MICRO-CREDIT DEFAULTER CLASSIFICATION USING ML

PROBLEM DEFINITION:

The company is 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. The company know the importance of communication so they have also focused on providing product and services to low-income families. To do this, they have a collaboration with MFI to provide micro credit on mobile balances to be paid back in 5 days. The customer is considered as defaulter if he fails to pay the sum of money within the stipulated period of time.
We can see there is huge population with no financial record is there to access all that remote areas to help them we need to come with more ground projects to help such population.In this whole scenario we encounter that there is a must use of Artificial Intelligence because as we can see that there is high variation of defaulters and non-defaulters.By the use of AI we can analyse patterns of peoples who are taking micro credit by the help of their history of recharge, date of recharge, daily usage, there payment history of loans etc. After putting all these constrains in AI and modelling with different models we will come to a point where we can suggest a point of view what more improvement is needed or who all are the ones which will be defaulter and should be stop from credit facility.

DATA ANALYSIS:

The company shared around 2 lakh data of their customer with different transaction behaviour to understand and to predict their future behaviour. The data is been provided in CSV format with 37 different variables in different columns and 209593 rows.

Next moving to column name msisdn, pcircle and pdate which are of object type and cannot give any help in best performance of model. Pcircle means a code given to certain areas which are not showing any relevance with population mobile credit taking.Msisdn,pcircle was removed along columns and from pdate the p month and pday was extracted using datetime functions.

```
df.shape
```

```
(209593, 36)
```

```
df.columns
```

```
Index(['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')
```

```
df.describe()
```

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.00000	209593.000000	209593.000000
mean	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.84780	3712.202921	2064.452797
std	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.89223	53374.833430	2370.786034
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.00000	-29.000000	0.000000
25%	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.00000	0.000000	770.000000
50%	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.00000	0.000000	1539.000000
75%	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.00000	0.000000	2309.000000
max	1.000000	999860.755200	265926.000000	320630.000000	198926.110000	200148.110000	998650.37770	999171.809400	55000.000000

- We can notice there are no null values in the dataset..but many columns have minimum values as 0.
- Almost all columns have right skewness and outliers present in it.
- AON has negative value,age cannot be negative.
- Certain features have amount in negative which needs to be checked for an anomaly.

Most of the data in the dataset was full of outliers. Those outliers were corrected by replacing them with Q3+1.5(IQR) if it is more than Q3+1.5(IQR). The data was also skewed. Some of them were negatively whereas some are positively skewed. All the skewed data was corrected using square root transformation where ever applicable. In some places the minimum values were negative which also seem to be abnormal in that case. Hence, it was replaced by Q1-1.5(IQR) if it is below the minimum value.Columns having negative values were converted to absolute values. We need to convert the p date from object to numerical by using date-time function and extracting the day and month of the year as the year remains the same for all the records.

```
#changing the pdate from object format to numerical values of date and month separately.
#Year is not taken into consideration as all records are of same year i.e. 2016.
df["P_month"]=pd.to_datetime(df.pdate, format="%d-%m-%Y").dt.month
df["P_Day"]=pd.to_datetime(df.pdate, format="%d-%m-%Y").dt.day
```

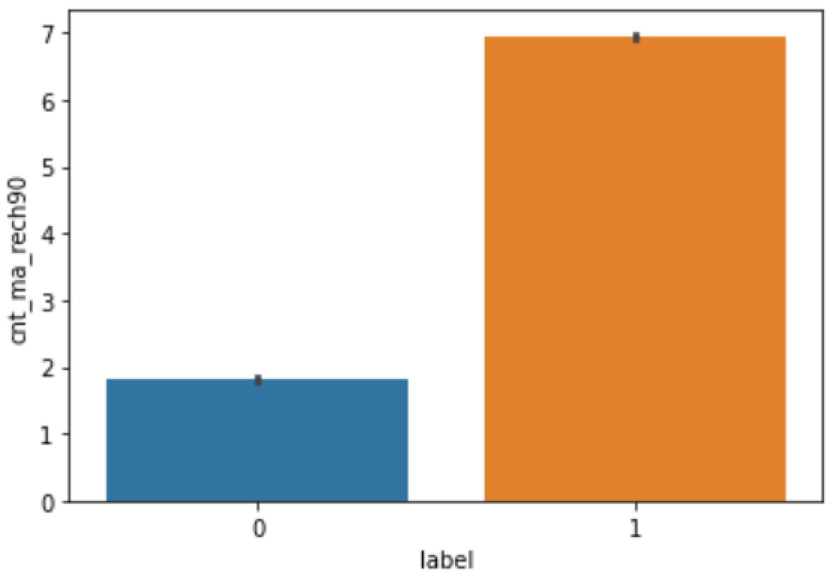
```
#removing the pdate column as we have got the information that is important for our modelling.
df.drop(['pdate'],axis=1,inplace=True)
```

