MICRO CREDIT DEFAULTER

Submitted by:

Lahari S.K

ACKNOWLEDGMENT

I would like to thank everyone who helped through this project directly and indirectly.

INTRODUCTION

A MicroFinance Institution is an organization that offers financial services to low income populations. The MFI mainly targets the low income families which have one or two sources of income due to which they will not be eligible to take loans from banks.

Microfinance is widely accepted as poverty reduction tool representing $170 billion in outstanding loans and global outreach of 200 million clients.

Microfinance Institutions serves several clients one such client is telecom Industry. They are fixed wireless telecommunication network provider.

They have launched various products and have developed its business and organization offering better products at low prices to all the conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

They understand the importance of communication and how it affect person's life, thus focusing on providing their services and products to low income families and poor customers that can help them in need of hour.

They are collaborating with MFI to provide micro-credit on mobile balances to be paid back in 5 days. The consumer is assumed defaulter if he fails to pay within the deadline and non-defaulter if he succeeds paying within the time.

For a loan amount of 5 payback amount should be 6rs(Indonesian Rupiah),while for amount of 12 payback amount should be 12.

This is exercised in order to improve the selection of customers for the credit and also some predictions that could help him in the further investment and improvement in the selection of cust

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

The data set is a comma separated value file.The dataset contains 37 columns and 2,0,9593 rows. It contains both numerical and object data types.13 features are of int64 dtype,3 are of object dtype and 26 features are of float64 dtypes.

When checked for the null values, no null values were found. Following snapshot is the one which describes the unique values found each feature.

```
Unnamed: 0              int64
label                   int64
msisdn                 object
aon                   float64
daily_decr30          float64
daily_decr90          float64
rental30              float64
rental90              float64
last_rech_date_ma     float64
last_rech_date_da     float64
last_rech_amt_ma        int64
cnt_ma_rech30           int64
fr_ma_rech30          float64
sumamnt_ma_rech30     float64
medianamnt_ma_rech30  float64
medianmarechprebal30  float64
cnt_ma_rech90           int64
fr_ma_rech90            int64
sumamnt_ma_rech90       int64
medianamnt_ma_rech90  float64
medianmarechprebal90  float64
cnt_da_rech30         float64
fr_da_rech30          float64
cnt_da_rech90           int64
fr_da_rech90            int64
cnt_loans30             int64
amnt_loans30            int64
maxamnt_loans30       float64
medianamnt_loans30    float64
cnt_loans90           float64
amnt_loans90            int64
maxamnt_loans90         int64
medianamnt_loans90    float64
payback30             float64
payback90             float64
pcircle                object
pdate                  object
dtype: object
```

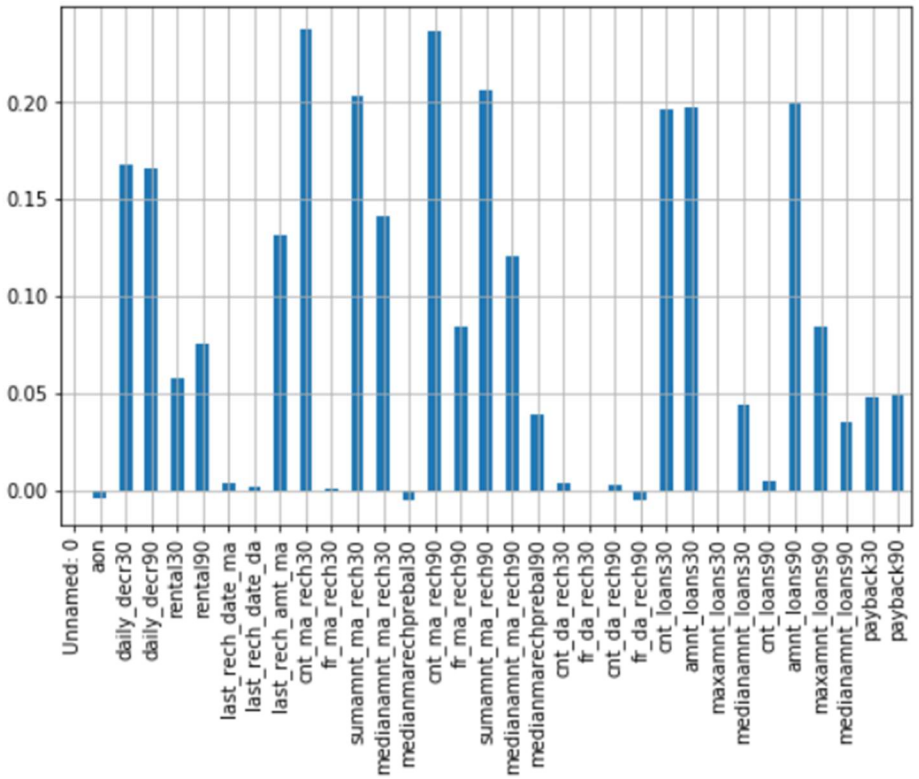
Statistical Summary

	Unnamed: 0	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000
mean	104797.000000	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921
std	60504.431823	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430
min	1.000000	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000
25%	52399.000000	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000
50%	104797.000000	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000
75%	157195.000000	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000
max	209593.000000	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410

last_rech_amt_ma	...	cnt_loans30	amnt_loans30	maxamnt_loans30	medianamnt_loans30	cnt_loans90	amnt_loans90	maxamnt_loans90	m
209593.000000	...	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	
2064.452797	...	2.758981	17.952021	274.658747	0.054029	18.520919	23.645398	6.703134	
2370.786034	...	2.554502	17.379741	4245.264648	0.218039	224.797423	26.469861	2.103864	
0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
770.000000	...	1.000000	6.000000	6.000000	0.000000	1.000000	6.000000	6.000000	
1539.000000	...	2.000000	12.000000	6.000000	0.000000	2.000000	12.000000	6.000000	
2309.000000	...	4.000000	24.000000	6.000000	0.000000	5.000000	30.000000	6.000000	
55000.000000	...	50.000000	306.000000	99864.560864	3.000000	4997.517944	438.000000	12.000000	

medianamnt_loans90	payback30	payback90
209593.000000	209593.000000	209593.000000
0.046077	3.398826	4.321485
0.200692	8.813729	10.308108
0.000000	0.000000	0.000000
0.000000	0.000000	0.000000
0.000000	0.000000	1.666667
0.000000	3.750000	4.500000
3.000000	171.500000	171.500000

Correlation



The above snapshot is the correlation of the feature variables along with the label.

