MICRO CREDIT DEFAULTER PROJECT

Submitted by:

ATUL DAHIMA

INTRODUCTION

- ***Business Problem Framing***

- 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). 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.

- ***Exercise:***

- 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. 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.

- ***Points to Remember:***

- • There are no null values in the dataset.

- • There may be some customers with no loan history.

- • The dataset is imbalanced. Label '1' has approximately 87.5% records, while, label '0' has approximately 12.5% records.

- • For some features, there may be values which might not be realistic. You may have to observe them and treat them with a suitable explanation.

- • You might come across outliers in some features which you need to handle as per your understanding. Keep in mind that data is expensive and we cannot lose more than 7-8% of the data.
- Find Enclosed the Data Description File and The Sample Data for the Modeling Exercise.

Analytical Problem Framing

- ***Mathematical/ Analytical Modeling of the Problem***

❖ ***DATA SCRAPPING***

```
In [1]: #import libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns


        import warnings
        warnings.filterwarnings("ignore")

In [2]: data=pd.read_csv(r'C:\Users\ankus\OneDrive\Desktop\Data file.csv')

In [3]: data.head()
```

Out[3]:

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	medianamı
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	

5 rows × 37 columns

First we will import some required libraries and then import the dataset. We have to make prediction on the test dataset using train dataset

Few libraries are used initially for different purpose like:-

- *Numpy*:- for mathematical operation
- *Pandas*:- for data preprocessing
- *Matplotlib and Seaborn*:- for data visualization
- *Warnings*:- for some warnings

- Drop the unwanted column “Unnamed:0”

❖ ***DATA INFO AND DESCRIBE:-***

```
In [5]: data.isnull().sum().sum()
Out[5]: 0
```

it is clear that there is no null values

```
In [6]: data.describe()
```

Out[6]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cn
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	

8 rows × 33 columns

data.isnull().sum() shows that there is no null values in the data .

While data description shows some calculation about the data like:-mean,median,mode,std and some other calculation

On the top of the data description we can see that all the columns have same value which also defines that there is no null value

❖ ***LABEL ENCODING:-***

label encoding

```
In [8]: #last two columns need encoding as they having text
        from sklearn.preprocessing import LabelEncoder
        le=LabelEncoder()

In [9]: data = data.copy()
        for column in data.columns:
            data[column] = le.fit_transform(data[column])
        data.head()
```

Out[9]:

	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	...	maxamnt_loans30	medianam
0	0	40191	279	80775	77458	13831	14958	31	11	14	...	1	
1	1	142291	719	121865	124531	87782	84434	49	11	38	...	2	
2	1	33594	542	65526	63370	39144	35464	32	11	14	...	1	
3	1	104157	248	14833	14764	11309	11227	70	11	10	...	1	
4	1	6910	954	47174	47056	45072	40939	33	11	23	...	1	

5 rows × 36 columns

some of the columns were in text form and some were having high values so I encoded all the columns

