subplots_adjust(bottom=0.25)
plt.show()
```



- Train test split Here:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=.20,random_state = 58)
```

```
x_train
```

	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	...	cnt_loans30	amnt_lo
198600	1	15031	433.0	55.284000	55.284000	1301.76	1301.76	3.0	0.0	6	...	3	
127598	1	41858	1276.0	9320.604000	9357.720000	1808.20	2237.39	4.0	0.0	23	...	5	
162895	1	36907	211.0	40.880000	40.880000	670.56	670.56	1.0	0.0	21	...	2	
157586	0	117791	737.0	2883.898000	2888.040000	2039.55	2708.43	6.0	0.0	14	...	2	
192273	1	170829	1092.0	8905.930000	8936.450000	741.98	829.98	3.0	0.0	14	...	8	
...	...	...	...	...	...	...	...	...	...	...	...	...	
33881	1	102200	1760.0	3880.000000	4672.850000	3852.30	4444.19	21.0	0.0	14	...	1	
162607	1	144273	116.0	4148.554000	4215.180000	1387.91	1978.84	2.0	0.0	14	...	2	
54652	0	164931	1725.0	-5.000000	-5.000000	2478.30	2478.30	0.0	0.0	0	...	1	
45124	0	138819	180.0	-0.058887	-0.058887	2142.80	2142.80	0.0	0.0	0	...	1	
83798	1	7371	129.0	6771.610000	6825.220000	2709.92	3154.87	4.0	0.0	14	...	3	

129620 rows × 34 columns

```
y_train
```

```
198600    1
127598    1
162895    1
157586    1
192273    1
         ..
33881     0
162607    0
54652     0
45124     0
83798     1
Name: payback30, Length: 129620, dtype: int64
```

```
y_test
```

```
114683    0
31381     1
204345    1
158409    0
52324     1
         ..
76508     1
158481    0
57914     0
111071    1
119424    0
Name: payback30, Length: 32405, dtype: int64
```

```
print(x.shape, x_train.shape, x_test.shape)
```

```
(162025, 34) (129620, 34) (32405, 34)
```

- Display creating model:

```
# croete model:
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
```

```
maxAccu = 0
maxRS = 0

for i in range(1,100):
    x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = .20, random_state = i)
    rfc = RandomForestClassifier()
    rfc.fit(x_train,y_train)
    pred = rfc.predict(x_test)
    acc = accuracy_score(y_test, pred)
    print('accuracy',acc,'random_state',i)

    if acc > maxAccu:
        maxAccu = acc
        maxRS = i
```

```
#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
pred = rfc.predict(x_test)
print('accuracy',accuracy_score(y_test, pred)*100)
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
accuracy 98.17620737540503
[[17013   404]
 [  187 14801]]
              precision    recall  f1-score   support

           0       0.99      0.98      0.98     17417
           1       0.97      0.99      0.98     14988

    accuracy                           0.98     32405
   macro avg       0.98      0.98      0.98     32405
weighted avg       0.98      0.98      0.98     32405
```

- Cross Validation Here:

```
#Cross Validation
from sklearn.model_selection import cross_val_score

scr = cross_val_score(rfc, x, y, cv=5)
print('cross validation score of random forest classifier model :',scr.mean())
```

```
cross validation score of random forest classifier model : 0.9823977781206604
```

```
scr2 = cross_val_score(lr, x, y, cv=5)
print('cross validation score of logistic regression model :',scr2.mean())
```

```
cross validation score of logistic regression model : 0.7694553309674432
```

```
scr3 = cross_val_score(dtc, x, y, cv=5)
print('cross validation score of decision tree classifier model :',scr3.mean())
```

```
cross validation score of decision tree classifier model : 0.9756951087795093
```

- Hyper parameter tuning here:

```
#Hyper parameter tuning
from sklearn.model_selection import GridSearchCV

parameters = {'max_features': ['auto','sqrt','log2'],
              'max_depth': [4,5,6,7,8],
              'criterion':['gini','entropy']}
```

```
GCV = GridSearchCV(RandomForestClassifier(),parameters,cv=5,scoring='accuracy')
GCV.fit(x_train,y_train)
GCV.best_params_
```

```
{'criterion': 'entropy', 'max_depth': 8, 'max_features': 'auto'}
```

```
type(GCV)
```

```
sklearn.model_selection._search.GridSearchCV
```

```
GCV.best_estimator_
```

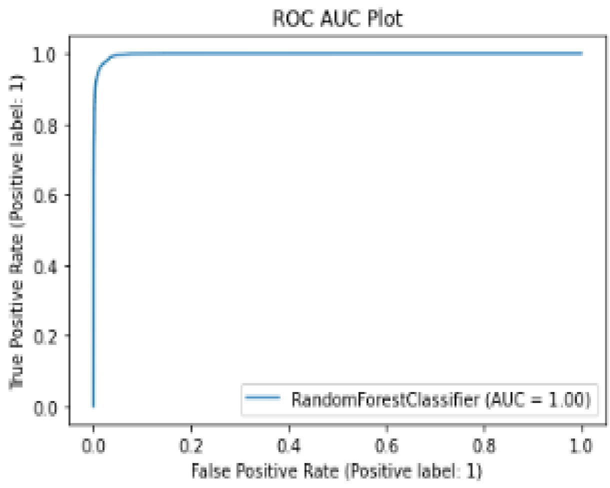
```
RandomForestClassifier(criterion='entropy', max_depth=8)
```

```
GCV_pred = GCV.best_estimator_.predict(x_test)
accuracy_score(y_test,GCV_pred)
```

```
0.9757753433112174
```

- Plot roc curve here:

```
: # Here plot_roc_curve:
from sklearn.metrics import plot_roc_curve
plot_roc_curve(GCV.best_estimator_,x_test,y_test)
plt.title("ROC AUC Plot")
plt.show()
```



- Save the mode here:

```
import pickle
filename = 'sales.pkl'
pickle.dump( rfc,open(filename,'wb'))
```

- Hardware and Software Requirements and Tools Used

➢ **Language :-** Python

➢ **Tool:-** Jupyter

➢ **OS:-** Windows 10

➢ **RAM:-** 8gb

CONCLUSION

- The Random Forest approach is appropriate for classification and regression tasks on datasets with many entries and features that are likely to have missing values when we need a highly accurate result while avoiding overfitting.

- the random forest provides relative feature significance, enabling you to select the most important features. It is more interpretable than neural network models but less interpretable than decision trees.

- Predicting Loan Default is highly dependent on the demographics of the people, people with lower income are more likely to default on loans.