EDA :

We can plot the univariate,bivariate and multivariate analysis of the features with respect to the output label.

```
sns.barplot(x="label",y="cnt_ma_rech90",data=df)
```

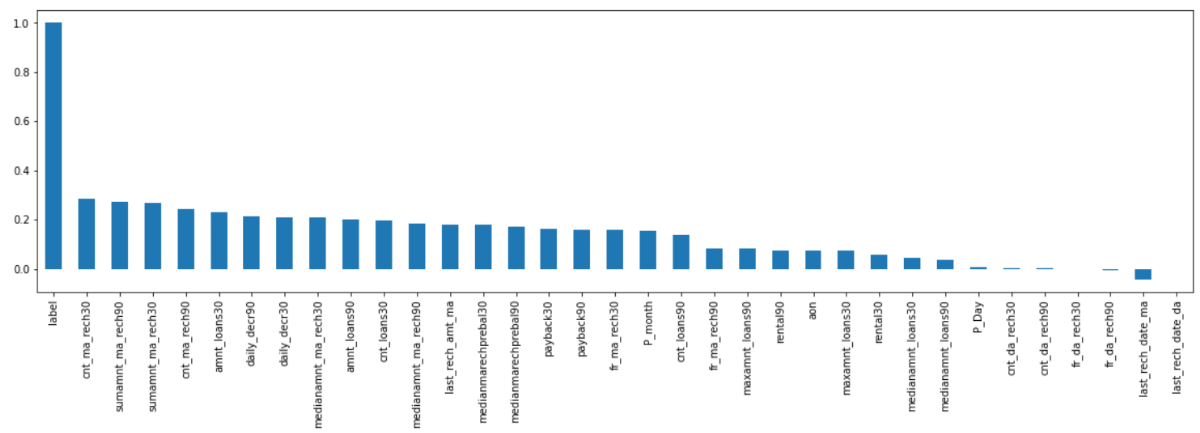
```
<AxesSubplot:xlabel='label', ylabel='cnt_ma_rech90'>
```



- Less no of times main account recharged more chances of credit default.