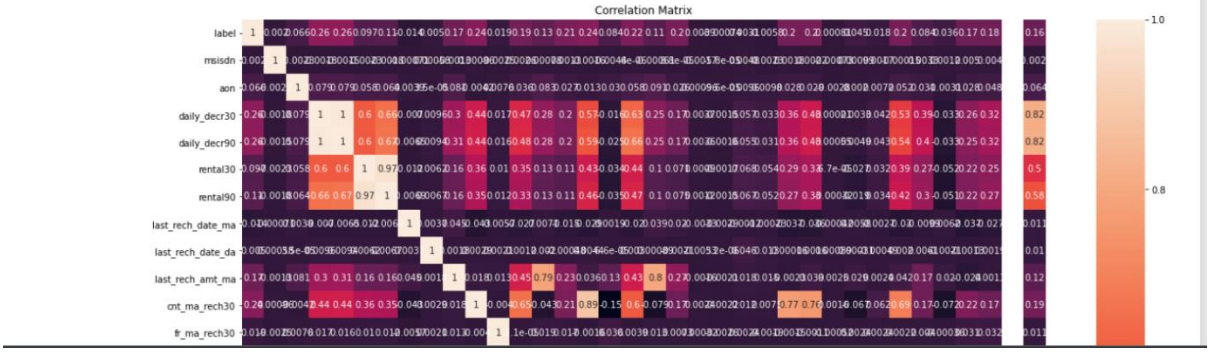
❖ ***Correlation :-***

check for correlation

In [10]:
```
data.corr()
corr_mat=data.corr()

#size of the canvas
plt.figure(figsize=([20,20]))

#plot the correlation matrix
sns.heatmap(corr_mat,annot=True)
plt.title("Correlation Matrix")
plt.show()
```



Above diagram shows the correlation among all the columns.

As here are so much column so we cannot easily identify the correlation among all the columns so lets print the correlation table.

In [33]:
```
corr_mat=data.corr()
corr_mat
```

Out[33]:

	label	msisdn	aon	daily_decr30	rental30	last_rech_date_ma	last_rech_date_da	cnt_ma_rech30	fr_ma_rech30	sumamnt
label	1.000000	0.001976	0.066149	0.261715	0.096589	-0.014236	0.004965	0.239192	0.019279	
msisdn	0.001976	1.000000	-0.002341	-0.001788	-0.002252	-0.000709	0.000582	0.000960	-0.002464	
aon	0.066149	-0.002341	1.000000	0.078593	0.057727	0.003947	0.000035	-0.004206	0.007570	
daily_decr30	0.261715	-0.001788	0.078593	1.000000	0.598928	-0.006978	0.009567	0.440957	0.016722	
rental30	0.096589	-0.002252	0.057727	0.598928	1.000000	-0.011588	0.006177	0.357525	0.010212	
last_rech_date_ma	-0.014236	-0.000709	0.003947	-0.006978	-0.011588	1.000000	0.003732	-0.042622	-0.005670	
last_rech_date_da	0.004965	0.000582	0.000035	0.009567	0.006177	0.003732	1.000000	0.002936	0.002145	
cnt_ma_rech30	0.239192	0.000960	-0.004206	0.440957	0.357525	-0.042622	0.002936	1.000000	-0.003963	
fr_ma_rech30	0.019279	-0.002464	0.007570	0.016722	0.010212	-0.005670	0.002145	-0.003963	1.000000	
sumamnt_ma_rech30	0.186063	0.002633	0.035528	0.471294	0.350136	-0.026567	0.001190	0.654281	-0.000021	
medianamnt_ma_rech30	0.132773	0.000782	0.083262	0.277373	0.134974	0.007068	0.001988	-0.043296	0.018787	
medianmarechprebal30	0.206274	0.001274	0.027131	0.203869	0.105207	-0.014818	-0.000484	0.211210	0.017050	
fr_ma_rech90	0.084385	-0.004413	0.029914	-0.016440	-0.034426	0.001881	-0.000066	-0.153823	0.035886	
medianmarechprebal90	0.200516	0.000088	0.025674	0.167870	0.070804	0.019579	-0.002063	0.174011	0.007271	
cnt_da_rech90	0.003127	0.004751	0.009636	0.057183	0.068385	-0.001165	0.046453	0.012305	0.002386	
fr_da_rech90	-0.005784	-0.002343	0.009753	0.032576	0.053964	-0.000226	0.012912	0.007501	-0.001867	
cnt_loans30	0.196413	0.001768	0.027602	0.363555	0.285234	-0.037466	0.000163	0.770173	-0.001466	
maxamnt_loans30	0.000810	-0.000731	-0.002758	0.000213	-0.000067	-0.000424	0.000894	0.001573	-0.000522	
cnt_loans90	0.018315	0.001659	0.007184	0.042034	0.032026	-0.002690	-0.000491	0.061844	0.002438	

Using abobe table we can easily identify the correlation among all the columns.

