MICRO CREDIT DEFAULTER PROJECT

Submitted by:

CHETHANA M

ACKNOWLEDGMENT

I express.my sincere gratitude to **Flip Robo Technologies** for giving me this opportunity to carry out the project work.

A special thanks to my mentor **Mohd Kashif** for guiding me in completing this project and being available to resolve my doubts whenever I raise any tickets.

I also would love to take this moment to thank **DataTrained** for giving me all the knowledge about how to build an effective machine learning model and providing this opportunity to work as intern in Flip Robo Technologies.

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 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.

Technical goals :

• Which variables are important to predict if the customer paid back the loan or not?

• How these variables are responsible describe the repayment of loan?

Our main aim today is to make 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 based on other variables. We are going to use Logistic Regression, Decision Tree Classifier, K Neighbors Classifier, Random forest classifier to build the different models for this dataset and see which model gives us a good accuracy.

The Motivation behind building this model is to understand how capable each applicant is of paying back the loan in the given period and tell the banker if the loan can be sanctioned to the applicant or not.

Analytical Problem Framing

In this project, prediction will be made on whether the customer will be paying back the loan or not in the given period of time using given explanatory variables that cover many aspects of the customers' financial history. The goal of this project is to create a model that is able to accurately predict if the customer will pay back the loan or not under the given features.

Here we can observe financial history of each customer in details and dataset includes 36 features such as,

Label	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}
Msisdn	mobile number of user
Aon	age on cellular network in days
daily_decr30	Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
daily_decr90	Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
rental30	Average main account balance over last 30 days
rental90	Average main account balance over last 90 days
last_rech_date_ma	Number of days till last recharge of main account
last_rech_date_da	Number of days till last recharge of data account
last_rech_amt_ma	Amount of last recharge of main account (in Indonesian Rupiah)
cnt_ma_rech30	Number of times main account got recharged in last 30 days
fr_ma_rech30	Frequency of main account recharged in last 30 days
sumamnt_ma_rech30	Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
medianamnt_ma_rech30	Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
medianmarechprebal30	Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
cnt_ma_rech90	Number of times main account got recharged in last 90 days
fr_ma_rech90	Frequency of main account recharged in last 90 days
sumamnt_ma_rech90	Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)
medianamnt_ma_rech90	Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)
medianmarechprebal90	Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)
cnt_da_rech30	Number of times data account got recharged in last 30 days
fr_da_rech30	Frequency of data account recharged in last 30 days
cnt_da_rech90	Number of times data account got recharged in last 90 days
fr_da_rech90	Frequency of data account recharged in last 90 days
cnt_loans30	Number of loans taken by user in last 30 days
amnt_loans30	Total amount of loans taken by user in last 30 days
maxamnt_loans30	maximum amount of loan taken by the user in last 30 days
medianamnt_loans30	Median of amounts of loan taken by the user in last 30 days
cnt_loans90	Number of loans taken by user in last 90 days

amnt_loans90	Total amount of loans taken by user in last 90 days
maxamnt_loans90	maximum amount of loan taken by the user in last 90 days
medianamnt_loans90	Median of amounts of loan taken by the user in last 90 days
payback30	Average payback time in days over last 30 days
payback90	Average payback time in days over last 90 days
Pcircle	telecom circle
Pdate	Date

Data contains 209593 instances each having 36 variables.
Data contains no Null values.
Extensive EDA has to be performed to gain relationships of important variable and label.
Data contains numerical as well as categorical variable which needs to handled accordingly.
Machine Learning models is needed to be built to predict if the customer pays back loan in the given period or not.
You need to find important features which affect the label positively or negatively.

```
In [93]: dataset=pd.read_csv('micro_credit_defaulter.csv')

In [94]: df=pd.DataFrame(dataset)

In [95]: df.head(5)
```

Out[95]:

	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_rech3(
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	2
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	(
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	7

Data Pre-processing Done :

There was no missing data, outliers were present and data was skewed, columns that doesn't create any impact on output variable. Treated the data for its skewness using 'Yeo-Johnson' method. The outliers were removed from continuous data using z-score method. Multicollinearity was checked and few columns were dropped which were creating high multicollinearity in the dataset.

