Micro-Credit Defaulter Case

Submitted by:

Bilam Roy

ACKNOWLEDGMENT

I would like to express my gratitude towards Flip-Robo for giving me the platform to show case my talent and for their constant guidance and support.

I would also like to thank some of the online platforms which I found useful to clarify my doubts and those helped me to complete my project.

References-

- https://colab.research.google.com
- https://scikit-learn.org
- https://www.kaggle.com
- http://github.com
- https://towardsdatascience.com

INTRODUCTION

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on. 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. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be6(in Indonesian Rupiah), while, for the loan amount of 10(in Indonesian Rupiah), the payback amount should be 12(in Indonesian Rupiah).

But it has been observed that the customers are failing to return the loaned amount and is believed to be a defaulter if he/she deviates from the path of paying back the loaned amount within the time duration of 5 days.

So we need to 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.

This model will further help the Microfinance Institution (MFI) to decide for the future investments and improvement in selection of customers.

Analytical Problem Framing

The sample data is provided to us from our client database.

1) Analysing the columns present in the data after loading it to jupyter notebook

```
#acquiring the data
micro_cr=pd.read_csv("/content/Data file.csv")
```

```
#analysing the data
print(micro_cr.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')
```

2) Previewing the top 5 rows of the data

```
#previewing the top 5 rows of the data
micro_cr.head()
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0

3) Analysing the basic information from the dataset (if there are any null values present, what is the data type)

```
#extracting the general information from the dataset
micro_cr.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
```

4) Statistical Report of the dataset (total data count, mean, std, minimum value, maximum value). This report helps us to decide which transformation will help to improve the model's accuracy, outliers, range of a particular column.

```
#statastical report
data_df.describe()
```

	label	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
count	366862.000000	366862.000000	366862.000000	366862.000000	366862.000000	366862.000000	366862.000000	366862.000000	366862.000000	366862.000000	366862.000000
mean	0.500000	8499.628789	3616.610693	4019.955115	2403.588683	2983.933776	3576.181481	3661.732708	1707.527844	2.829557	3602.901010
std	0.500001	77678.013689	7538.280044	8827.253660	4073.433896	5223.072178	52473.497688	53057.595890	2283.393032	3.724733	52363.603997
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	0.000000	0.000000	0.000000
25%	0.000000	209.000000	12.400000	12.650000	169.040000	182.700000	1.000000	0.000000	770.000000	0.000000	0.000000
50%	0.500000	471.000000	518.900000	525.280000	876.960000	1026.600000	3.000000	0.000000	777.000000	2.000000	0.000000
75%	1.000000	916.000000	3835.504000	3950.000000	2866.327500	3511.652500	8.000000	0.000000	1547.000000	4.000000	4.000000
max	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	55000.000000	203.000000	999606.368132

5) Checking for the class distribution of the Target

```
#class distribution of the target value
micro_cr.groupby("label").size()
```

```
label
0     26162
1    183431
dtype: int64
```

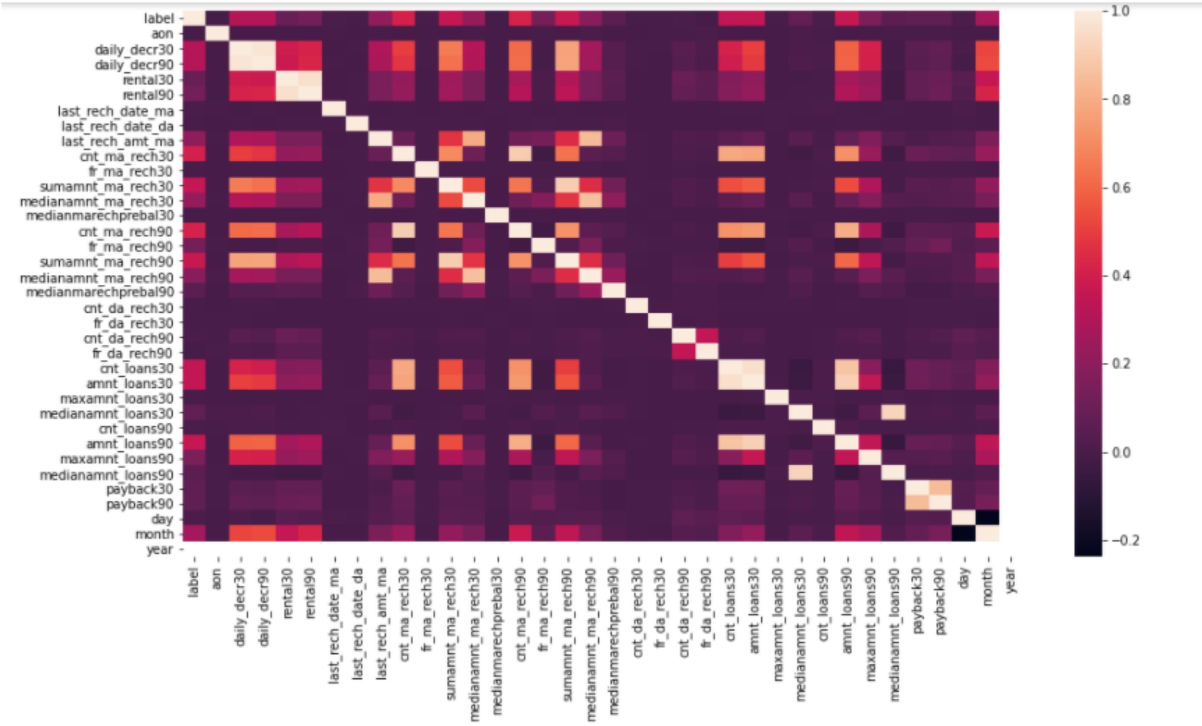
6) As the dataset was imbalanced, hence it was up-sampled