Below are the features which are related to each other

DAILY_DECR30

1.daily_decr30:daily_decr90=0.98

2.daily_decr30:sumamnt_ma_rech90=0.64

3.daily_decr30:cnt_ma_rech90=0.58

4.daily_decr30:sumamnt_ma_rech90=0.76

5.daily_decr30:amnt_loans90=0.56

DAILY_DECR90

1.daily_decr90:cnt_ma_rech90=0.59

2.daily_decr90:sumamnt_ma_rech90=0.76

3.daily_decr90:sumamnt_ma_rech30=0.6

RENTAL30

1.rental30:rental90=0.95

CNT_MA_RECH30

1.cnt_ma_rech30:amnt_loans90=0.69

2.cnt_ma_rech30:amnt_loans30=0.75

3.cnt_ma_rech30:cnt_loans30=0.77

4.cnt_ma_rech30:cnt_ma_rech90=0.89

5.cnt_ma_rech30:sumamnt_ma_rech30=0.66

SUMAMNT_MA_RECH30

1.sumamnt_ma_rech30:sumamnt_ma_rech90=0.88

MEDIANAMNT_MA_RECH30

1.meadianamnt_ma_rech30:median_ma_rech90=0.86

2.medianamnt_ma_rech30:last_rech_amnt_ma=0.79

CNT_MA_RECH90

1.cnt_ma_rech90:amnt_loans90=0.78

2.cnt_ma_rech90:amnt_loan30=0.7

3.cnt_ma_rech90:cnt_loan30=0.69

4.cnt_ma_rech90:sumamnt_ma_rech90=0.69

5.cnt_ma_rech90:cnt_ma_rech30=0.89

SUMAMNT_MA_RECH90

1.sumamnt_ma_rech90:amnt_loans90=0.56

2.sumamnt_ma_rech90:sumamnt_ma_rech30=0.89

MEDIANAMNT_MA_RECH90

1.medianamnt_ma_rech90:medianamnt_ma_rech30=0.86

2.medianamnt_ma_rech90:last_rech_amnt_ma=0.52

CNT_LOANS30

1.cnt_loans30:amnt_loans90=0.85

2.cnt_loans30:amnt_loans=0.95

AMNT_LOANS30

1.amnt_loans30:amnt_loans90=0.85

MEDIANAMNT_LOANS30

1.medianamnt_loans30:medianamntloans90=0.91

PAYBACK30

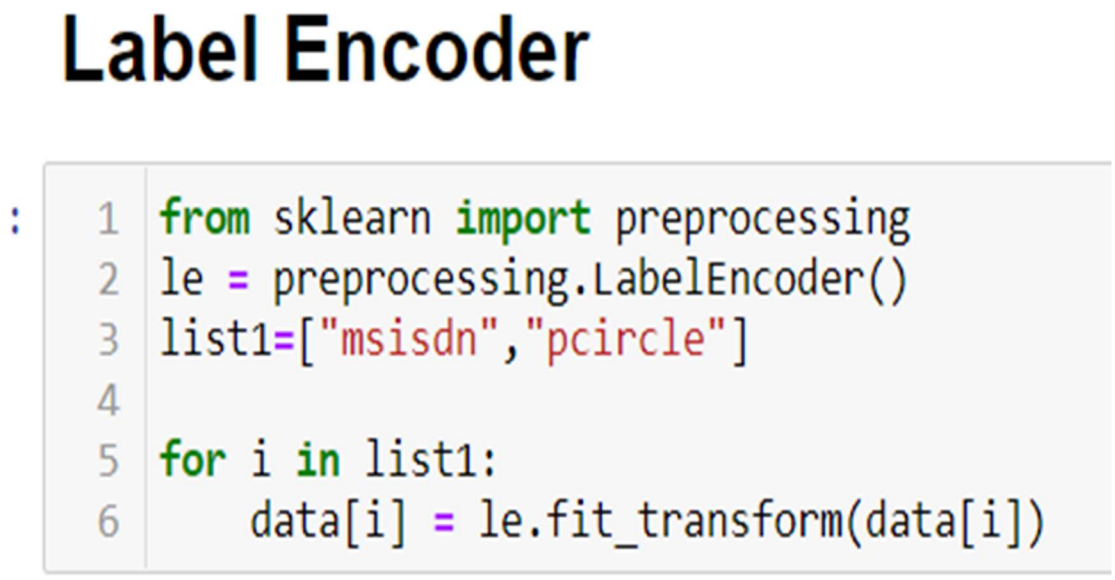
1.payback30:payback90=0.83

Data Pre-processing Done

In order to get the best results, reprocessing plays a vital role. The pre-processing steps followed are as follows

1.Label Encoding:

Label Encoding is used to convert the categorical features into the numeric ones. One of the alternative for the label encoding is get_dummies. The decision for choosing label encoder in place of get_dummies is that when get_dummies is implemented there will be increase in the dimensions. The size is already bit large. It might consume much larger space. Due to space constraint Label encoder was apt.



2.VIF:

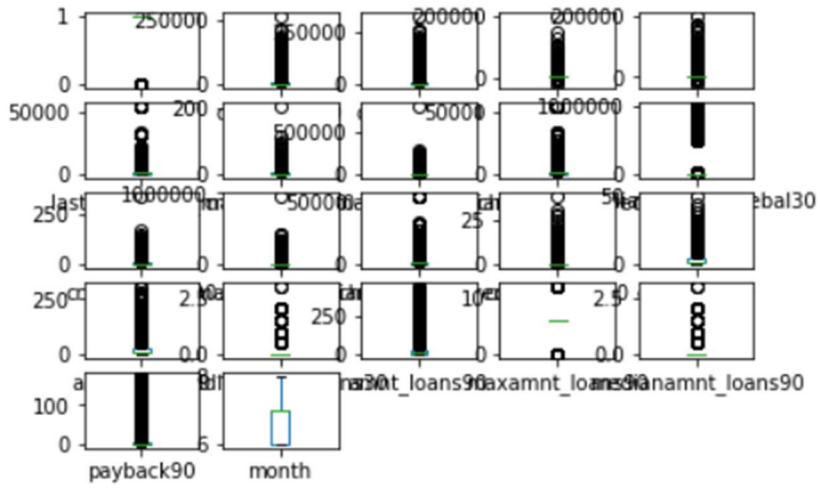
Variance Inflation Factor is used to determine the correlation between the independent variables. The columns having greater vif values of more than 5 are kept and remaining are dropped keeping in mind they have high collinearity and the strength. After evaluating VIF 17 columns have been dropped due to low collinearity.

	variables	VIF
0	Unnamed: 0	3.936579
1	label	8.951075
2	msisdn	3.932397
3	aon	1.011568
4	daily_decr30	39.219822
5	daily_decr90	42.395803
6	rental30	18.355275
7	rental90	18.943695
8	last_rech_date_ma	1.004979
9	last_rech_date_da	1.004934
10	last_rech_amt_ma	6.040311
11	cnt_ma_rech30	28.259965
12	fr_ma_rech30	1.004964
13	sumamnt_ma_rech30	20.065622
14	medianamnt_ma_rech30	8.993545
15	medianmarechprebal30	1.005214
16	cnt_ma_rech90	28.965063
17	fr_ma_rech90	1.485738
18	sumamnt_ma_rech90	23.404345
19	medianamnt_ma_rech90	10.152467
20	medianmarechprebal90	1.132056
21	cnt_da_rech30	1.004151
22	fr_da_rech30	1.004971
23	cnt_da_rech90	1.161609
24	fr_da_rech90	1.143080
25	cnt_loans30	49.660547
26	amnt_loans30	60.649188
27	maxamnt_loans30	1.004321
28	medianamnt_loans30	6.386202
29	cnt_loans90	1.007236
30	amnt_loans90	19.451515
31	maxamnt_loans90	21.266861
32	medianamnt_loans90	6.330202
33	payback30	3.755771
34	payback90	3.899129
35	pcircle	NaN
36	month	31.345161
37	day	3.800255

3.Outliers:

The presence of outliers is big disadvantage in obtaining good results because some algorithms are very muchsensitive to the outliers.Modelling along with the outliers present would resuly in very bad results.hence outliers had to be removed.

Inorder to remove outliers z scores was implemented .Zscores of data points having more than the threshold 3 is considered as outliers and have been removed.The shape of the data set after removal of outliers is (174154, 21).



Hardware and Software Requirements and Tools Used

Hardware Requirements

Software Requirements

1.**JUPYTER NOTEBOOK**: The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning.

Libraries used in jupyter notebook

1.pandas: Here pandas is used to read the csv file.

2.numpy: It is used to compute the zscores.

3.matplotlib.pyplot: Creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels.

4. seaborn: Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Ex: heatmaps,pie charts,scatterplot,pairplot,countplot,violinplot and distplot

5. sklearn: Provides supervised algorithms and pre-processing.

a.**sklearn.preprocessing import LabelEncoder**:

Label encoder is used to encode categorical values into numerical values.

b. sklearn.preprocessing import StandardScaler:

Standard scaler is used to scale all values in the same range so that the the graph becomes normalized, mean somewhere lies nearly 0.

6.scipy: It is used to solve scientific and mathematical calculations.

a.scipy.stats import zscore: Compute the z score of each value in the sample, relative to the sample mean and standard deviation.

7. from statsmodels.stats.outliers_influence import variance_inflation_factor : Used to calculate the vif values as a part of feature engineering.