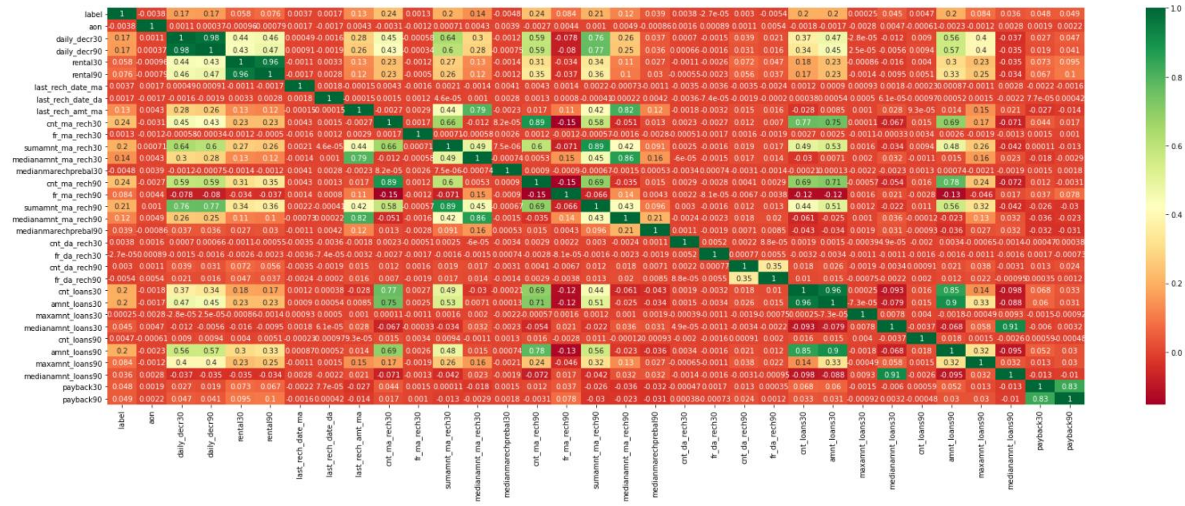
-

cnt_ma_rech30,sumamt_ma_rech90,sumamnt_ma_rech30,cnt_ma_rech90,amnt_loans30 are the main independent features for the label.



Similarly we can check the correlation of the features with the output label and other independent features.

```
plt.figure(figsize = (30,10))
sns.heatmap(df.corr(), annot = True, cmap = "RdYlGn")
plt.show()
```



PRE-PROCESSING PIPELINE:

```
df.skew()
```

```
label                  -2.270254
aon                     0.952374
daily_decr30            1.239238
daily_decr90            1.239001
rental30                4.560510
rental90                4.467282
last_rech_date_ma       1.133958
last_rech_date_da       0.000000
last_rech_amt_ma        1.003446
cnt_ma_rech30           0.904157
fr_ma_rech30            1.253282
sumamnt_ma_rech30       1.080771
medianamnt_ma_rech30    0.728219
medianmarechprebal30    1.105328
cnt_ma_rech90           2.451653
fr_ma_rech90            2.285423
sumamnt_ma_rech90       1.126421
medianamnt_ma_rech90    0.763904
medianmarechprebal90    1.053444
cnt_da_rech30          13.257074
fr_da_rech30           13.250037
cnt_da_rech90          27.267278
fr_da_rech90           26.081188
cnt_loans30             2.617377
amnt_loans30            1.231090
maxamnt_loans30         1.435587
medianamnt_loans30      4.551043
cnt_loans90             6.190825
amnt_loans90            3.150006
maxamnt_loans90         1.678304
```

We need to remove the skewness.For removing the skewness we have used square root transformation here.

```
1  df1.skew()
```

```
label                  -2.270254
aon                     0.318991
daily_decr30            0.575904
daily_decr90            0.598369
rental30                1.294750
rental90                1.358978
last_rech_date_ma       0.312328
last_rech_date_da       0.000000
last_rech_amt_ma       -0.237208
cnt_ma_rech30          -0.141319
fr_ma_rech30            0.408171
sumamnt_ma_rech30       0.081148
medianamnt_ma_rech30   -0.563881
medianmarechprebal30    0.261988
cnt_ma_rech90           0.555533
fr_ma_rech90            1.038194
sumamnt_ma_rech90       0.213672
medianamnt_ma_rech90   -0.561451
medianmarechprebal90    0.193764
cnt_da_rech30          10.651890
fr_da_rech30           12.504925
cnt_da_rech90           8.187518
fr_da_rech90           19.346530
cnt_loans30             1.066586
amnt_loans30            0.543700
maxamnt_loans30        -1.570323
medianamnt_loans30      3.701226
```

After removing the skewness we can see there is a huge imbalance in the dataset.We have used standard scaling technique for scaling the records.As the data is very expensive rather than using under sampling the majority class I opted to apply SMOTE technique to re sample the minority class by adding synthetic minorities.

```
df1['label'].value_counts()
```

```
1.0    183431
0.0     26162
Name: label, dtype: int64
```

```
y=df1['label']
```

```
x=df1.drop(['label'],axis=1)
```

```
x.shape
```

```
(209593, 34)
```

```
y.shape
```

```
(209593,)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
```

```
from imblearn.over_sampling import SMOTE
```

```
smt=SMOTE()
x_s,y_s=smt.fit_resample(x,y)
```

BUILDING MACHINE LEARNING MODELS:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score,accuracy_score,confusion_matrix,classification_report

from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV

from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score
```

We have used five models to check the best performing model and later cross validation is done.

