Micro Credit Defaulter

Submitted by:

Nidhi Singh

Acknowledgment

To solve many problems regarding imbalanced data set medium.com played an important role: https://medium.com
Documentation of different API's was also helpful:

https://matplotlib.org/3.3.1/api/_as_gen/matplotlib.pyplot.html#module - matplotlib.pyplot

https://pandas.pydata.org/pandas-docs/stable/reference/frame.html

https://scikit-learn.org

The below site was reported for understanding what Micro Credit financing companies face in absence of Machine Learning models.

https://www.smartcampaign.org/storage/documents/what_happens_to_microfinance_clients_who_default_eng.pdf

Business Problem

A telecommunication company is providing services and to low-income families by collaborating with an MFI (Microfinance Institution). MFI is an organization that offers financial services to low-income families. This telecommunication company is customer-centric and is very keen on providing quality services to its customers. They understand the importance of communications and thereby providing seamless service to their customers.

To provide swift services this telecommunication company is providing micro- credit in mobile balances which need to be paid back by 5 days. For the loan amount of 5 (in Indonesian Rupiah) payback amount should be 6 (in Indonesian Rupiah), while for a loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah).

The client requires to detect the defaulters who are taking disadvantage of this service by not paying back the loan amount in time.
With the given problem statement we are given several features like Average daily amount spent in 30 days, the median amount of recharge done in 30 days, Frequency of recharge in 30 days, Average payback time of users in the last 30days, and many more. Also, there is a target column which indicates whether the user paid back the credit amount within 5 days of issuing the loan where 1 indicates Non- Defaulter (Loan is paid) and 0 indicates Defaulter (Loan is not paid)

We are expected to create a model that identifies whether the customer will be a defaulter or not with a given set of parameters.

Conceptual Background of the Problem

To dive deep into this domain we need to understand the importance of some features from the pool of features. We need to verify the authenticity of the values which are unsure of given definitions for features like Average main account balance of last 30 days and 90 days. Outliers can be easily detected through some visual plots while analyzing numeric fields. Also, we can check whether or not the pcircle feature (which indicates the telecom circle) can help us or not.

Analytical modelling of the problem

Firstly, null values were checked and cross-verified as per the documentation. There were no null values. Thereafter, the Standard Deviation and mean of all the columns were checked. By looking at the mean of the label we could detect that label is highly imbalanced as there are only two unique values 0 and 1 and the mean of the label was 0.875 which means there are more values of 1 than 0.

	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	209593.000000	209593.000000	209593.000000
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

Data Sources and their formats

While a wide range of data was available in int/float64 data type there were few columns which were in object data type which needed to be referred to and encoded if necessary.

```
#Check Object type columns
df.select_dtypes('object').columns
```

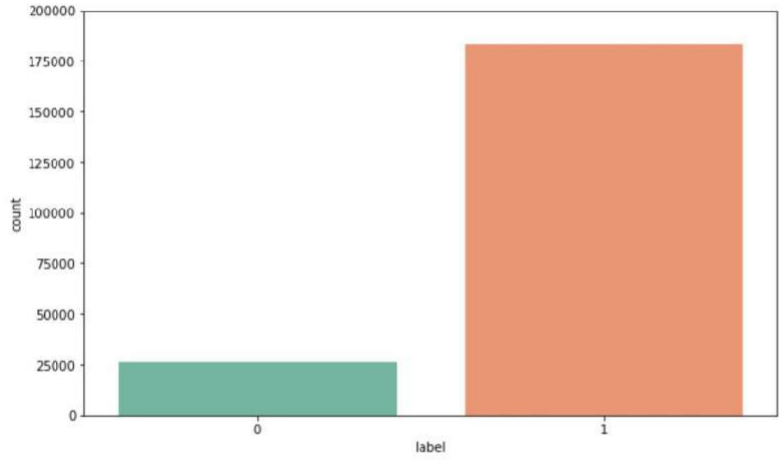
```
Index(['msisdn', 'pcircle', 'pdate'], dtype='object')
```

From the above object columns we directly dropped msisdn column as all the values in this field were unique and we also dropped pcircle column as there was only one unique value which would not have helped us in training the model.
While pdate column was spliced into pmonth by extracting the month from the date and thereafter pdate column was also dropped.