Testing of Identified Approaches

The algorithms used for the training and testing data are as follows

1.Logistic Regression:

Imported from sklearn. linear_model import LogisticRegression

```
Accuracy of  LogisticRegression() is :
0.8650884794070053
[[  294  5007]
 [  162 32851]]
              precision    recall  f1-score   support

           0       0.64      0.06      0.10      5301
           1       0.87      1.00      0.93     33013

    accuracy                           0.87     38314
   macro avg       0.76      0.53      0.51     38314
weighted avg       0.84      0.87      0.81     38314
```

We can use logistic regression only when the label is of binary. It is achieving good accuracy score but recall and f1 scores are to low so we can't rely on this model

1.Guassian NB:

from sklearn.naive_bayes import GaussianNB

```
Accuracy of  GaussianNB() is :
0.6224878634441718
[[ 4553   748]
 [13716 19297]]
              precision    recall  f1-score   support

           0       0.25      0.86      0.39      5301
           1       0.96      0.58      0.73     33013

    accuracy                           0.62     38314
   macro avg       0.61      0.72      0.56     38314
weighted avg       0.86      0.62      0.68     38314
```

Naïve Bayes is a probabilistic machine learning algorithm based on the Bayes Theorem. Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.GuassianNB is used when there are only 2 classes in target variable.

3.KNeighbour Classifier

from sklearn.neighbors import KNeighborsClassifier

```
Accuracy of  KNeighborsClassifier() is :
0.8881348854204729
[[ 2531  2770]
 [ 1516 31497]]
              precision    recall  f1-score   support

           0       0.63      0.48      0.54      5301
           1       0.92      0.95      0.94     33013

    accuracy                           0.89     38314
   macro avg       0.77      0.72      0.74     38314
weighted avg       0.88      0.89      0.88     38314
```

KNN works on the concept that the similar things exist in close proximity.

4.SVC

from sklearn.svm import SVC

```
Accuracy of  SVC() is :
0.8723443127838388
[[  854  4447]
 [  444 32569]]
              precision    recall  f1-score   support

           0       0.66      0.16      0.26      5301
           1       0.88      0.99      0.93     33013

    accuracy                           0.87     38314
   macro avg       0.77      0.57      0.59     38314
weighted avg       0.85      0.87      0.84     38314
```

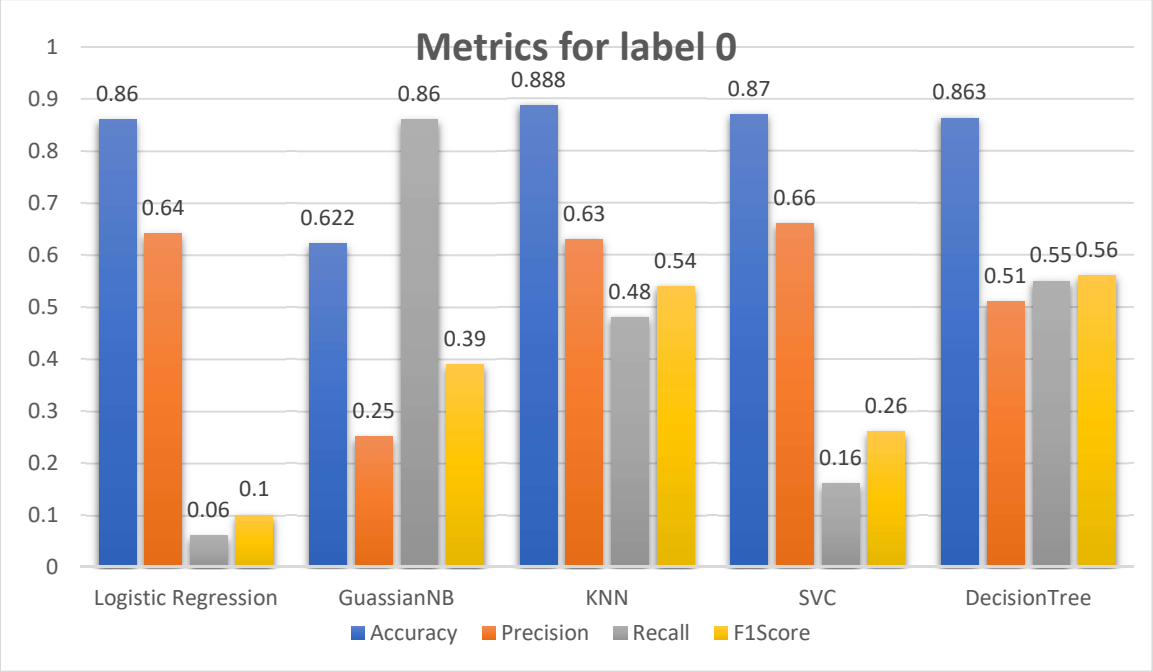
SVM s main objective is to fit to the data we provide, returning a "best fit" hyperplane that divides, or categorizes, your data.

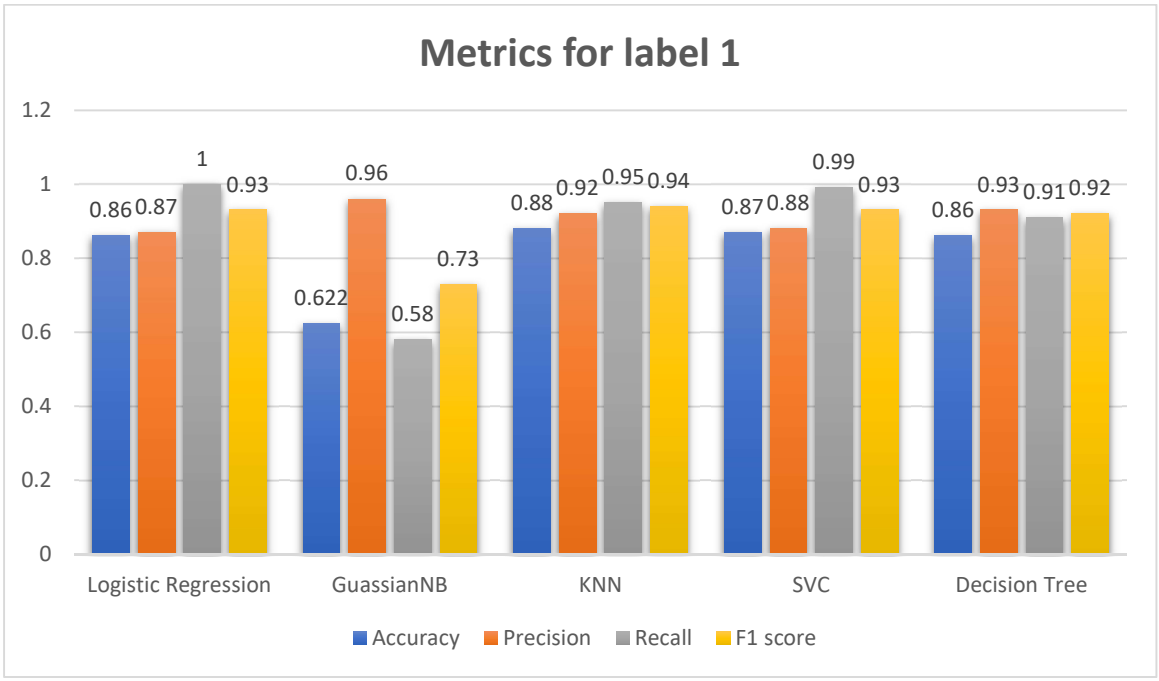
5.DecisionTree Classifier

from sklearn.tree import DecisionTreeClassifier

```
Accuracy of  DecisionTreeClassifier() is :
0.8633919716030694
[[ 2908  2393]
 [ 2841 30172]]
              precision    recall  f1-score   support

           0       0.51      0.55      0.53      5301
           1       0.93      0.91      0.92     33013

    accuracy                           0.86     38314
   macro avg       0.72      0.73      0.72     38314
weighted avg       0.87      0.86      0.87     38314
```

Decision tree builds **classification** in the form of a **tree** structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated **decision tree** is incrementally developed. The final result is a **tree** with **decision** nodes and leaf nodes.


Metrics for label 0
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.86
0.64
0.06
0.1
0.622
0.25
0.86
0.39
0.888
0.63
0.48
0.54
0.87
0.66
0.16
0.26
0.863
0.51
0.55
0.56
Logistic Regression
GuassianNB
KNN
SVC
DecisionTree
Accuracy
Precision
Recall
F1Score


Metrics for label 1
1.2
1
0.8
0.6
0.4
0.2
0
0.86
0.87
1
0.93
0.622
0.96
0.58
0.73
0.88
0.92
0.95
0.94
0.87
0.88
0.99
0.93
0.86
0.93
0.91
0.92
Logistic Regression
GuassianNB
KNN
SVC
Decision Tree
Accuracy
Precision
Recall
F1 score

We can observe that the accuracies are fine but other metrics are not so good so let's implement the ensemble techniques.