```
x_train,x_test,y_train,y_test=train_test_split(x_s,y_s,random_state=42,test_size=0.27)
```

```
log_class1 = LogisticRegression(penalty='l2',C=1.0,class_weight='balanced',n_jobs=-1)

training1 = log_class1.fit(x_train,y_train)
pred1 = log_class1.predict(x_test)
CM1 = confusion_matrix(y_test,pred1)
CR1 = classification_report(y_test,pred1)
acc1 = accuracy_score(y_test,pred1)

print('confusion metrics :', '\n', CM1)
print('classification report ', '\n', CR1)
print('accuracy score: ', '\n' , acc1)
```

```
confusion metrics :
 [[39582  9819]
 [12553 37099]]
classification report
               precision    recall  f1-score   support

         0.0       0.76      0.80      0.78     49401
         1.0       0.79      0.75      0.77     49652

    accuracy                           0.77     99053
   macro avg       0.77      0.77      0.77     99053
weighted avg       0.78      0.77      0.77     99053

accuracy score:
 0.7741411163720433
```

```
k=StratifiedKFold(n_splits=10,shuffle=False)
lg_score=cross_val_score(log_class1,x_train,y_train,cv=k,scoring='f1_weighted',n_jobs=-1)
print("cross validation score for Logistic Regression:",np.mean(lg_score))
```

```
cross validation score for Logistic Regression: 0.7762776740595008
```

```
rf_class1 = RandomForestClassifier()
rf_model1 = rf_class1.fit(x_train,y_train)
rf_pred1 = rf_class1.predict(x_test)
rf_CM1 = confusion_matrix(y_test,rf_pred1)
rf_CR1 = classification_report(y_test,rf_pred1)
rf_acc1 = accuracy_score(y_test,rf_pred1)

print('confusion metrics :', '\n', rf_CM1)
print('classification report ', '\n', rf_CR1)
print('accuracy score: ', '\n' , rf_acc1)
```

```
confusion metrics :
 [[47201  2200]
 [ 2624 47028]]
classification report
               precision    recall  f1-score   support

         0.0       0.95      0.96      0.95     49401
         1.0       0.96      0.95      0.95     49652

    accuracy                           0.95     99053
   macro avg       0.95      0.95      0.95     99053
weighted avg       0.95      0.95      0.95     99053

accuracy score:
 0.95129879963252
```

```
k=StratifiedKFold(n_splits=10,shuffle=False)
rf_score=cross_val_score(rf_class1,x_train,y_train,cv=k,scoring='f1_weighted',n_jobs=-1)
print("cross validation score for Random Forest Classifier:",np.mean(rf_score))
```

```
cross validation score for Random Forest Classifier: 0.9511771388457699
```

```
dt_clf=DecisionTreeClassifier()
dt_model1 = dt_clf.fit(x_train,y_train)
dt_pred1 = dt_clf.predict(x_test)
dt_CM1 = confusion_matrix(y_test,dt_pred1)
dt_CR1 = classification_report(y_test,dt_pred1)
dt_acc1 = accuracy_score(y_test,dt_pred1)

print('confusion metrics :', '\n', dt_CM1)
print('classification report ', '\n', dt_CR1)
print('accuracy score: ', '\n' , dt_acc1)
```

```
confusion metrics :
 [[45542  3859]
 [ 4633 45019]]
classification report
               precision    recall  f1-score   support

         0.0       0.91      0.92      0.91     49401
         1.0       0.92      0.91      0.91     49652

    accuracy                           0.91     99053
   macro avg       0.91      0.91      0.91     99053
weighted avg       0.91      0.91      0.91     99053

accuracy score:
 0.9142681190877611
```

```
k=StratifiedKFold(n_splits=10,shuffle=False)
dt_score=cross_val_score(dt_clf,x_train,y_train,cv=k,scoring='f1_macro',n_jobs=-1)
print("cross validation score for decision tree Classifier:",np.mean(dt_score))
```