```
from sklearn.utils import resample
# Separate majority and minority classes
df_majority = micro_cr[micro_cr.label==1]
df_minority = micro_cr[micro_cr.label==0]

# Upsample minority class
df_minority_upsampled = resample(df_minority,
                                 replace=True,     # sample with replacement
                                 n_samples=183431, # to match majority class
                                 random_state=123) # reproducible results

# Combine majority class with upsampled minority class
micro_cr_df = pd.concat([df_majority, df_minority_upsampled])

# Display new class counts
micro_cr_df.label.value_counts()
```

```
1    183431
0    183431
Name: label, dtype: int64
```

7) Correlation of the columns

```
#checking for the correlation
corr_hmap=micro_cr_df.corr()
plt.figure(figsize=(15,8))
sns.heatmap(corr_hmap)
```



This gives the relationship of each column with the other columns, positively correlated are those are having light gradient and vice versa. This helps to decide which columns can be dropped or which column has what kind of dependency on others. (Here year column can be dropped as it is positively related to every column as not adding any exclusive information to the data)

8) Dropping down the columns that were not required

```
#droping the column 'Unnamed: 0' as it was same as indexing
micro_cr = micro_cr.drop(['Unnamed: 0'], axis=1)
```

```
micro_cr_df["day"] = pd.to_datetime(micro_cr_df.pdate, format="%Y-%m-%d").dt.day

micro_cr_df["month"] = pd.to_datetime(micro_cr_df.pdate, format = "%Y-%m-%d").dt.month

micro_cr_df["year"] = pd.to_datetime(micro_cr_df.pdate, format = "%Y-%m-%d").dt.year
```

```
#droping the column 'pdate' as it was expanded seperately
micro_cr_df = micro_cr_df.drop(['pdate'], axis=1)
```

```
data_df=micro_cr_df.drop(['msisdn','year','pcircle'],axis=1)
print(data_df.shape)
```

```
(366862, 35)
```

Note:

- as msisdn was nothing but phone number which was not adding to any significant positive result to the model, hence dropped it

- as year and pcircle was same for each and every entry hence dropped it as it would not put any effect on the modelling.

9) Removed the outliers using z-score (of degree 4)

```
#removing outliers
from scipy import stats
z = np.abs(stats.zscore(data_df))
print(data_df.shape)
data_df=data_df.iloc[(z<4).all(axis=1)]
print(data_df.shape)
```

```
(366862, 35)
(309182, 35)
```

10) Power Transformation was done to minimise the skewness (yeo-johnson was considered as negative data values were present in the dataset)

```
p=PowerTransformer('yeo-johnson')
p.fit_transform(data_df)
```

```
array([[ 1.04864243,  0.36547194,  1.4911547 , ..., -0.73511882,
        -0.42467057,  1.72366699],
       [ 1.04864243,  0.01014193,  0.52444645, ..., -0.73511882,
         0.63574377,  1.72366699],
       [ 1.04864243, -0.68409674, -0.6895228 , ..., -0.73511882,
        -0.99367092, -1.01971093],
       ...,
       [-0.9536139 , -0.45171134,  0.41308487, ..., -0.73511882,
        -0.8425449 ,  0.73393069],
       [-0.9536139 , -0.67594219, -0.78692187, ..., -0.73511882,
         1.72010995, -1.01971093],
       [-0.9536139 , -1.11528984,  1.15613198, ..., -0.73511882,
         0.9487265 ,  0.73393069]])
```

11)Separating the input and the output