Data Inputs- Logic- Output :

As discussed before there are 36 variables in the dataset, 1 output (Label) variable and 35 input variables.

INPUT VARIABLES : 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, r_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

OUTPUT VARIABLE : Label

About the Algorithms used :

The major aim in this project is to predict if the applicant pays back the loan or not in the given period of time based on the features using some of the regression and classifier algorithms.

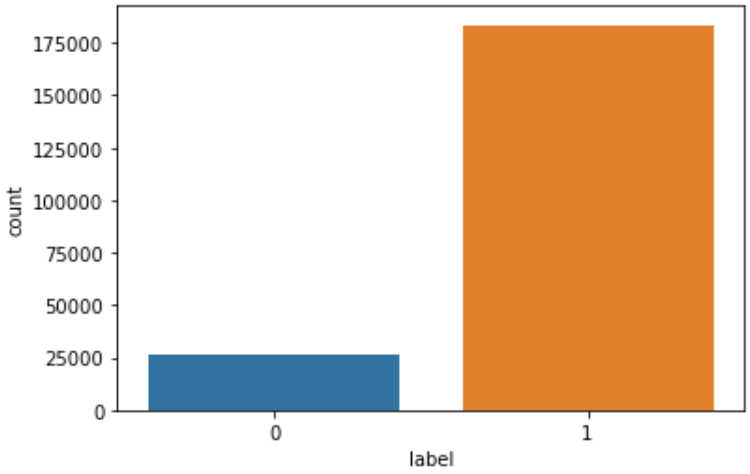
1. Logistic Regression
2. Decision Tree Classifier
3. KNeighbors Classifier
4. Random Forest Classifier

Machine Learning Packages are used for in this Project

The dataset was available in the csv file which had 209593 instances and 36 variables, this csv file was uploaded to Jupyter and read using pandas library. Once the dataset is read DataFrame is created and further EDA process is done on the dataset using different functions available in pandas.

The Seaborn and Matplotlib are used to plot the different graphs and understand the relationship between each variable and output variable.

EDA and Visualization



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

The data has 209593 column, it has a variable ‘msisdn’ which is mobile numbers of the customers, which we check for the unique values of this variable we can see that it has 186243 unique values, which means that duplicates are there in the dataset. Now we can remove the duplicate values from the dataset.

```
In [107]: df['msisdn'].nunique()
Out[107]: 186243
```

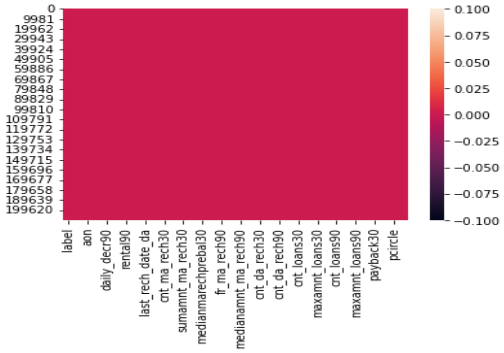
msidn is a mobile number of user and mobile number is unique for every customers. There are only 186243 unique number out of 209593 so rest of the data is duplicates entry so we have to remove those duplicate entries

```
In [108]: df = df.drop_duplicates(subset = 'msisdn',keep='first')
          df.shape
Out[108]: (186243, 36)
```

Hence after removing the duplicates we can see that we have only 186243 rows and 34 columns

Hence after removing duplicates we are have 186243 rows and 36 columns.

```
In [102]: sns.heatmap(df.isnull())
Out[102]: <AxesSubplot:>
```



From the above plot we can observe that there are no missing values in the dataset.

```
In [109]: df.describe(include=['object','datetime'])
```