ENSEMBLE TECHNIQUES

1.RandomForestClassifier

from sklearn.ensemble import RandomForestClassifier

```
Accuracy of  RandomForestClassifier() is :
0.9049172626194081
[[ 2677  2624]
 [ 1019 31994]]
              precision    recall  f1-score   support

           0       0.72      0.50      0.60      5301
           1       0.92      0.97      0.95     33013

    accuracy                           0.90     38314
   macro avg       0.82      0.74      0.77     38314
weighted avg       0.90      0.90      0.90     38314
```

The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree

2.AdaBoost Classifier

from sklearn.ensemble import AdaBoostClassifier

```
Accuracy of  AdaBoostClassifier() is :
0.9008195437699014
[[ 2114  3187]
 [  613 32400]]
              precision    recall  f1-score   support

           0       0.78      0.40      0.53      5301
           1       0.91      0.98      0.94     33013

    accuracy                           0.90     38314
   macro avg       0.84      0.69      0.74     38314
weighted avg       0.89      0.90      0.89     38314
```

An AdaBoost [1] classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

3.GradientBoosting Classifier

from sklearn.ensemble import GradientBoostingClassifier

```
Accuracy of  GradientBoostingClassifier() is :
0.9069791721041917
[[ 2447  2854]
 [  710 32303]]
              precision    recall  f1-score   support

           0       0.78      0.46      0.58      5301
           1       0.92      0.98      0.95     33013

    accuracy                           0.91     38314
   macro avg       0.85      0.72      0.76     38314
weighted avg       0.90      0.91      0.90     38314
```

It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error.If a small change in the prediction for a case causes no change in error, then next target outcome of the case is zero.

4.Bagging Classifier

from sklearn.ensemble import BaggingClassifier

```
Accuracy of  BaggingClassifier() is :
0.8977919298428773
[[ 2947  2354]
 [ 1562 31451]]
              precision    recall  f1-score   support

           0       0.65      0.56      0.60      5301
           1       0.93      0.95      0.94     33013

    accuracy                           0.90     38314
   macro avg       0.79      0.75      0.77     38314
weighted avg       0.89      0.90      0.89     38314
```

Bagging constructs n classification trees using bootstrap sampling of the training data and then combines their predictions to produce a final meta-prediction.

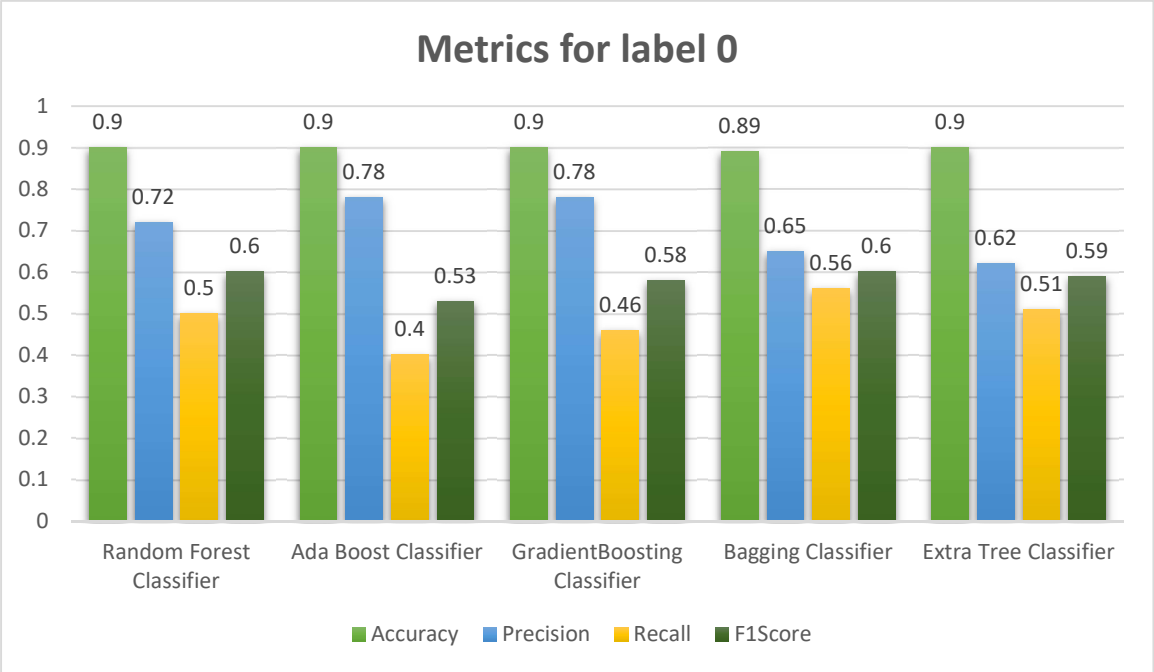
5.Extra Tree Classifier

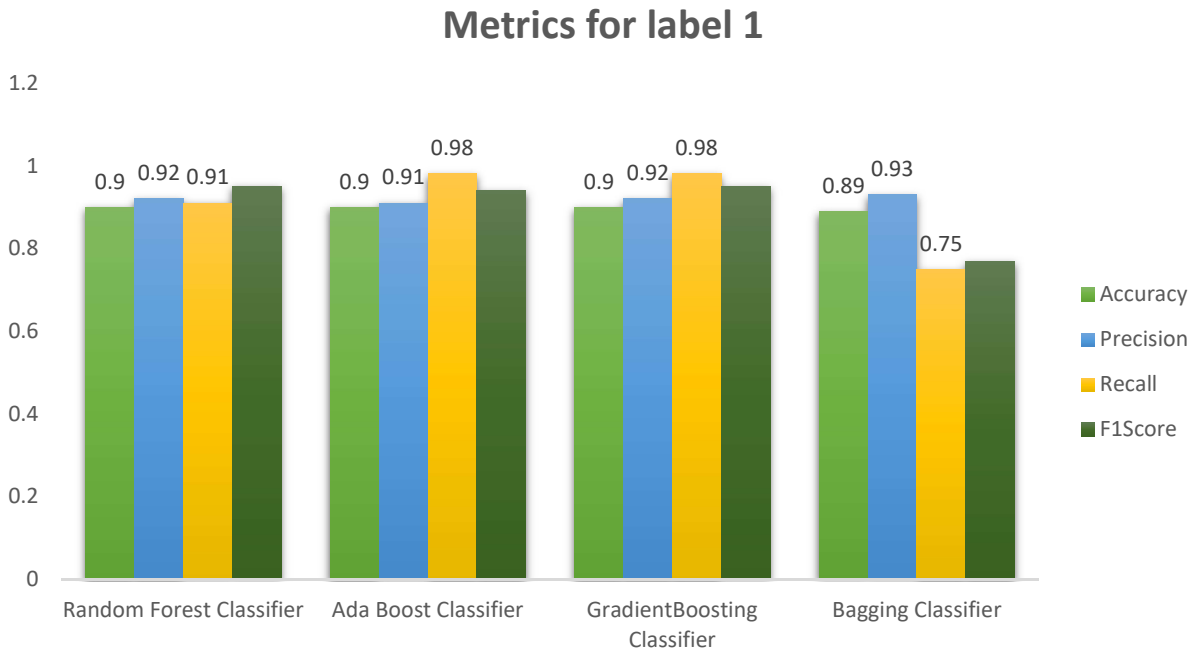
from sklearn.ensemble import ExtraTreesClassifier

```
Accuracy of  ExtraTreesClassifier() is :
0.9008717440100225
[[ 2723  2578]
 [ 1220 31793]]
              precision    recall  f1-score   support

           0       0.69      0.51      0.59      5301
           1       0.92      0.96      0.94     33013

    accuracy                           0.90     38314
   macro avg       0.81      0.74      0.77     38314
weighted avg       0.89      0.90      0.89     38314
```

The Extra Trees algorithm works by creating a large number of unpruned decision trees from the training dataset.