Some columns are highly correlated to each other so lets drop some columns

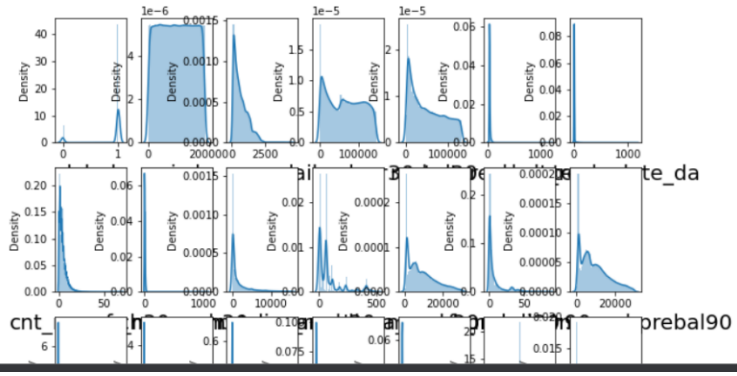
In [12]:
```
#some columns are highle correlated to each other so lets remove one of those column
data=data.drop(["amnt_loans90","payback90","medianamnt_loans90","amnt_loans30","medianamnt_ma_rech90","cnt_da_rech30","fr_da_rech
```

❖ ***DATA CORRELATION:-***

```
In [35]: #Let's check the data distribution among all the columns
plt.figure(figsize=(10,10))
plotnumber=1

for column in data:
    if plotnumber<=28:
        ax=plt.subplot(4,7,plotnumber)
        sns.distplot(data[column])
        plt.xlabel(column,fontsize=20)

    plotnumber+=1
plt.show()
```



Above code shows the data distribution among all the columns , the data distribution is looking not so good , let go ahead with checking skewness in the data distribution.

- ***Data Preprocessing Done***

Separate target and variables

seperate features and target variable

```
In [16]: x=data.drop(["label"],axis=1)
y=data["label"]
```

Check skewness:-

checking skewness

```
In [19]: x.skew().sort_values()
```

```
Out[19]: msisdn                  0.000719
         daily_decr30            0.083579
         pdate                   0.116409
         rental30                0.441806
         medianmarechprebal90    0.777023
         medianmarechprebal30    0.853740
         payback30               1.188146
         maxamnt_loans90         1.678304
         aon                     1.806204
         medianamnt_ma_rech30    2.035013
         sumamnt_ma_rech30       2.269945
         fr_ma_rech90            2.285423
         cnt_ma_rech30           2.684106
         cnt_loans30             2.685234
         last_rech_date_ma      16.524637
         last_rech_date_da      16.904463
         cnt_loans90            17.607075
         fr_ma_rech30           17.638960
         maxamnt_loans30        18.285158
         cnt_da_rech90          23.642903
         fr_da_rech90           26.899282
         dtype: float64
```

As we can see that lots of column are highly skeweed so we will try some method for handle the skewness of the data

```
In [36]: #remove skewness

         from sklearn.preprocessing import power_transform
         x_new=power_transform(x)
         x=pd.DataFrame(x_new,columns=x.columns)
```

```
In [21]: #validating that skewness has been removed or not

         x.skew().sort_values(ascending=False)
```

```
Out[21]: fr_da_rech90           15.469700
         cnt_da_rech90            6.006577
         fr_ma_rech30             0.147462
         fr_ma_rech90             0.142952
         payback30                0.120125
         cnt_loans90              0.097372
         cnt_loans30              0.036257
         cnt_ma_rech30           -0.000221
         aon                     -0.014895
         medianamnt_ma_rech30    -0.069444
         medianmarechprebal30    -0.110851
         medianmarechprebal90    -0.112579
         sumamnt_ma_rech30       -0.130283
         maxamnt_loans90         -0.133914
         rental30                -0.178777
         pdate                   -0.217699
         msisdn                  -0.291511
         daily_decr30            -0.352855
         last_rech_date_ma       -0.405624
         maxamnt_loans30         -1.110382
         last_rech_date_da      -25.420242
         dtype: float64
```