Data Preprocessing

Our first aim for data preprocessing would be to balance the label column first which was highly imbalanced. The below image depicts the exact value count of each value in the label column.

```
#Check count of Label
plt.figure(figsize=(10,6))
sns.countplot(df['label'],palette='Set2')
plt.ylim(0,200000)
plt.savefig('Images//1.Unsampled_Label.jpeg',dpi=300)
plt.show()
#As we can see the label is highly imbalanced
```




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

```
1    183430
0     26162
Name: label, dtype: int64
```

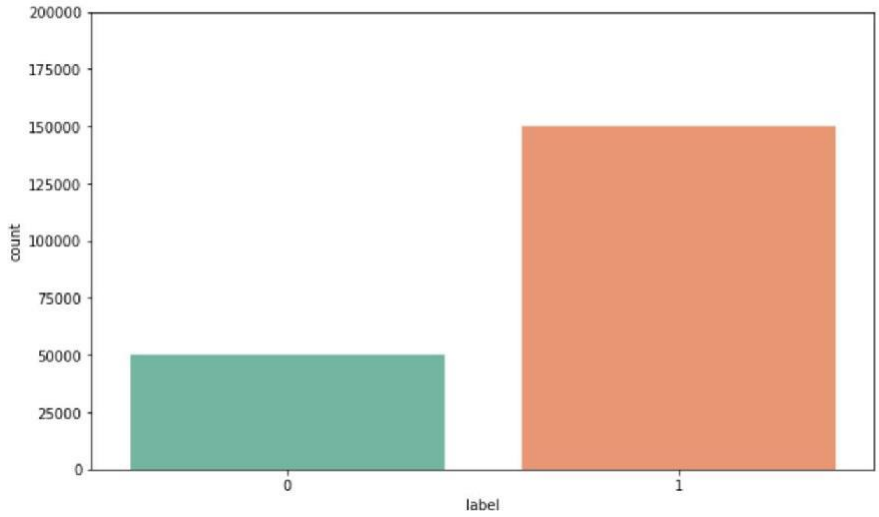
We would have to resample the data to balance the column but as the data is highly imbalanced and the minor value (0) is having only 26162 counts we could not downsample the major value (1) data because it could have led to major data loss and as the data is expensive we could not thinking of downsampling the major class.

Also, if we upsample the minor class it would have led to a lot of duplicate values so we have downsampled the major class first by 8-9% and then upsampled the minor class to 50000 counts. The below image shows the barplot of the label column after resampling the data.

```
#Check resmapled data
df_downsampled['label'].value_counts()
```

```
1    150000
0     50000
Name: label, dtype: int64
```

```
#Check count of Label
plt.figure(figsize=(10,6))
sns.countplot(df_downsampled['label'],palette='Set2')
plt.ylim(0,200000)
plt.savefig('Images//5.Sampled_Data.jpeg',dpi=300)
plt.show()
#As we can see now the data is enough balanced to train a model
```