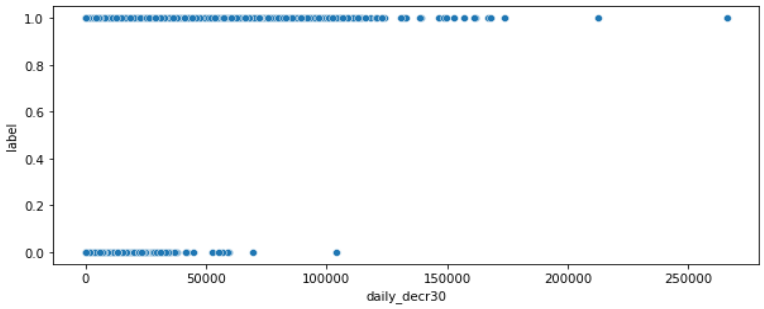
Out[109]:

	msisdn	pcircle	pdate
count	186243	186243	186243
unique	186243	1	82
top	21408I70789	UPW	2016-07-05
freq	1	186243	2825

We can see that in 'msisdn' variable there are 186243 unique values and is object type, this variable consists data of mobile number of user, Hence this variable has nothing to do with deciding target variable and we can drop off this variable.
The variable 'pdate' is about date when the loan was issued even this data has no impact on the target variable hence this column can also be dropped.

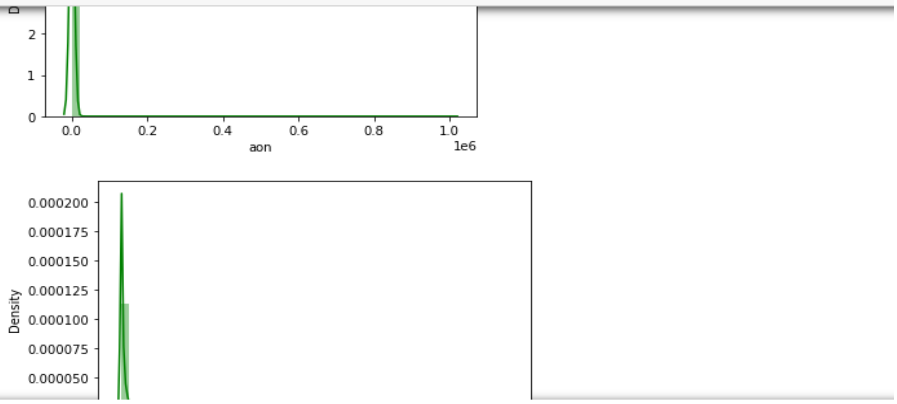
```
In [115]: plt.figure(figsize = (10,4))
          sns.scatterplot(x='daily_decr30', y='label', data=df)
```

Out[115]: <AxesSubplot:xlabel='daily_decr30', ylabel='label'>

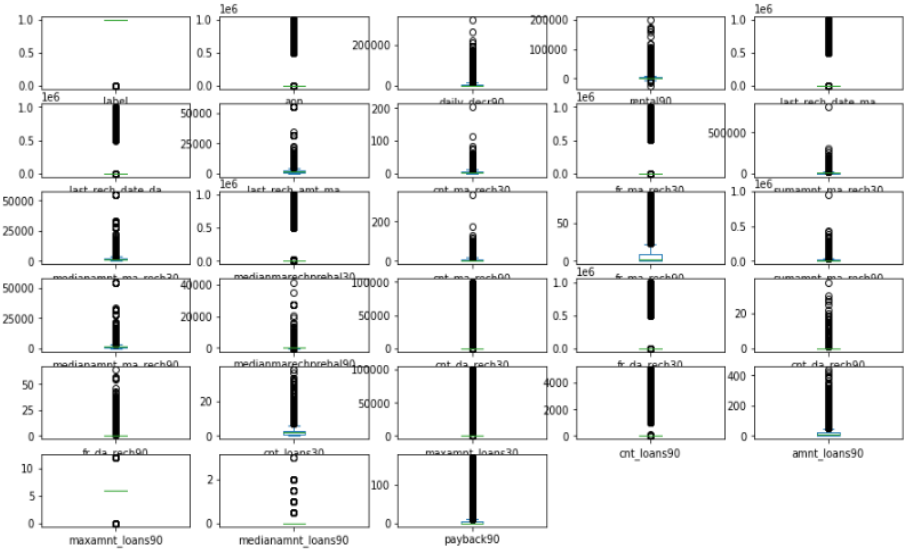


From the above plot we can observe that the label is success in most of the cases for those customers whose daily amount spent from main account averaged for 30 days is higher.

```
In [120]: for col in df.columns:
              sns.distplot(df[col],color='g')
              plt.show()
```



By plotting graphs as shown above we can observe that all the variables are skewed, the same can be removed in the further steps.