Predictions are made by averaging the prediction using majority voting in the case of classification.





Among all the models that we have seen let's select the bagging classifier as the best model and process with it.

Key Metrics for success in solving problem under consideration

Bagging Classifier

The code for bagging classifier and the evaluation metrics are as follows

Bagging Classifier

```
:  1 bgc=BaggingClassifier()
   2 bgc.fit(x_train,y_train)
   3 bgc.score(x_train,y_train)
   4 ypred=g.predict(x_test)
   5 print("Accuracy of ",bgc,"is :")
   6 print(accuracy_score(y_test,ypred))
   7 print(confusion_matrix(y_test,ypred))
   8 print(classification_report(y_test,ypred))
   9 print("_____")
  10 print('\n')
```

```
Accuracy of  BaggingClassifier() is :
0.8977919298428773
[[ 2947  2354]
 [ 1562 31451]]
              precision    recall  f1-score   support

           0       0.65      0.56      0.60      5301
           1       0.93      0.95      0.94     33013

    accuracy                           0.90     38314
   macro avg       0.79      0.75      0.77     38314
weighted avg       0.89      0.90      0.89     38314
```

1.**Accuracy**: When we have imbalanced data we cannot really rely on just the accuracy scores. Accuracy should be used when the data set is nearly balanced ,definitely not in our case.

2.Precision: Precisions tells us what proportion of customers we identified as defaulters/non-defaulters are really defaulters/non-defaulters respectively.

So out of 100 non-defaulters our model can identify only 63 as non-defaulters.

Precision is about being precise. So even if we managed to capture only one defaulter, and we captured it correctly, then we are 100% precise.

3.Recall/Sensitivity: Recall is not so much about capturing cases correctly but more about capturing all non-defaulter with the answer as “non-defaulter”. So if we simply always say every non-defaulter as “non-defaulter”, we have 100% recall.

4.F1 Score: The F-score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the model's precision and recall**.**

So here in our case we should try to increase our true positive and true negative and reduce false positive and false negative.

I have carried out two methods to increase TP and TN that are

1.Oversampling

2.ADAYSN

1.Over Sampling

Over sampling is a technique of increasing the data points frequently on the same data so that the minority class data points are increased.

Before the over sampling the count of minority class was 18795 and after over sampling the count of minority class is 93636.Its increased 5% times.

2.ADASYN(Adaptive Synthetic Sampling Method)

ADASYN (Adaptive Synthetic) is an algorithm that generates synthetic data, and its greatest advantages are not copying the same minority data, and generating more data for “harder to learn” examples.

Before the over sampling the count of minority class was 18795 and after over sampling the count of minority class is 93636.Its increased 5% times.

RandomOverSampler

In [63]:
```
from collections import Counter
from imblearn.over_sampling import RandomOverSampler
```

In [64]:
```
os=RandomOverSampler(0.8)
x_train_os,y_train_os=os.fit_sample(x_train,y_train)
print(Counter(y_train))
print(Counter(y_train_os))
```

```
Counter({1: 117045, 0: 18795})
Counter({1: 117045, 0: 93636})
```

In [65]:
```
bgc=BaggingClassifier()
bgc.fit(x_train_os,y_train_os)
ypred=bgc.predict(x_test)
print(confusion_matrix(y_test,ypred))
print(classification_report(y_test,ypred))
print(accuracy_score(y_test,ypred))
```

```
[[ 3234  2067]
 [ 2286 30727]]
              precision    recall  f1-score   support

           0       0.59      0.61      0.60      5301
           1       0.94      0.93      0.93     33013

    accuracy                           0.89     38314
   macro avg       0.76      0.77      0.77     38314
weighted avg       0.89      0.89      0.89     38314

0.8863861773764159
```

ADASYN

|:
```
from imblearn.over_sampling import ADASYN
```

|:
```
ada = ADASYN(random_state=45)
```

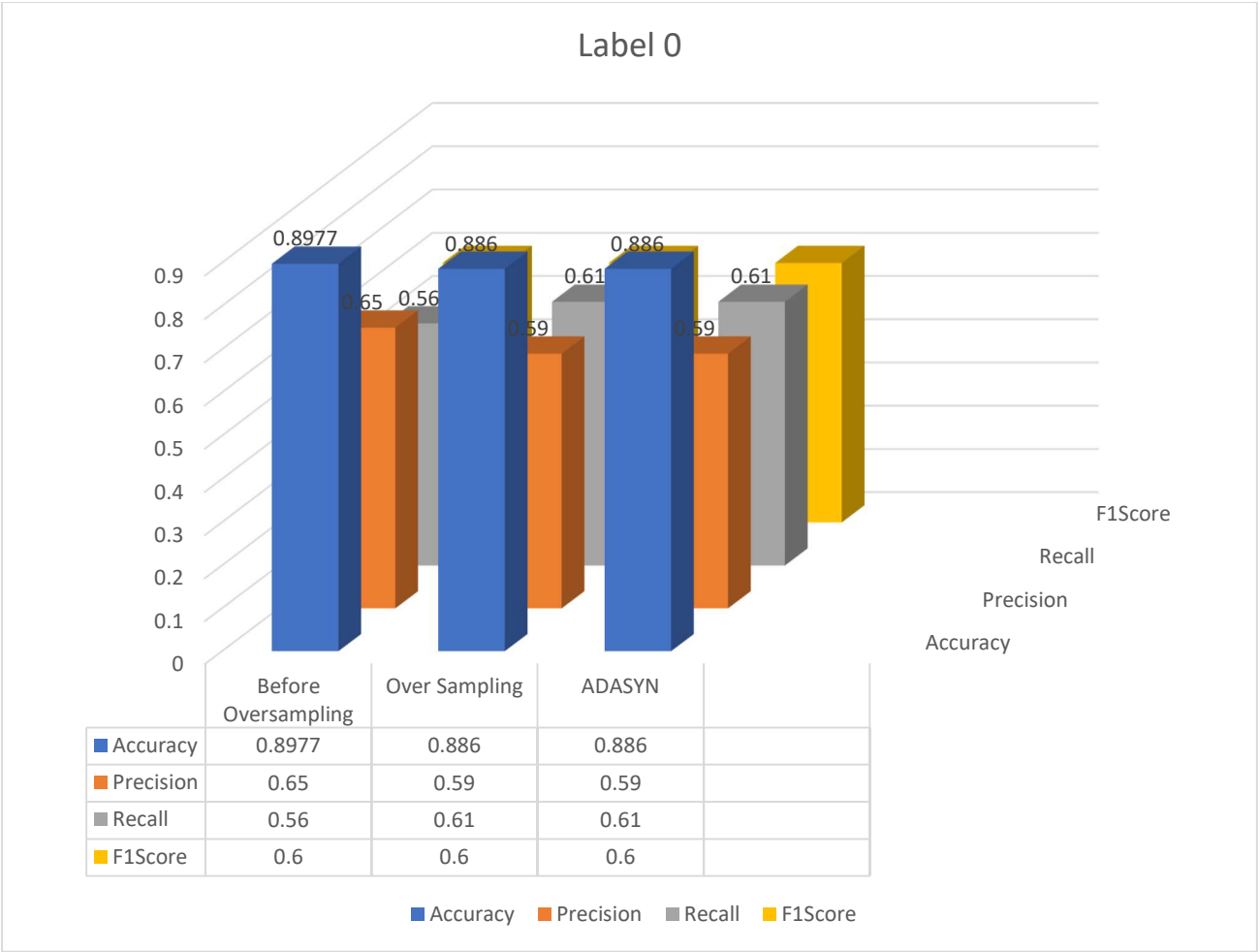
|:
```
x_res, y_res = ada.fit_resample(x_train,y_train)
```