Before resampling the data we also checked the outliers and skewness of the data expecting that there would be some outliers as there are a variety of different numeric columns with real data. For example, the cnt_da_rech30 column is for the Number of timings data account got recharged in the last 30 days, now this column can have a variety of values according to usage.
Performing a statistic computation will always find some values as anomalies but they might not be anomalies as they are real values that cannot be ignored. Also, considering the cost of data we cannot delete more than 8-9% of rows so we generated a report for detecting how much percent of outliers are present in each column. The below image shows the report.

```
#Run Function for each column
for x in df.columns:
    remove_outlier(df,str(x),False)
```

```
26162 outliers are detected for column label with percent change being 12.48234665445246
3607 outliers are detected for column aon with percent change being 1.720962632161533
16350 outliers are detected for column daily_decr30 with percent change being 7.80087026222375
18187 outliers are detected for column daily_decr90 with percent change being 8.677335012786749
18526 outliers are detected for column rental30 with percent change being 8.839077827397993
19399 outliers are detected for column rental90 with percent change being 9.25560135883049
20145 outliers are detected for column last_rech_date_ma with percent change being 9.611530974464674
6732 outliers are detected for column last_rech_date_da with percent change being 3.2119546547578155
20864 outliers are detected for column last_rech_amt_ma with percent change being 9.954578419023626
11294 outliers are detected for column cnt_ma_rech30 with percent change being 5.388564449024772
11450 outliers are detected for column fr_ma_rech30 with percent change being 5.462994770792778
13219 outliers are detected for column sumamnt_ma_rech30 with percent change being 6.307015534944081
24928 outliers are detected for column medianamnt_ma_rech30 with percent change being 11.89358372456964
27252 outliers are detected for column medianmarechprebal30 with percent change being 13.002404671934043
14155 outliers are detected for column cnt_ma_rech90 with percent change being 6.7535974655521205
26845 outliers are detected for column fr_ma_rech90 with percent change being 12.808217870911104
13954 outliers are detected for column sumamnt_ma_rech90 with percent change being 6.657696858658728
25457 outliers are detected for column medianamnt_ma_rech90 with percent change being 12.14597885415474
25933 outliers are detected for column medianmarechprebal90 with percent change being 12.373086759036605
4114 outliers are detected for column cnt_da_rech30 with percent change being 1.9628611779075538
1579 outliers are detected for column fr_da_rech30 with percent change being 0.7533684491774495
5367 outliers are detected for column cnt_da_rech90 with percent change being 2.560689339287759
865 outliers are detected for column fr_da_rech90 with percent change being 0.412706591854651
7817 outliers are detected for column cnt_loans30 with percent change being 3.7296270850032447
10416 outliers are detected for column amnt_loans30 with percent change being 4.96965533035612
30400 outliers are detected for column maxamnt_loans30 with percent change being 14.504370395816634
14148 outliers are detected for column medianamnt_loans30 with percent change being 6.750257643421505
11523 outliers are detected for column cnt_loans90 with percent change being 5.497824344440627
12590 outliers are detected for column amnt_loans90 with percent change being 6.006908660635902
28648 outliers are detected for column maxamnt_loans90 with percent change being 13.668460628268257
12169 outliers are detected for column medianamnt_loans90 with percent change being 5.806042215351731
16531 outliers are detected for column payback30 with percent change being 7.887228520172526
17849 outliers are detected for column payback90 with percent change being 8.516069315622733
0 outliers are detected for column pmonth with percent change being 0.0
```

However, we did not delete any columns based on outliers as the above report shows that almost all the columns were having more than 6-8% of outliers.

Thereafter, we checked skewness and applied PowerTransformer with the yeo- Jonson method.

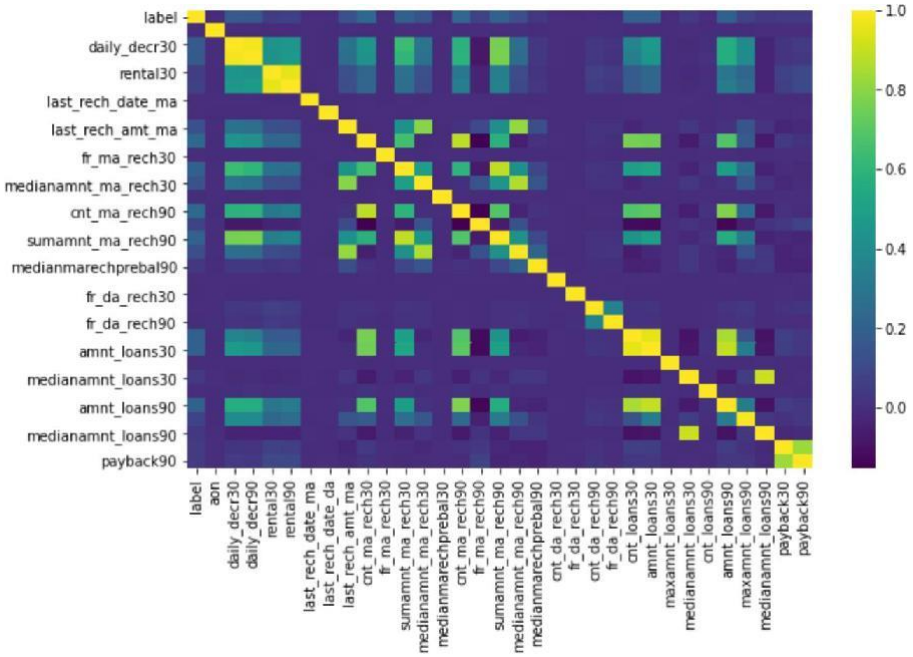
```
df.skew()
```

```
label                   -2.270245
aon                      1.654515
daily_decr30            -6.582798
daily_decr90            -7.080271
rental30                -1.022864
rental90                -0.962734
last_rech_date_ma       -5.362508
last_rech_date_da     -106.606388
last_rech_amt_ma        -0.061265
cnt_ma_rech30           -0.000144
fr_ma_rech30             0.165014
sumamnt_ma_rech30       -0.291322
medianamnt_ma_rech30    -0.189630
medianmarechprebal30    -0.119068
cnt_ma_rech90           -0.002465
fr_ma_rech90             0.142950
sumamnt_ma_rech90       -0.192160
medianamnt_ma_rech90    -0.044606
medianmarechprebal90     7.506249
cnt_da_rech30            6.925803
fr_da_rech30            11.390638
cnt_da_rech90            6.006561
fr_da_rech90            15.469663
cnt_loans30              0.036243
amnt_loans30             0.001947
maxamnt_loans30         -1.680102
medianamnt_loans30       3.447725
cnt_loans90              0.105583
amnt_loans90            -0.008989
maxamnt_loans90          0.364281
medianamnt_loans90       3.779587
payback30                0.298562
payback90                0.210885
pmonth                   0.343245
dtype: float64
```

```
pt=PowerTransformer()
for x in df.columns.drop('label'):
    if abs(df.loc[:,x].skew())>0.55:
        df.loc[:,x]=pt.fit_transform(df.loc[:,x].values.reshape(-1,1))
```