From the above plots we can observe that outliers are present in almost all the columns which needs to be treated in the up coming process.

```
In [116]: pd.set_option('display.max_columns', None)
          df.describe().transpose()
```

Out[116]:

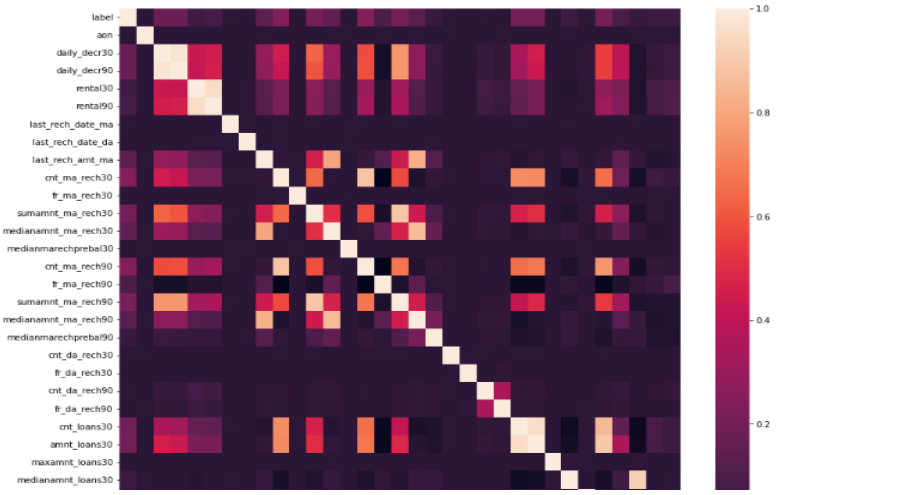
	count	mean	std	min	25%	50%	75%	max
label	186243.0	0.861149	0.345792	0.000000	1.000	1.000000	1.000000	1.000000
aon	186243.0	8145.172831	75865.800335	-48.000000	241.000	522.000000	979.000000	999860.755168
daily_decr30	186243.0	5049.834558	8758.493684	-93.012667	38.522	1332.110667	6710.066667	265926.000000
daily_decr90	186243.0	5696.623026	10340.696921	-93.012667	38.791	1360.000000	7229.220000	320630.000000
rental30	186243.0	2634.459831	4291.392727	-23737.140000	256.000	1036.700000	3247.565000	198926.110000
rental90	186243.0	3401.366654	5720.521590	-24720.580000	288.000	1281.860000	4070.115000	200148.110000
last_rech_date_ma	186243.0	3728.482692	53744.726961	-29.000000	1.000	3.000000	7.000000	998650.377733
last_rech_date_da	186243.0	3732.953680	53563.651356	-29.000000	0.000	0.000000	0.000000	999171.809410
last_rech_amt_ma	186243.0	2073.202891	2415.927330	0.000000	770.000	1539.000000	2309.000000	55000.000000
cnt_ma_rech30	186243.0	3.646295	3.915321	0.000000	1.000	3.000000	5.000000	203.000000
fr_ma_rech30	186243.0	3710.729377	53423.442186	0.000000	0.000	2.000000	6.000000	999606.368132
sumamnt_ma_rech30	186243.0	7196.403199	9675.574979	0.000000	1539.000	4617.000000	9335.000000	810096.000000
medianamnt_ma_rech30	186243.0	1819.384882	2116.884332	0.000000	770.000	1539.000000	1924.000000	55000.000000
medianmarechprebal30	186243.0	3909.633397	54427.326799	-200.000000	9.500	33.000000	82.910000	999479.419319
cnt_ma_rech90	186243.0	5.769575	6.560925	0.000000	1.000	4.000000	8.000000	336.000000
fr_ma_rech90	186243.0	7.864827	12.762714	0.000000	0.000	2.000000	9.000000	88.000000
sumamnt_ma_rech90	186243.0	11567.675902	15967.474119	0.000000	2309.000	6609.000000	14991.000000	953036.000000
medianamnt_ma_rech90	186243.0	1881.044818	2132.332449	0.000000	773.000	1539.000000	1924.000000	55000.000000
medianmarechprebal90	186243.0	93.788935	387.276953	-200.000000	13.600	35.380000	79.000000	41456.500000