Data is much better then before but some columns are still skeweed so we can try log transform method to remove the skewness

DATA SCALING:-

data scaling

In [22]:
```
#data scaling using standard scaler
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [23]:
```
x_scaled=scaler.fit_transform(x)
x_scaled
```

Out[23]:
```
array([[-0.92855164, -0.68070707,  0.43051207, ..., -0.27047705,
         1.28594442,  0.57775419],
       [ 0.90402732,  0.36130212,  1.13026074, ...,  2.25810926,
        -0.95865368,  1.36067787],
       [-1.07962307,  0.02609381,  0.13524444, ..., -0.27047705,
        -0.95865368,  1.67272967],
       ...,
       [-0.63401932,  0.80743456,  1.11955707, ...,  2.25810926,
         1.06572313,  0.92409065],
       [ 0.41109115,  1.56099594,  1.14383033, ...,  2.25810926,
        -0.95865368,  0.77241039],
       [ 0.57252135,  1.42609049,  0.61573518, ...,  2.25810926,
        -0.95865368,  0.03958262]])
```

Training Process:-

Import the required libraries for the training process and split the data into training and testing process

Training Process

In [25]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
```

In [26]:
```
#spliting data into training and testing data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state = 180)
```

In [27]:
```
x_train
```

Out[27]:

	msisdn	aon	daily_decr30	rental30	last_rech_date_ma	last_rech_date_da	cnt_ma_rech30	fr_ma_rech30	sumamnt_ma_rech30	medianamnt_m
209034	0.925567	-0.694904	0.153976	1.087607	-0.293134	-0.153574	0.620196	0.179831	0.027705	
179233	0.542354	0.155888	-0.843001	-1.751368	-0.020372	-0.153574	0.095937	0.646702	-0.329492	
195096	0.748389	0.674862	0.783996	0.077866	-0.108413	4.702022	1.144806	0.179831	0.539474	
60718	1.270755	-1.023850	-0.190617	1.143199	-0.199274	-0.153574	1.716867	-1.128194	1.418872	
83499	0.476475	1.472720	-0.759473	-0.235137	-0.390191	-0.153574	0.095937	0.646702	0.025579	
...	...	...	...	...	...	...	...	...	...	
196485	0.435726	-1.395410	-1.146617	-0.853299	0.382842	-0.153574	-0.799483	-1.128194	-0.874114	
83997	0.092171	1.234347	1.186100	0.358346	0.065014	-0.153574	0.095937	1.496106	0.692582	
205835	-0.653023	-1.804580	-1.820962	-1.287883	-0.490663	-0.153574	-1.693882	-1.128194	-1.628181	
68374	-0.556270	1.620488	0.502597	-0.941567	0.800764	-0.153574	-0.799483	-1.128194	-0.874114	

Now perform with different model and use the best model for cross validation and hyper parameter tunning

Logistic regression

logistic regression

In [31]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
pred=lr.predict(x_test)
print("Accuracy",accuracy_score(y_test,pred)*100)
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
Accuracy 88.93108647111586
[[ 1425  5050]
 [  750 45174]]
              precision    recall  f1-score   support

           0       0.66      0.22      0.33      6475
           1       0.90      0.98      0.94     45924

    accuracy                           0.89     52399
   macro avg       0.78      0.60      0.63     52399
weighted avg       0.87      0.89      0.86     52399
```

Decision tree classifier

Decision tree classifier

In [32]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
predict=dt.predict(x_test)
print("Accuracy",accuracy_score(y_test,predict)*100)
print(confusion_matrix(y_test,predict))
print(classification_report(y_test,predict))
```

```
Accuracy 88.21160709173839
[[ 3574  2901]
 [ 3276 42648]]
              precision    recall  f1-score   support

           0       0.52      0.55      0.54      6475
           1       0.94      0.93      0.93     45924

    accuracy                           0.88     52399
   macro avg       0.73      0.74      0.73     52399
weighted avg       0.89      0.88      0.88     52399
```