Considering the number of features we also needed to perform feature selection to discard multicollinearity and select the correct features which can help train the model.

The below Image shows the correlation heatmap.

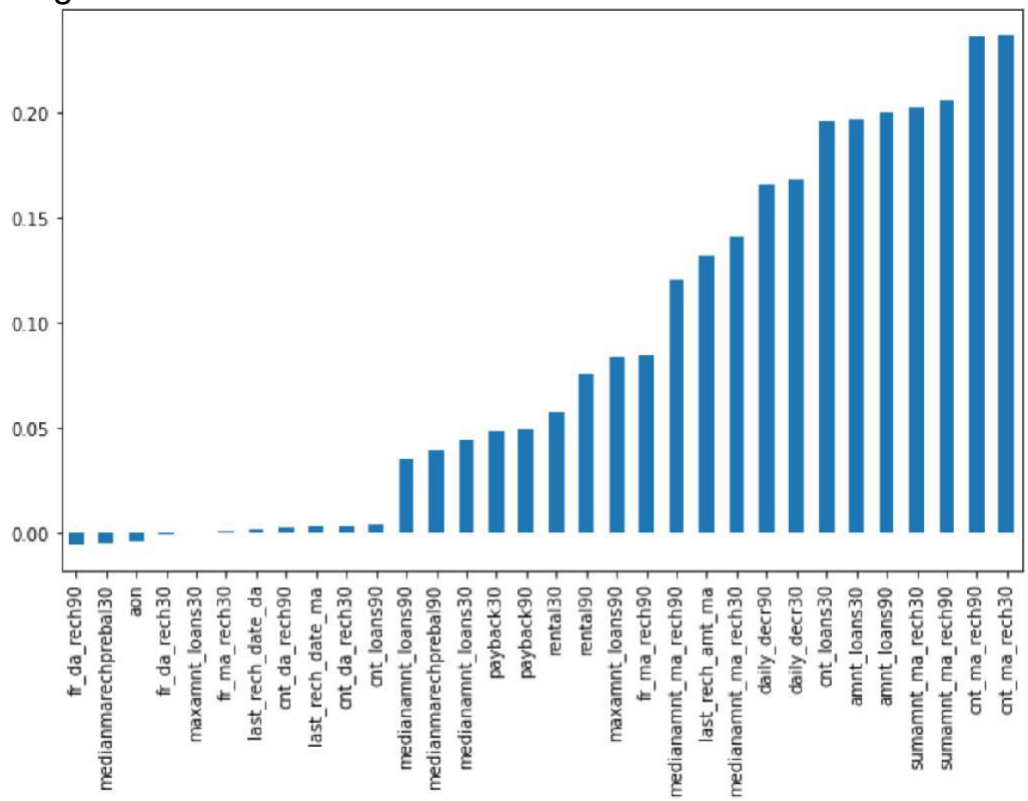


With the help of the Variance Inflation Factor, we discarded 6 columns to discard multicollinearity.