```
cross validation score for decision tree Classifier: 0.9120783492494642
```

```
xgb_clf=XGBClassifier(learning_rate=0.001,n_estimators=100)
xgb_model1 = xgb_clf.fit(x_train,y_train)
xgb_pred1 = xgb_clf.predict(x_test)
xgb_CM1 = confusion_matrix(y_test,xgb_pred1)
xgb_CR1 = classification_report(y_test,xgb_pred1)
xgb_acc1 = accuracy_score(y_test,xgb_pred1)
print('confusion metrics :', '\n', xgb_CM1)
print('classification report ', '\n', xgb_CR1)
print('accuracy score: ', '\n' , xgb_acc1)
```

```
[12:47:13] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.0/
0, the default evaluation metric used with the objective 'binary:logistic' was ch
t eval_metric if you'd like to restore the old behavior.
confusion metrics :
 [[39054 10347]
 [ 4731 44921]]
classification report
               precision    recall  f1-score   support

         0.0       0.89      0.79      0.84     49401
         1.0       0.81      0.90      0.86     49652

    accuracy                           0.85     99053
   macro avg       0.85      0.85      0.85     99053
weighted avg       0.85      0.85      0.85     99053

accuracy score:
 0.8477784620354759
```

By studying the different models and its metrices we found out random forest classifier is working best for the model.

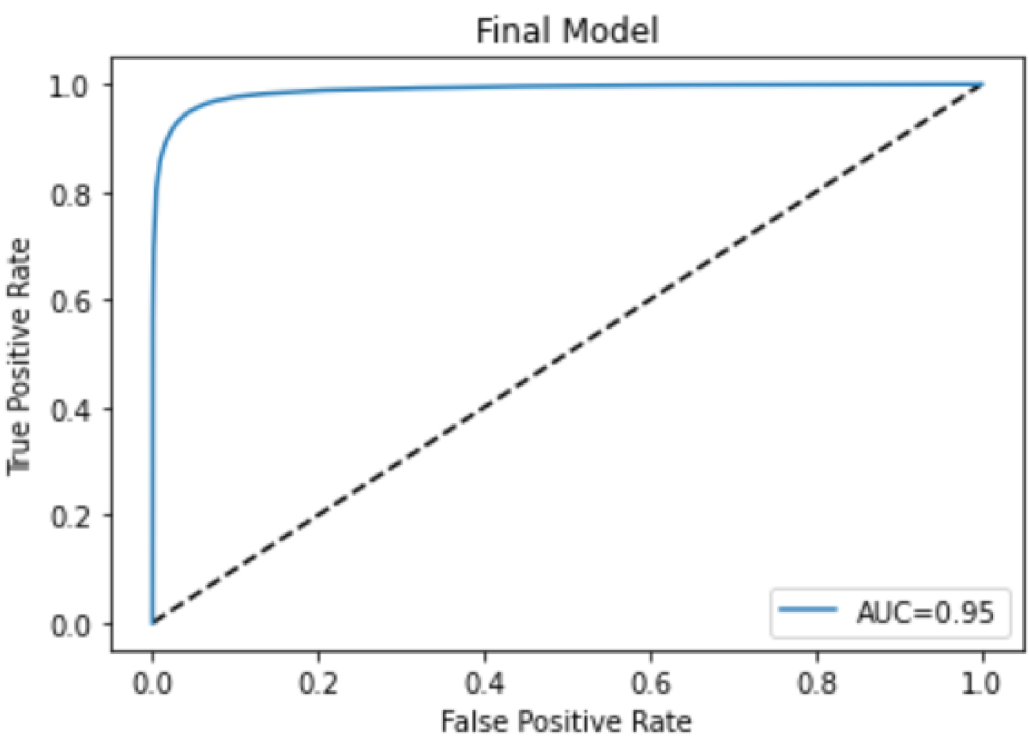
Hence we hypertuned its parameters using RandomizedSearchCV,as it is faster.

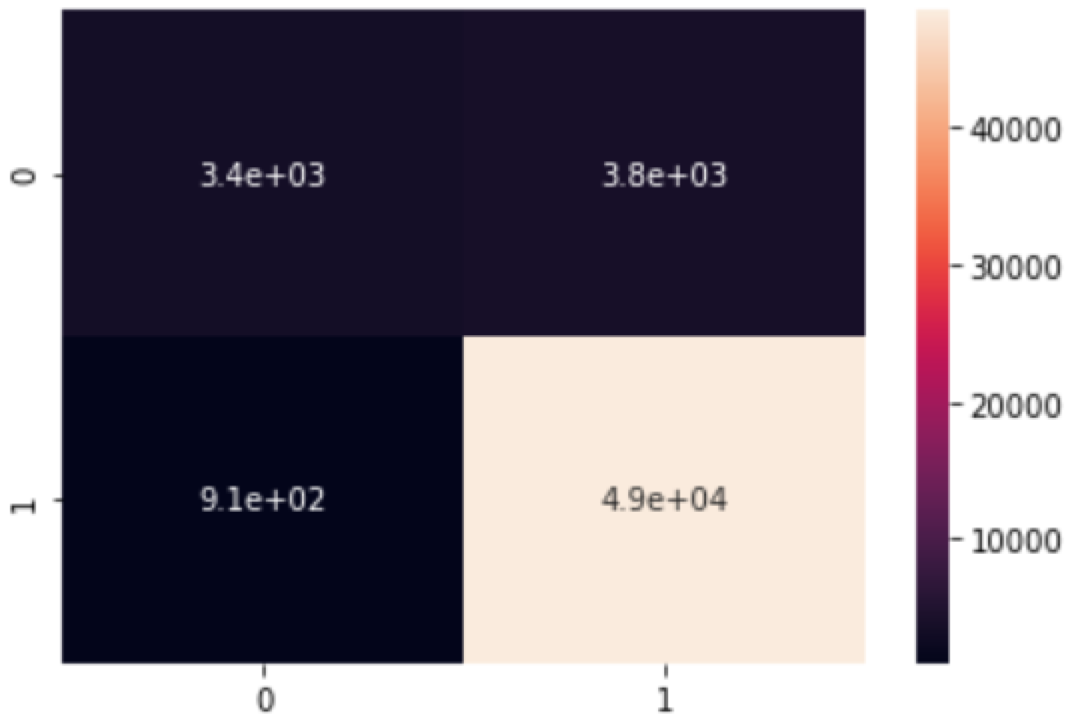
```
parameter={'n_estimators':[100,300,400],'class_weight':['balanced', 'balanced_subsample'],
           'criterion':['gini','entropy'],
           'max_features':['auto','sqrt']}
RCV=RandomizedSearchCV(rf_class1,parameter,cv=5,scoring='f1_macro',n_jobs=-1)
RCV.fit(x_train,y_train)
RCV.best_params_
```