Random forest classifier

Random forest classifier

In [33]:
```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier()
rf.fit(x_train,y_train)
predrf=rf.predict(x_test)
print("Accuracy",accuracy_score(y_test,predrf)*100)
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
Accuracy 92.23649306284472
[[ 3313  3162]
 [  906 45018]]
              precision    recall  f1-score   support

           0       0.79      0.51      0.62      6475
           1       0.93      0.98      0.96     45924

    accuracy                           0.92     52399
   macro avg       0.86      0.75      0.79     52399
weighted avg       0.92      0.92      0.92     52399
```

LogisticRegression,RandomForest and Decision tree classifier are predicting good accuracy. now we will check cross validation score as well for overfitting

---

All the above model are performing good so lets move ahead for cross validation

cross validation

Cross validation

In [34]:
```
from sklearn.model_selection import cross_val_score
scr=cross_val_score(lr,x,y,cv=5)
print("Cross Validation score of Logistic Regression model:",scr.mean())
```

```
Cross Validation score of Logistic Regression model: 0.8884647877288387
```

In [35]:
```
scr=cross_val_score(rf,x,y,cv=5)
print("Cross Validation score of Random FOrest:",scr.mean())
```

```
Cross Validation score of Random FOrest: 0.9208752205821373
```

In [37]:
```
dt=cross_val_score(dt,x,y,cv=5)
print("Cross Validation score of DecisionTreeClassifier:",dt.mean())
```

```
Cross Validation score of DecisionTreeClassifier: 0.8825962744275945
```

Random forest is performing best among all so we will continue with Random Forest

---

In above all the model random forest classifier is performing best so lets proceed further for hyper parameter tunning

Hyper parameter Tunning

In [38]:
```
from sklearn.model_selection import GridSearchCV

#creating parameter list to pass in GridSearchCv
parameters={'max_features':['auto','sqrt','log2'],
            'max_depth':[4,5,6,7,8],
            'criterion':['gini','entropy']}
```

In [39]:
```
GCV=GridSearchCV(RandomForestClassifier(),parameters,cv=5,scoring="accuracy")
GCV.fit(x_train,y_train)    #fitting the data in the model
GCV.best_params_          #print the best parameter found by GridSearchCV
```

Out[39]:
```
{'criterion': 'entropy', 'max_depth': 8, 'max_features': 'auto'}
```

In [40]:
```
 GCV_pred=GCV.best_estimator_.predict(x_test)     #prediccting with best parameters
accuracy_score(y_test,GCV_pred)
```

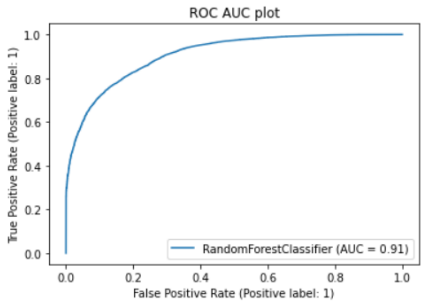
Out[40]:
```
0.9083570297143075
```

Model is giving good accuracy of 90% after the hyper parameter tunning

Now plot the roc_auc_curve

ROC AUC Plot

In [42]:
```
from sklearn.metrics import plot_roc_curve
plot_roc_curve(GCV.best_estimator_,x_test,y_test)
plt.title("ROC AUC plot")
plt.show()
```



AUC score is pretty good of 91%

CONCLUSION

We started with the data exploration where we got a feeling for the dataset, checked about missing data and learned which features are important. During this process we used seaborn and matplotlib to do the visualizations. During the data pre-processing part. Afterwards we started training model using Logistic Regression , Decision Tree Classifier , Random Forest Classifier and applied cross validation on it. Then we do the Hyper Parameter tunning of the model and get the final accuracy score for the model. Later we save the model for later use for prediction