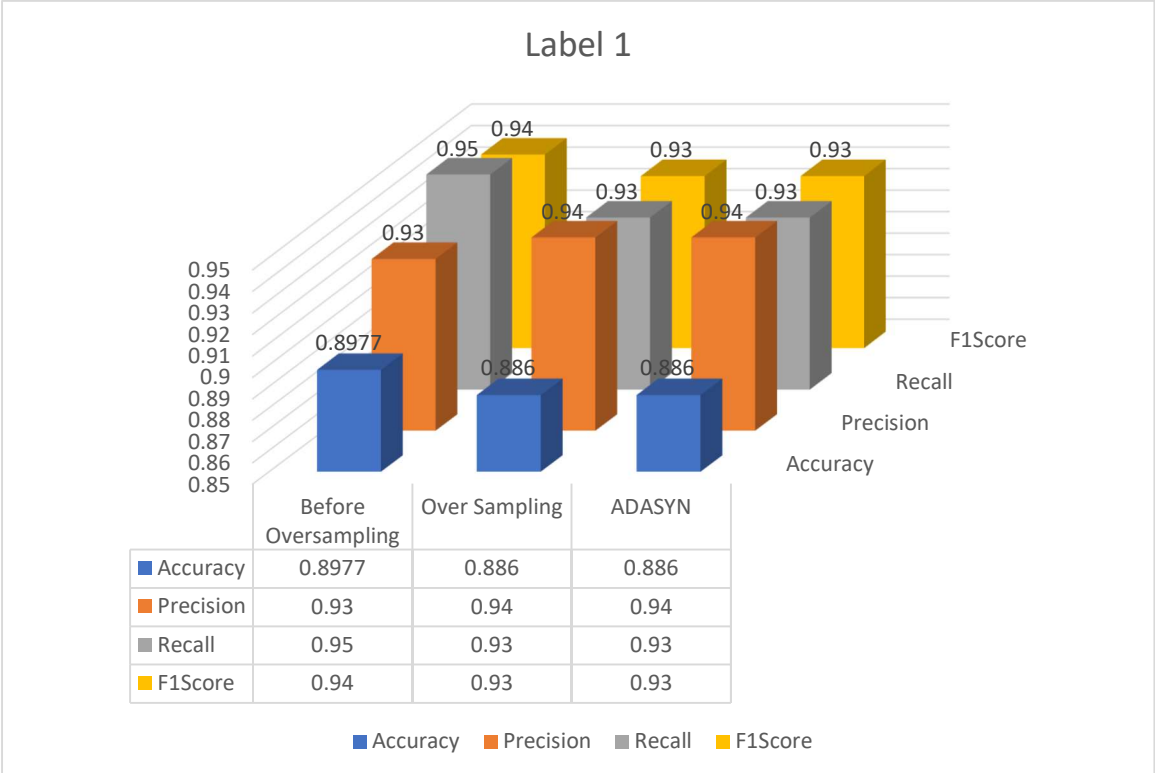
|:
```
print(Counter(y_train))
print(Counter(y_train_os))
```

```
Counter({1: 117045, 0: 18795})
Counter({1: 117045, 0: 93636})
```

|:
```
bgc=BaggingClassifier()
bgc.fit(x_train_os,y_train_os)
ypred=bgc.predict(x_test)
print(confusion_matrix(y_test,ypred))
print(classification_report(y_test,ypred))
print(accuracy_score(y_test,ypred))
```

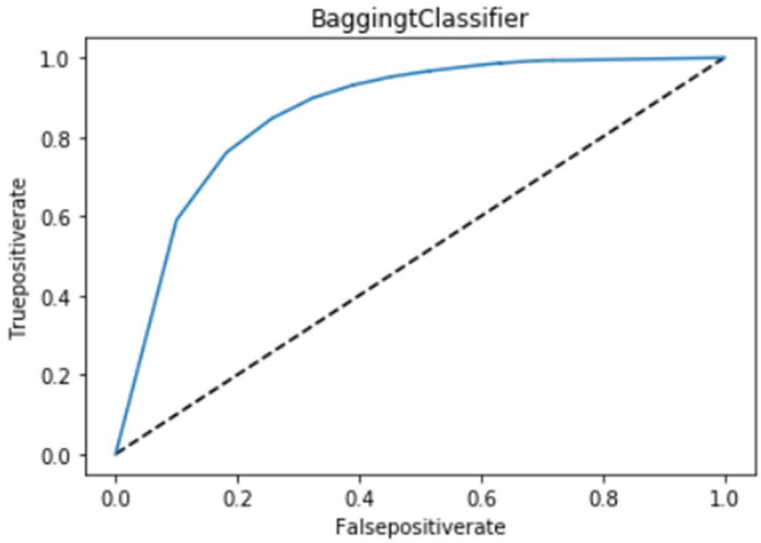
```
[[ 3234  2067]
 [ 2285 30728]]
              precision    recall  f1-score   support

           0       0.59      0.61      0.60      5301
           1       0.94      0.93      0.93     33013

    accuracy                           0.89     38314
   macro avg       0.76      0.77      0.77     38314
weighted avg       0.89      0.89      0.89     38314

0.8864122774964764
```


Label 0
0.8977
0.65
0.56
0.886
0.59
0.61
0.886
0.59
0.61
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
F1Score
Recall
Precision
Accuracy
Before Oversampling
Over Sampling
ADASYN
Accuracy 0.8977 0.886 0.886
Precision 0.65 0.59 0.59
Recall 0.56 0.61 0.61
F1Score 0.6 0.6 0.6
Accuracy
Precision
Recall
F1Score


Label 1
0.8977
0.93
0.95
0.94
0.886
0.94
0.93
0.93
0.886
0.94
0.93
0.93
0.95
0.94
0.93
0.92
0.91
0.9
0.89
0.88
0.87
0.86
0.85
F1Score
Recall
Precision
Accuracy
Before Oversampling
Over Sampling
ADASYN
Accuracy 0.8977 0.886 0.886
Precision 0.93 0.94 0.94
Recall 0.95 0.93 0.93
F1Score 0.94 0.93 0.93
Accuracy
Precision
Recall
F1Score

Recall has been increases from 0.56 to 0.61.

AUC/ROC Curve

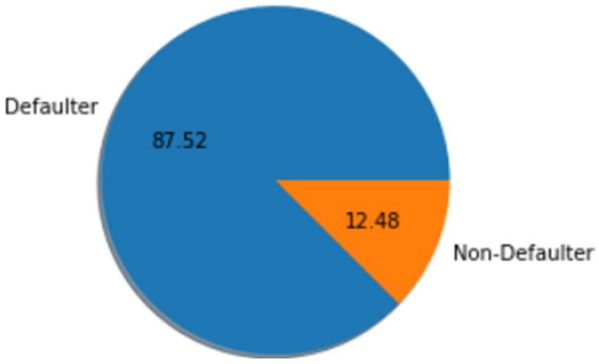



```
In [ ]: 1 auc_score=roc_auc_score(y_test,bgc.predict(x_test))

In [ ]: 1 auc_score

Out[ ]: 0.7704292066795865
```

So 0.77 is a good value which means our model can diffrentiate between defaulter and non defaulters correctly by 77%.