```
#Now separating input and output variable
#Predicting
x=data_df.drop(['label'],axis=1)
y=data_df['label']
print(x.shape)
print(y.shape)
```

```
(309182, 34)
(309182,)
```

12)Standardizing the input dataset (Standard-Scaler makes the mean of the distribution 0)

```
#standardizing the input dataset
sc=StandardScaler()
x=sc.fit_transform(x)
x
```

```
array([[ 0.23104995,  1.94672455,  1.68972229, ..., -0.47462975,
        -0.52299253,  2.12531918],
       [-0.13091499, -0.25984725, -0.27305126, ..., -0.47462975,
         0.58464202,  2.12531918],
       [-0.73214489, -0.54313205, -0.52497208, ..., -0.47462975,
        -1.01527455, -0.95898232],
       ...,
       [-0.54809492, -0.33350938, -0.33855786, ..., -0.47462975,
        -0.89220404,  0.58316843],
       [-0.72600989, -0.54482505, -0.52647763, ..., -0.47462975,
         1.93841758, -0.95898232],
       [-1.01844484,  0.71843141,  0.60117458, ..., -0.47462975,
         0.95385354,  0.58316843]])
```

Model/s Development and Evaluation

1) Basic Models Chosen to check the best performing model

```
#Machine Learning Models
models=[]
models.append(('LR', LogisticRegression()))
models.append(('DT', tree.DecisionTreeClassifier()))
models.append(('GNB', GaussianNB()))
models.append(('ETC', ExtraTreesClassifier()))
models.append(('RFC', RandomForestClassifier()))
models.append(('ABC', AdaBoostClassifier()))
models.append(('GBC', GradientBoostingClassifier()))
models.append(('XGB', xgb.XGBClassifier()))
```

2) Training and Test data were split and models were trained and accuracy was evaluated

```
accuracy_results = []
names = []
for name, model in models:
    print(name)
    max_acc_score=0
    for r_state in range(42,50):
        x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=r_state,test_size=0.20)
        model_name=model
        model_name.fit(x_train,y_train)
        y_pred=model_name.predict(x_test)
        accuracy_scr=accuracy_score(y_test,y_pred)
        print("random state: ",r_state," accuracy score: ",accuracy_scr)
        if accuracy_scr>max_acc_score:
            max_acc_score=accuracy_scr
            final_r_state=r_state
    accuracy_results.append(max_acc_score*100)
    print()
    print("max accuracy score at random state:",final_r_state," for the model ",name," is: ",max_acc_score)
    print()
    print()
```

3) Cross-validation parameters were evaluated to choose the best model

```
#cross_val of the models
results = []
names = []
cvs=[]
for name, model in models:
    cv_result=cross_val_score(model, x_train, y_train, cv=5, scoring="accuracy")
    results.append(cv_result)
    names.append(name)
    print("Model name: ",name)
    print("Cross Validation Score(Mean): ",cv_result.mean())
    cvs.append(cv_result.mean()*100)
    print("Cross Validation Score(Std): ",cv_result.std())
    print()
```

4) Choosing the Best Model and evaluating the other parameters like f1 score and confusion matrix.

```
#Choosing the Best Model
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=47,test_size=0.20)
model_name=ExtraTreesClassifier()
model_name.fit(x_train,y_train)
training_scr=model_name.score(x_train,y_train)
y_pred=model_name.predict(x_test)
accuracy_scr=accuracy_score(y_test,y_pred)
cfm=confusion_matrix(y_test,y_pred)
cr=classification_report(y_test,y_pred)
print("training score: ",training_scr)
print("accuracy score: ",accuracy_scr)
print("confusion matrix: ")
print(cfm)
print("classification report: ")
print(cr)
```

5) Hyper-Tuning the model

```
#Hyper-Tuning the Best Model

#Randomized Search CV

from sklearn.model_selection import RandomizedSearchCV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(100, 2000, 10)]
# Number of features to consider at every split
max_features = ['auto']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, 6)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
```