Key Observations :

- mean > median (50th percentile) almost in all the columns, hence the data has right skewness in it.
- We can see that the difference between 75% percentile and max is high is case of all the variables, hence outliers are present in the data.

Checking for correlation of the variables :

```
In [118]: plt.figure(figsize=(14,14))
          sns.heatmap(df.corr())
Out[118]: <AxesSubplot:>
```



Key Observations :

- daily_decr30 and daily_decr90 are having good correlation to each others.
- In the similar way cnt_loans30 and amnt_loans30, payback30 and payback90 have good correlations.
- label has a better correlation with daily_decr30, amnt_loans90, cnt_loans30, cnt_ma_rech90 compared to other variables.
- Label has very poor correlation with few variables like cnt_loans90, fr_da_rech90, medianmarechprebal30 etc as we can observe in the above table and map.

Removing Outliers :

```
In [123]: for col in outliers:
              from scipy.stats import zscore
              z=np.abs(zscore(df[col]))

In [124]: Threshold=3
          print(np.where(z>3))
(array([    22,     50,    212, ..., 185999, 186045, 186113], dtype=int64),)

In [125]: df_new=df[(z<3)]
          df_new
```

Out[125]:

	label	aon	daily_decr90	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_rech30	fr_ma_rech30	sumamnt_ma_rech30	me
0	0	272.0	3065.150000	260.13	2.0	0.0	1539	2	21.0	3078.0	
1	1	712.0	12124.750000	3691.26	20.0	0.0	5787	1	0.0	5787.0	
2	1	535.0	1398.000000	900.13	3.0	0.0	1539	1	0.0	1539.0	
3	1	241.0	21.228000	159.42	41.0	0.0	947	0	0.0	0.0	
4	1	947.0	150.619333	1098.90	4.0	0.0	2309	7	2.0	20029.0	
...	...	...	...	...	...	...	...	...	...	...	
209585	1	793.0	5356.210000	640.20	2.0	0.0	2309	4	16.0	7696.0	
209587	1	239.0	14704.900000	16775.60	8.0	0.0	3178	3	3.0	12143.0	
209588	1	404.0	151.872333	1089.19	1.0	0.0	4048	3	2.0	10404.0	
209591	1	1732.0	12574.370000	984.58	2.0	38.0	773	5	4.0	12154.0	
209592	1	1581.0	4534.820000	631.20	13.0	0.0	7526	2	1.0	9065.0	

So the outliers are removing from the dataset using Z-score method and new data (df_new) set is created, This new dataset has 182937 rows and 36 columns. Hence total data we lost

during this process is 1.8% which is less than 10% and tells it is appropriate to remove the outliers using z-score method.

Removing Skewness from the data :

```
In [130]: from sklearn.preprocessing import PowerTransformer
          scaler = PowerTransformer(method='yeo-johnson')
```

```
In [131]: for col in df_new.skew().index:
              if col in df_new.columns:
                  if df_new[col].skew()>0.55:
                      df_new[col]=scaler.fit_transform(df_new[[col]].values)
```

To remove the skewness we have used PowerTransformer – yeo-johnson method as show in the above picture.