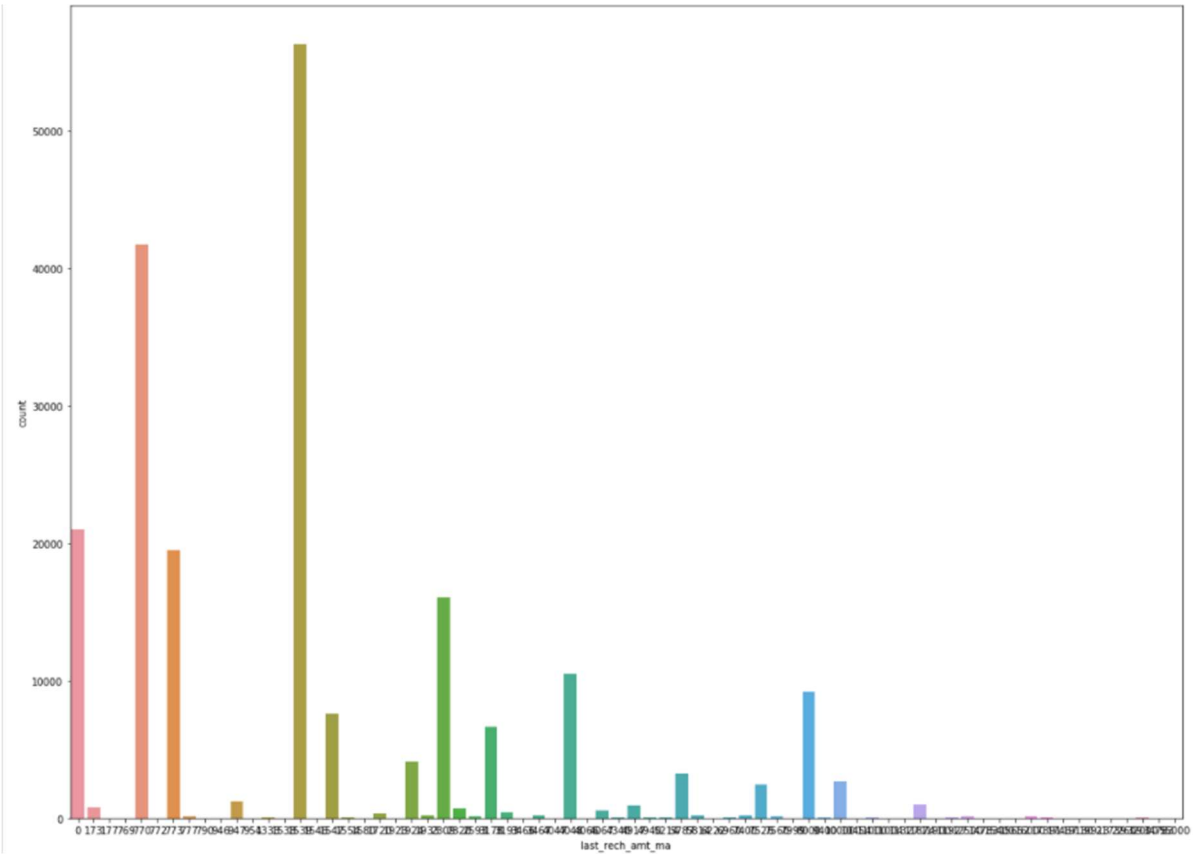
Visualizations

1. label



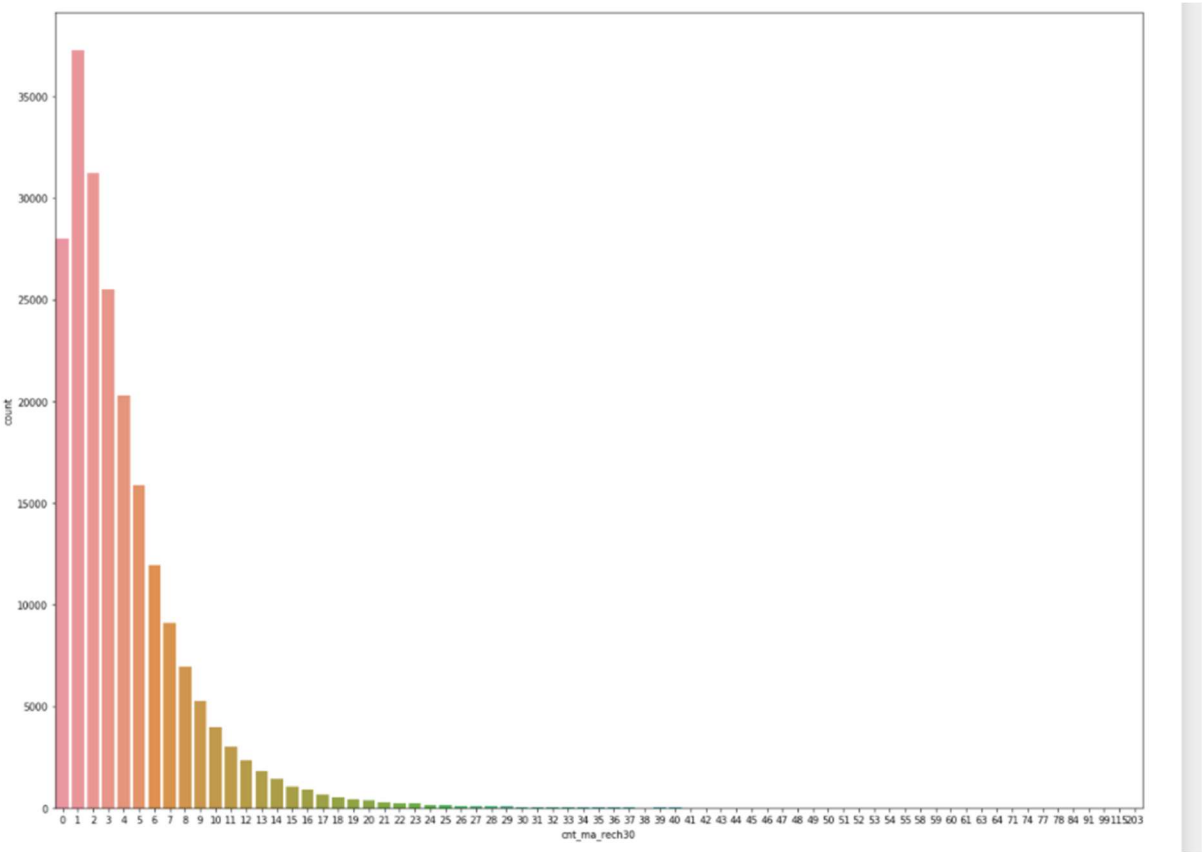
88% of the customers have paid the loan in time whereas the remaining have failed to do so.

2. last_rech_amt_ma



We can observe that 20995 customers have maintained 0 balance which means these groupl of people are potential customers of credit.

3. cnt_ma_rech30



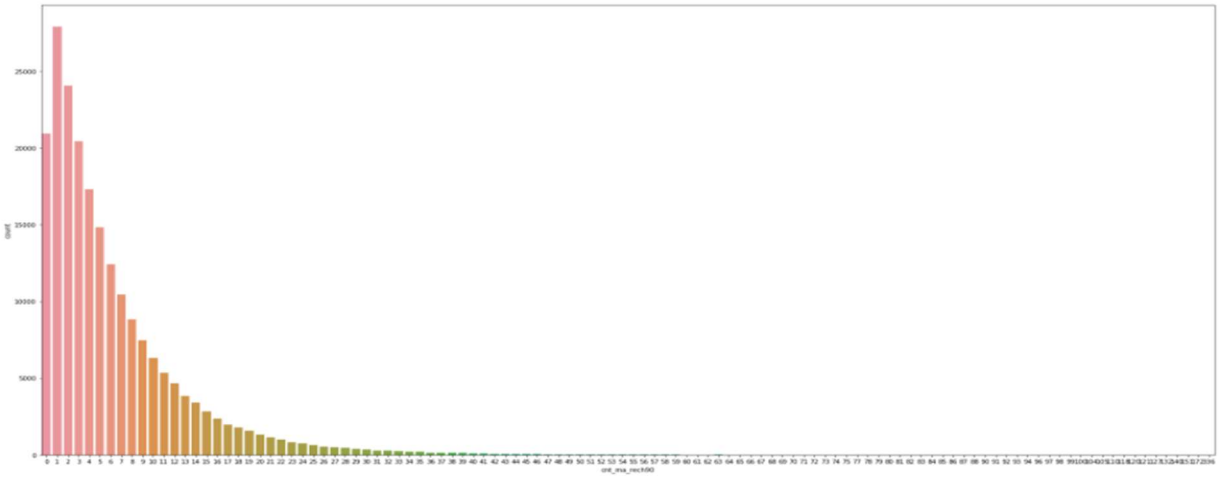
27979 of the customers have never recharged the balance from past 30 days.They can be potential customers as well.

4.cnt_ma_rech90

cnt_ma_rech90

```
plt.figure(figsize=(30,15))
sns.countplot(data["cnt_ma_rech90"])
print(data["cnt_ma_rech90"].value_counts())
```

```
1      27898
2      24052
0      20950
3      20446
4      17329
       ...
151        1
336        1
121        1
172        1
127        1
Name: cnt_ma_rech90, Length: 110, dtype: int64
```



20950 of the customers have never recharged the balance from past 90 days.They can be potential customers as well.