The last step that was performed for preprocessing this data was scaling the data and hence the data was scaled with MinMaxScaler and was sliced into X (features) and y (label).

Data Input-Output Relationship

To view the impact of each column on the label we plotted the below image which shows that the cnt_ma_rech30 was having a strong correlation with the label.



In the above process of preprocessing the data, we have assumed that the data is definite, and as it might have a wide range of numeric values we have not discarded the detected outliers.

Hardware/Software Requirements

- Hardware Requirements:
  - Processor: 7th gen Core i5 or above
  - Ram: 4GB DDR3 or above
  - HDD/SDD: 128GB or above
  - GPU: Intel Iris Plus Graphics 640 1536 MB or above

- Software Requirements:
  - Anaconda Software must be installed with all necessary libraries like pandas, NumPy, matplotlib, seaborn, scipy, sklearn, statmodels, and xgboost.

Model Development and Evaluation

We already detected from prior statistic checks that the target column is highly imbalanced which will affect the performance of the model by underfitting the data. To resolve this we resampled the data and made it machine learnable.
Other necessary checks were made for outliers, skewness, and scaling the dataset.

After preprocessing the data we made a function which can be called by the passing model as a parameter to check the performance of each model.
We ran different models like LogisticRegression, RandomForestClassifier, AdaBoostClassifier, SVC, KNeighborsClassifier, and XGBoost wherein we found that XGBoost and RandomForestClassifier outperformed other models.

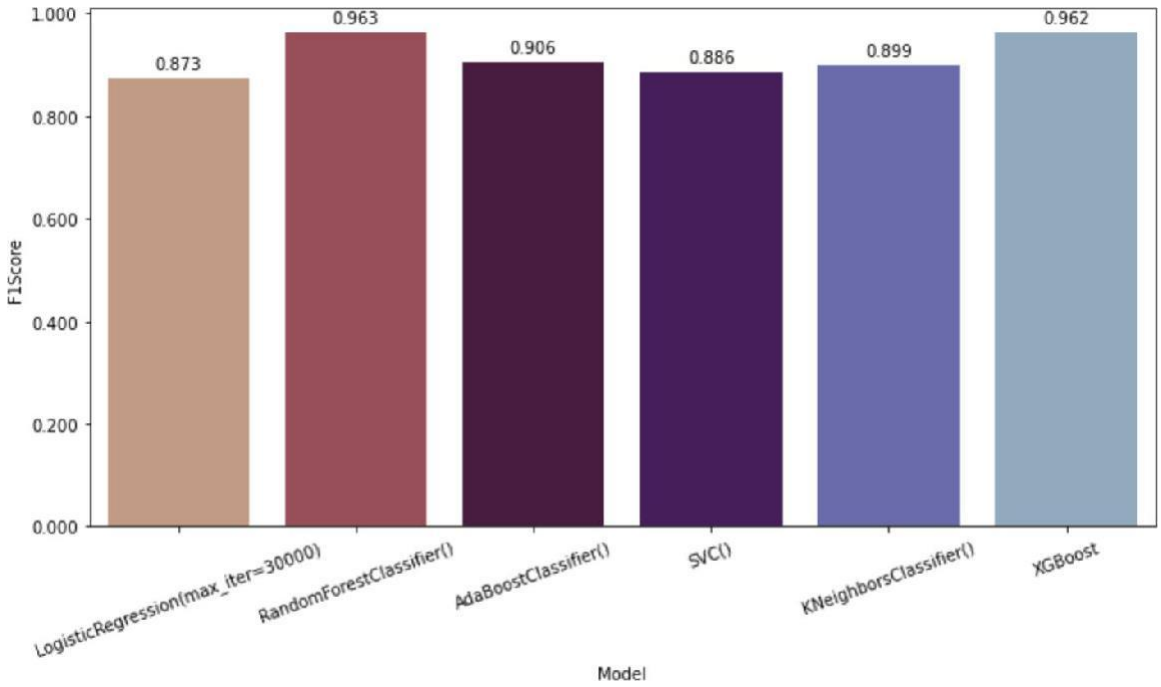
To evaluate the models we made a report with different metrics but to compare them side-by-side we chose the F1 score as the F1 score is the harmonic mean of Precision and Recall which will give high effectiveness to evaluate the model trained with an imbalanced dataset.