```
# Create the random grid

random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split}
```

```
# Random search of parameters, using 5 fold cross validation,
# search across 100 different combinations
ET_RSCV = RandomizedSearchCV(estimator = ExtraTreesClassifier(), param_distributions = random_grid,scoring='accuracy',
                             n_iter = 5, cv = 5, verbose=2, random_state=47, n_jobs = 1)

ET_RSCV.fit(x_train,y_train)
```

CONCLUSION

1) Matrices evaluated

```
ET_RSCV.best_params_
```

```
{'max_depth': 30,
 'max_features': 'auto',
 'min_samples_split': 15,
 'n_estimators': 1155}
```

```
y_pred=ET_RSCV.predict(x_test)
```

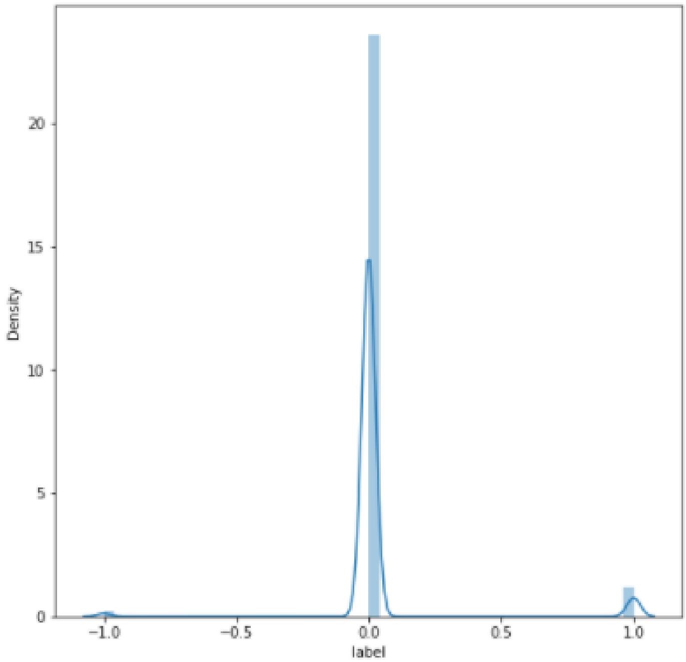
```
ET_RSCV.score(x_train, y_train)
```

```
0.9581151832460733
```

```
ET_RSCV.score(x_test, y_test)
```

```
0.9446771350485955
```

```
plt.figure(figsize = (8,8))
sns.distplot(y_test-y_pred)
plt.show()
```



```python
print("roc_auc_score: ",metrics.roc_auc_score(y_test, y_pred))
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
roc_auc_score:  0.9429162338924262
MAE: 0.055322864951404496
MSE: 0.055322864951404496
RMSE: 0.23520813113369293
```

```python
#saving the model as pickle in a file
pickle.dump(ET_RSCV.best_estimator_, open('Microcredit.pkl','wb'))
```

```python
#loading the model for testing
loaded_model=pickle.load(open('Microcredit.pkl','rb'))
pred=loaded_model.predict(x_test)
```

As ExtraTree Classifier done exceptionally good over the dataset, hence it can be used for predictions.

Note: Basic Libraries that need to imported to perform the model training and testing

```python
#data analysis and wrangling
import pandas as pd
import numpy as np

#visualizing the data
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

#model developing-machine learning
import sklearn
from scipy.stats import zscore                                                  #for removing the outliers
from sklearn.preprocessing import PowerTransformer
from sklearn.preprocessing import StandardScaler                                #for standardizing the input dataset
from sklearn.model_selection import train_test_split                            #to train the model
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score       #for reporting purposes

#boosting techniques
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier, RandomForestClassifier, ExtraTreesClassifier
import xgboost as xgb

#saving the model using joblib
import pickle
#for filtering the warnings
import warnings
warnings.filterwarnings("ignore")
```

Scope for Future Work

As the accuracy level is more than 90%, hence this model can be used to decide selection of customers.

For more information please visit:
https://github.com/bilamroy/FlipRoboProjects

- Key Findings and Conclusions of the Study

  Describe the key findings, inferences, observations from the whole problem.

- Learning Outcomes of the Study in respect of Data Science

  List down your learnings obtained about the power of visualization, data cleaning and various algorithms used. You can describe which algorithm works best in which situation and what challenges you faced while working on this project and how did you overcome that.

- Limitations of this work and Scope for Future Work

  What are the limitations of this solution provided, the future scope? What all steps/techniques can be followed to further extend this study and improve the results.