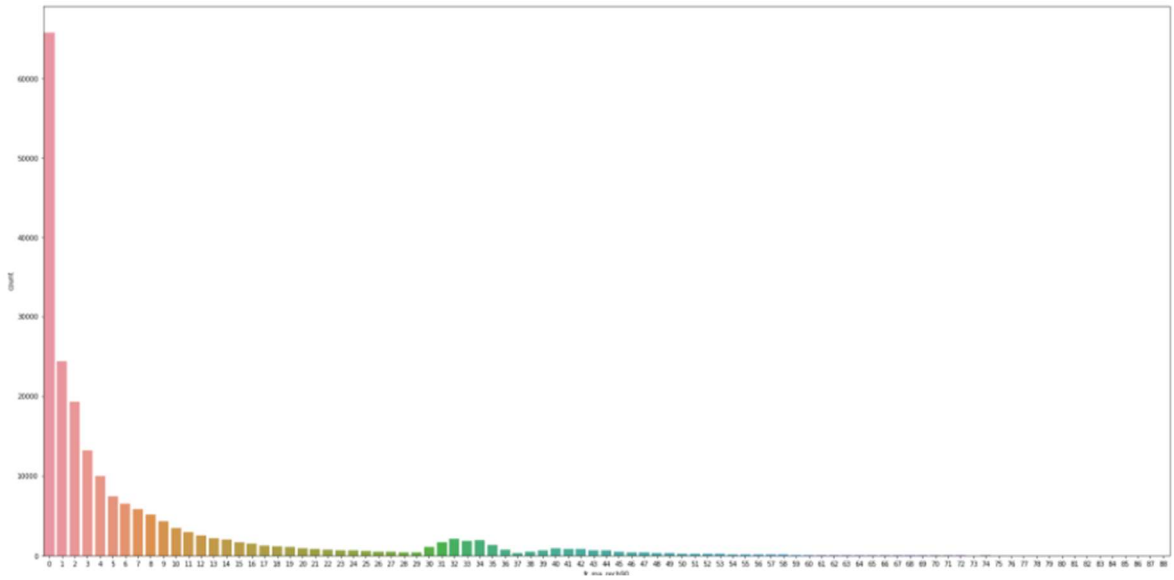
¶

5.fr_ma_rech90

fr_ma_rech90

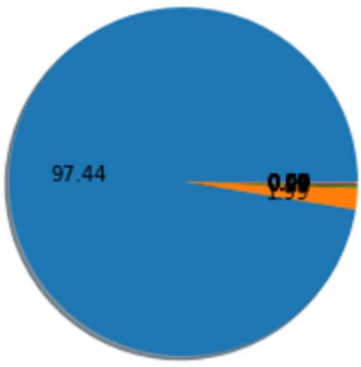
```
plt.figure(figsize=(30,15))
sns.countplot(data["fr_ma_rech90"])
print(data["fr_ma_rech90"].value_counts())
```

```
0     65753
1     24373
2     19285
3     13192
4     10021
      ...
80        7
81        7
88        5
84        4
87        1
Name: fr_ma_rech90, Length: 89, dtype: int64
```



65753 maintain a frequency of 0 means they have not recharged from past 90 days.

6. cnt_da_rech90



In [20]:

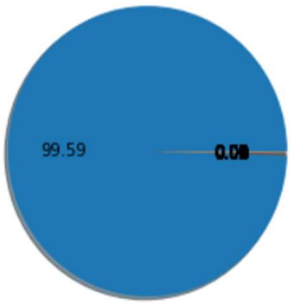
98% of the customers have not recharged the data in past 3 months,this group of customers might take loan or,they are aslo chances that they have not paid the loan

7.fr_da_rech90

fr_da_rech90

```
plt.pie(data["fr_da_rech90"].value_counts(),autopct="%.2f",shadow=True)
print(data["fr_da_rech90"].value_counts())
```

```
43    1
42    1
39    1
64    1
Name: fr_da_rech90, dtype: int64
```



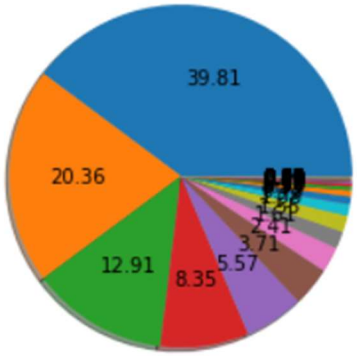
99.59% of the customers have never recharged in past 90 days

8.cnt_loans30

cnt_loans30

```
plt.pie(data["cnt_loans30"].value_counts(),autopct="%.2f",shadow=True)
print(data["cnt_loans30"].value_counts())
```

```
44    1
36    1
35    1
50    1
Name: cnt_loans30, dtype: int64
```



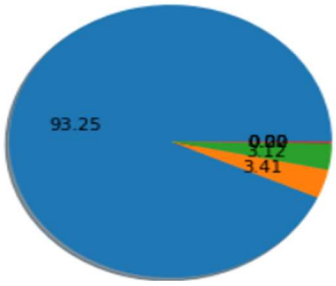
There are very less or easily countable number of customers who have taken loan more than 30 times a month whereas there are 83432customers who have taken loan only once in a month.

9.medianamnt_loans30

medianamnt_loans30

```
plt.pie(data["medianamnt_loans30"].value_counts(),autopct="%.2f",shadow=True)
print(data["medianamnt_loans30"].value_counts())
```

```
0.0    195445
1.0      7149
0.5      6538
2.0       420
1.5        38
3.0         3
Name: medianamnt_loans30, dtype: int64
```

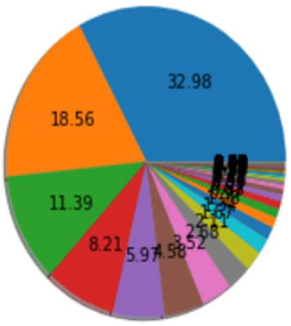


10.amnt_loans90¶

amnt_loans90

```
plt.pie(data["amnt_loans90"].value_counts(),autopct="%.2f",shadow=True)
print(data["amnt_loans90"].value_counts())

```

```
6      69131
12     38908
18     23867
24     17216
30     12503
       ...
360        1
426        1
396        1
438        1
342        1
Name: amnt_loans90, Length: 69, dtype: int64
```



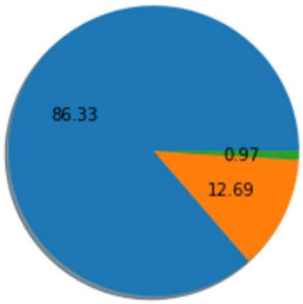
Most of the people have taken of loan of amount 6 in the past 90 days

11.maxamnt_loans90

maxamnt_loans90

```
]: 1 plt.pie(data["maxamnt_loans90"].value_counts(),autopct="%.2f",shadow=True)
   2 print(data["maxamnt_loans90"].value_counts())
   3
```

```
6     180945
12     26605
0       2043
Name: maxamnt_loans90, dtype: int64
```



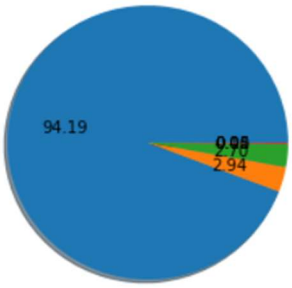
Customers take loan amount of 6 more than 12.

12.medianamnt_loans90

medianamnt_loans90

```
7]: 1 plt.pie(data["medianamnt_loans90"].value_counts(),autopct="%.2f",shadow=True)
   2 print(data["medianamnt_loans90"].value_counts())
```

```
0.0    197424
1.0      6172
0.5      5668
2.0       307
1.5        19
3.0         3
Name: medianamnt_loans90, dtype: int64
```



Interpretation of the Results

1. The data set is imbalanced.
2. All the feature variables have outliers.
3. Accuracy is not the suitable metrics to evaluate the performance.

CONCLUSION

- ## 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.

  Visualization gives the idea about the contribution of each feature in deciding whether it is defaulter and non-defaulter.

  Heat maps helps to understand better the relationship between other feature variables not just the target variable.

  Even though accuracies are good the f1 score are comparatively low

  Hence they might fail in distinguishing between defaulters and non-defaulters.

  Over sampling plays a major role in increasing the f1 score considerably.