```
{'n_estimators': 300,
 'max_features': 'auto',
 'criterion': 'gini',
 'class_weight': 'balanced'}
```

```
final_mod=RandomForestClassifier(n_estimators=300,max_features= 'auto',criterion='gini',class_weight='balanced')
final_mod.fit(x_train,y_train)
pred=final_mod.predict(x_test)
fin_CM = confusion_matrix(y_test,pred)
fin_CR = classification_report(y_test,pred)
fin_acc = accuracy_score(y_test,pred)

print('confusion metrics :', '\n', fin_CM)
print('classification report ', '\n', fin_CR)
print('accuracy score: ', '\n' , fin_acc)
```

```
confusion metrics :
 [[47183  2218]
 [ 2512 47140]]
classification report
               precision    recall  f1-score   support

         0.0       0.95      0.96      0.95     49401
         1.0       0.96      0.95      0.95     49652

    accuracy                           0.95     99053
   macro avg       0.95      0.95      0.95     99053
weighted avg       0.95      0.95      0.95     99053

accuracy score:
 0.9522477865385198
```





Saving the best model and using it for prediction:

```
#Saving the best model
import joblib
joblib.dump(final_mod,'Micro_credit.obj')
```

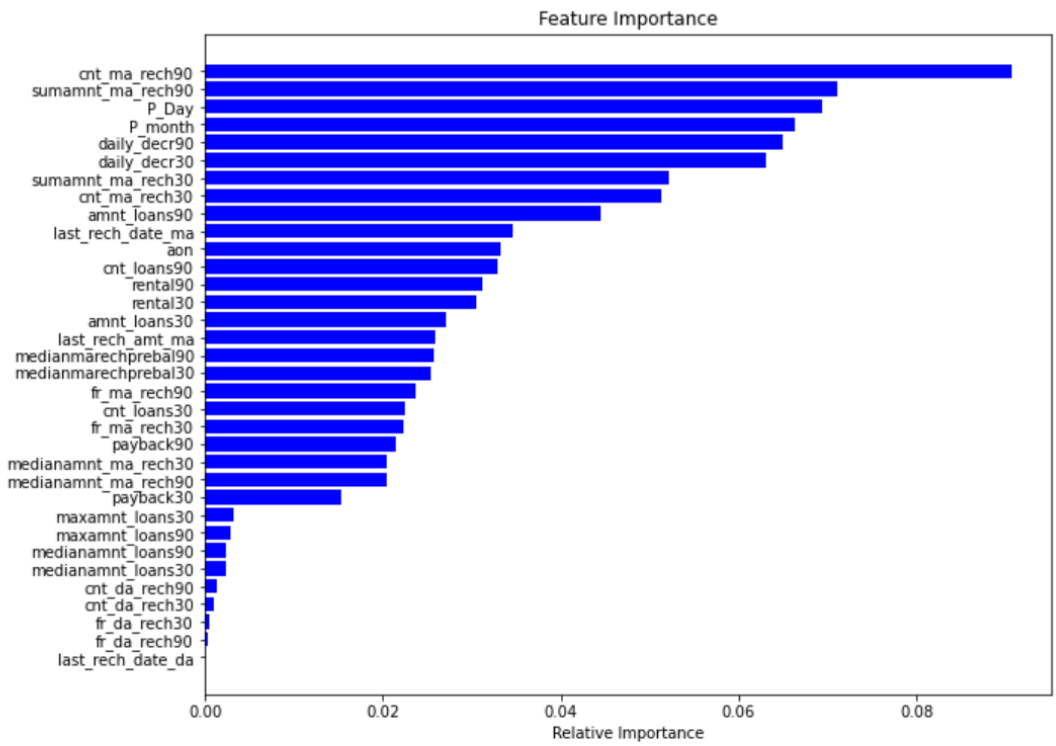
['Micro_credit.obj']

```
a=np.array(y_test)
predicted=np.array(final_mod.predict(x_test))
df_com=pd.DataFrame({"original":a,"predicted":predicted},index=range(len(a)))
df_com
```

	original	predicted
0	0.0	0.0
1	1.0	1.0
2	1.0	1.0
3	1.0	1.0
4	0.0	0.0
...	...	...
99048	0.0	0.0
99049	1.0	1.0
99050	0.0	0.0
99051	0.0	0.0
99052	0.0	0.0

99053 rows × 2 columns

To check the best features who are contributing for the classification.



CONCLUDING REMARKS:

The dataset was full of outliers, skewness and unbalanced data which was the biggest challenge to overcome. Hence data cleaning was very important to get proper prediction.Feature scaling was done by help of standard scaler.And the imbalance in the dataset was handled using SMOTE technique.For cross validation of the model K-Fold cross validation was used. I have used Logistic Regression, K-Nearest Neighbour,XG-Boost, Decision Tree and Random Forest Classifier. Among the five algorithms Random Forest Classifier gave the best outcome. We used RandomizedSearchCV for hyper parameter tuning of the best model as it is fast as compared to GridSearchCV.

The solution can be applied to the customer having a transaction history but the model may not perform well with customer having new profile and no transaction history. Nevertheless, the model will perform well with customer having transaction history and can predict whether a person will be a defaulter or non-defaulter. Hence, we can say that this statistical model will be helpful in future for the prediction of micro credit defaulter and non-defaulter customer.

By-Ashish Kumar Samal,Data Scientist