Splitting input and target variables :

```
In [132]: x=df_new.drop(['label'],axis=1)
          x.sample()
```

Out[132]:

	aon	daily_decr90	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_rech30	fr_ma_rech30	sumamnt_ma_rech30	mediana
28933	0.341246	-0.752582	-0.585531	0.081171	-0.075718	1.904838	-0.729135	-1.088976	0.516673	

```
In [133]: y=df_new['label']
          y
```

```
Out[133]: 0         0
          1         1
          2         1
          3         1
          4         1
                   ..
          209585    1
          209587    1
          209588    1
          209591    1
          209592    1
          Name: label, Length: 182937, dtype: int64
```

Checking for multicollinearity :

```
In [135]: from statsmodels.stats.outliers_influence import variance_inflation_factor

In [136]: def vif_calc():
              vif=pd.DataFrame()
              vif["VIF Factor"]=[variance_inflation_factor(x.values,i)for i in range(x.shape[1])]
              vif["Features"]=x.columns
              print(vif)

In [137]: vif_calc()
```

	VIF Factor	Features
0	1.008229	aon
1	2.389239	daily_decr90
2	1.317027	rental90
3	1.010885	last_rech_date_ma
4	1.053525	last_rech_date_da
5	6.628861	last_rech_amt_ma
6	54.941151	cnt_ma_rech30
7	1.968078	fr_ma_rech30
8	90.995425	sumamnt_ma_rech30
9	19.520529	medianamnt_ma_rech30
10	1.230565	medianmarechprebal30
11	57.804235	cnt_ma_rech90
12	1.852911	fr_ma_rech90
13	82.487816	sumamnt_ma_rech90
14	19.009701	medianamnt_ma_rech90
15	1.315274	medianmarechprebal90
16	1.839131	cnt_da_rech30
17	1.231929	fr_da_rech30
18	1.876680	cnt_da_rech90
19	1.468052	fr_da_rech90

Once we check the multicollinearity in the data, we can see which are the variables that are causing multicollinearity and by observing the correlation table we can decide on which variable is creating least impact on Label and drop them.
In the same format we have dropped few variables like sumamnt_ma_rech30, amnt_loans90, sumamnt_ma_rech90, cnt_ma_rech30, daily_decr30, rental30, amnt_loans30, medianamnt_loans30, payback30. So once after all these process we can see that the dataset we have is 24 columns and 182937 rows.

Standardizing the data and diving train and test data :

```
In [146]: #we can now scale the data using standard scaler
          from sklearn.preprocessing import StandardScaler
          scale=StandardScaler()
          x = pd.DataFrame(scale.fit_transform(x), columns=x.columns)

In [147]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25,random_state=42)

In [148]: x_train.shape,x_test.shape,y_train.shape,y_test.shape

Out[148]: ((137202, 23), (45735, 23), (137202,), (45735,))
```

As we discussed before we are going to use the below algorithms and build the model

- Logistic Regression
- Decision Tree Classifier
- KNeighbors Classifier
- Random Forest Classifier

In [149]:
```
lg=LogisticRegression()
dtc=DecisionTreeClassifier()
knn=KNeighborsClassifier()
rf=RandomForestClassifier()
```

In [150]:
```
model=[lg,dtc,knn,rf]
for m in model:
    m.fit(x_train,y_train)
    m.score(x_train,y_train)
    predm=m.predict(x_test)
    print("Accuracy score of ",m,"is : ")
    print(accuracy_score(y_test,predm))
    print(confusion_matrix(y_test,predm))
    print(classification_report(y_test,predm))
    print('\n')
```

Scores of these models were as :

In this problem, the data is imbalanced. So we can't use accuracy as a error metric. When data is imbalanced we can use Log loss, F1-score and AUC. We can now consider a ccuracy and F1-score to finalize the model.

We can see the accuracy and F1-scores of the model as :

Model	Accuracy	F1-score
LogisticRegression()	87.46%	38%(0) 93%(1)
DecisionTreeClassifier()	85.20%	48%(0) 91%(1)
KNeighborsClassifier()	88.84%	51%(0) 94%(1)
RandomForestClassifier()	90.35%	56%(0) 95%(1)

From the above table we can observe that Random Forest Classifier is the model which is g iving best score and F1-Score, Hence we can consider this model to be best as of now.