The below image shows the report which gets generated for each of the models:

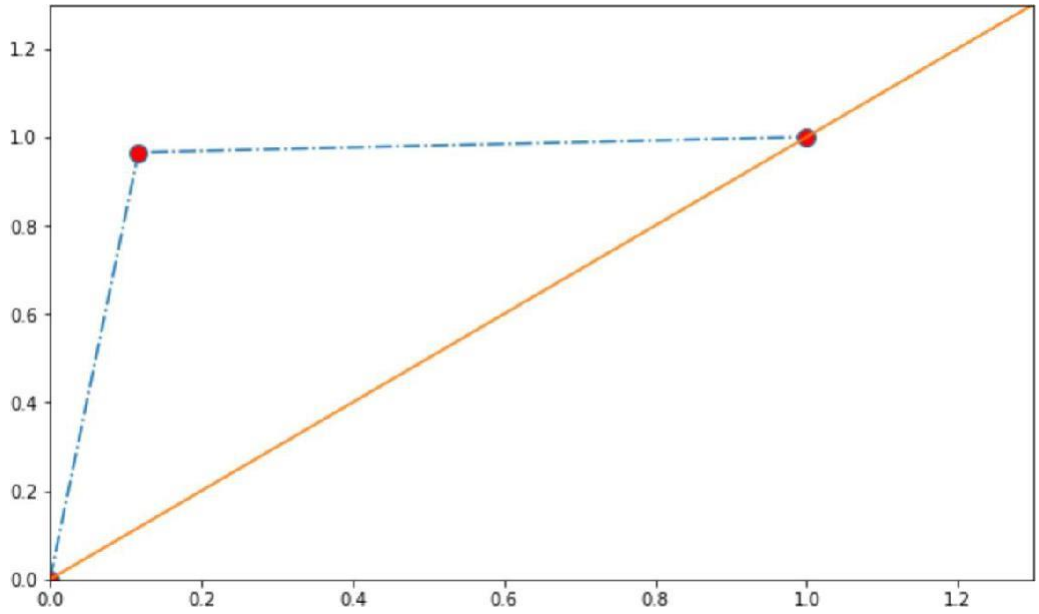
```
#Run Model for RandomForestClassifier
model_sel(RandomForestClassifier())
```

```
****************** Metrics **********************

Accuracy of the model is 0.9448166666666666
Recall of the model is 0.9653285889024797
Precision of the model is 0.9612028876389566
F1 score of the model is 0.9632613206395704

************** Confusion Matrix ****************

[[13283  1752]
 [ 1559 43406]]

************ Classification Report **************

              precision    recall  f1-score   support

           0       0.89      0.88      0.89     15035
           1       0.96      0.97      0.96     44965

    accuracy                           0.94     60000
   macro avg       0.93      0.92      0.93     60000
weighted avg       0.94      0.94      0.94     60000
```

To visualize comparison of the matrix we have plotted below bar plot which shows F1 score on Y-Axis and Model name on X-Axis:



After creating a RandomForestClassifier model we also depicted the AUC-ROC curve with a plot:



We hyper tuned the parameters of Random Forest to choose the best n_estimators value.

Conclusions

From the above, we can conclude that for this dataset Random Forest works well and can be moved to production as it has given around 95% accuracy, Precision, and Recall.
Random Forest is managing the outliers very well and thus we don't have to treat outliers explicitly and thereby we are saving the data-keeping it for training rather than discarding the outliers.

- Scope of Future Work:

With Random Forest making trees of different nodes we can also solve inference problems rather than only predicting the target label. With the help of Random Forest feature importance we can let the client know that which feature is highly important to attenuate the Micro Credit defaulters.