Applying Cross Validation for the same models :

```
In [151]: from sklearn.model_selection import cross_val_score

In [152]: for m in model:
              score=cross_val_score(m,x_train,y_train,cv=5)
              print("Score of : ",m )
              print(score*100)
              print(score.mean()*100)
              print(score.std()*100)
              print('\n')

Score of :  LogisticRegression()
[87.44214861 87.43486025 87.35787172 87.29591837 87.43804665]
87.39376911801125
0.058056779385831855


Score of :  DecisionTreeClassifier()
[85.17182318 85.29208119 85.22594752 85.22230321 85.21865889]
85.2261627995308
0.03839083152412566


Score of :  KNeighborsClassifier()
[88.64837287 88.80871688 88.84110787 88.70626822 88.85204082]
88.77130133316506
0.08016539448675301


Score of :  RandomForestClassifier()
[90.22265952 90.40122445 90.33892128 90.41909621 90.39358601]
90.35509749355708
0.07142322100550272
```

Even with CV we can observe that Random Forest Classifier is giving the best mean score of 90.35%.

Parameter tuning for Random Forest Classifier :

```
In [153]: # Parameter tuning for RandomForestClassifier()

          from sklearn.model_selection import GridSearchCV

In [154]: rf=RandomForestClassifier()
          params={'criterion':['gini','entropy','log_loss']}

In [155]: grd=GridSearchCV(estimator=rf,param_grid=params,cv=5)

In [156]: grd.fit(x_train,y_train)

Out[156]: GridSearchCV(cv=5, estimator=RandomForestClassifier(),
                       param_grid={'criterion': ['gini', 'entropy', 'log_loss']})
```

Best Score : 0.9035509733419052
Best Param : ‘criterion’: ‘gini’

Saving the final model with the above parameters :

In [159]:
```
final_model=RandomForestClassifier(criterion='gini')
final_model.fit(x_train,y_train)
pred=final_model.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.9037498633431726
[[ 2808  3524]
 [  878 38525]]
              precision    recall  f1-score   support

           0       0.76      0.44      0.56      6332
           1       0.92      0.98      0.95     39403

    accuracy                           0.90     45735
   macro avg       0.84      0.71      0.75     45735
weighted avg       0.89      0.90      0.89     45735
```

In [160]:
```
# We can save the model now
import joblib
joblib.dump(final_model,'Micro_Credit_Defaulter.obj')
```

Out[160]:
```
['Micro_Credit_Defaulter.obj']
```

Loading the saved model and predicting the values

In [161]:
```
Micro_Credit_Defaulter=joblib.load('Micro_Credit_Defaulter.obj')
```

In [162]:
```
pred=Micro_Credit_Defaulter.predict(x_test)
print("Predicted values :",pred)
```

```
Predicted values : [1 0 0 ... 1 1 1]
```

Now we can predict the values and compare it with the original values using the saves final model.

In [164]:
```
# Making a DataFrame of Predicted values and Original values
df1=pd.DataFrame({'Predicted values':pred,'Original values':y_test})
df1
```

Out[164]:

	Predicted values	Original values
25388	1	1
97381	0	0
66932	0	0
100447	1	1
205069	1	1
...	...	...
132203	1	1
109769	1	1
139548	1	1
175620	1	1
83014	1	0

45735 rows × 2 columns

CONCLUSION

From the Exploratory Data Analysis, we could generate insight from the data. How each of the features relates to the target. Also, while training the model we could observe that Random Forest Classifier was performing well with the accuracy 90.35%.

From the EDA we could also observe that there were some duplicates present in the dataset which had to be removed, Label had better correlation with few variables like daily_decr30, amnt_loans90, cnt_loans30, cnt_ma_rech90 compared to other variables.

Few variables like cnt_loans90, fr_da_rech90, medianmarechprebal30 had poor correlation with target variable.

Cleaning of the data included few techniques like removing outliers, treating skewed data, dropping the variables which were creating multicollinearity and finally the data was scaled.

While building the model we could observe that it is always good to build 4 to 5 models for the same train test data and compare the scores of the models then pick the best model instead of sticking onto single model.

To make the model more accurate I think it is good to use SMOTE and PCA techniques if applicable. As of now we could finalize **Random Forest Classifier** with **gini** as the criterion as the best model which was giving **90.35%** accuracy score which was best compared to other models.