

NAME OF THE PROJECT

**Micro-Credit Defaulter Model**

Submitted by:

Bishwajit Bhattacharya

**ACKNOWLEDGMENT**

# I take Data from “internshipfliprobo.xlsx”. I taken the help from “A Complete Machine Learning Project Walk-Through in Python” bye William Koehrsen.

**INTRODUCTION**

Problem Statement: -

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).

In data set the target columns is ‘label’ which has integer value so we will apply classification technique.

We want to develop a model that is both **accurate**— it can predict the label close to the true value — and **interpretable** — we can understand the model predictions. Once we know the goal, we can use it to guide our decisions as we dig into the data and build models.

# Data Cleaning

Contrary to what most data science courses would have you believe, not every dataset is a perfectly curated group of observations with no missing values or anomalies (looking at you [mtcars](http://stat.ethz.ch/R-manual/R-devel/library/datasets/html/mtcars.html" \t "_blank) and[iris](https://archive.ics.uci.edu/ml/datasets/iris) datasets). Real-world data is messy which means we need to[clean and wrangle](https://www.springboard.com/blog/data-wrangling/) it into an acceptable format before we can even start the analysis. Data cleaning is an un-glamorous, but necessary part of most actual data science problems.

First, we can load in the data as a Pandas DataFrame and take a look:

import pandas as pd

import numpy as np

import seaborn as sns

df = pd.DataFrame(pd.read\_excel("internshipfliprobo.xlsx"))

df

Unnamed: 0 label msisdn aon daily\_decr30 daily\_decr90 \

0 1 0 21408I70789 272.0 3055.050000 3065.150000

1 2 1 76462I70374 712.0 12122.000000 12124.750000

2 3 1 17943I70372 535.0 1398.000000 1398.000000

3 4 1 55773I70781 241.0 21.228000 21.228000

4 5 1 03813I82730 947.0 150.619333 150.619333

... ... ... ... ... ... ...

209588 209589 1 22758I85348 404.0 151.872333 151.872333

209589 209590 1 95583I84455 1075.0 36.936000 36.936000

209590 209591 1 28556I85350 1013.0 11843.111667 11904.350000

209591 209592 1 59712I82733 1732.0 12488.228333 12574.370000

209592 209593 1 65061I85339 1581.0 4489.362000 4534.820000

rental30 rental90 last\_rech\_date\_ma last\_rech\_date\_da ... \

0 220.13 260.13 2.0 0.0 ...

1 3691.26 3691.26 20.0 0.0 ...

2 900.13 900.13 3.0 0.0 ...

3 159.42 159.42 41.0 0.0 ...

4 1098.90 1098.90 4.0 0.0 ...

... ... ... ... ... ...

209588 1089.19 1089.19 1.0 0.0 ...

209589 1728.36 1728.36 4.0 0.0 ...

209590 5861.83 8893.20 3.0 0.0 ...

209591 411.83 984.58 2.0 38.0 ...

209592 483.92 631.20 13.0 0.0 ...

[show more (open the raw output data in a text editor) ...](vscode-file://vscode-app/c:/Users/bisue/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

209589 2016-06-12 209590 2016-07-29 209591 2016-07-25 209592 2016-07-07 [209593 rows x 37 columns]

Total data set has 209593 rows and 37 columns. First we need to check is there any null value, also we need to check what is the data type of the data set.

df.isnull().sum()

Unnamed: 0 0

label 0

msisdn 0

aon 0

daily\_decr30 0

daily\_decr90 0

rental30 0

rental90 0

last\_rech\_date\_ma 0

last\_rech\_date\_da 0

last\_rech\_amt\_ma 0

cnt\_ma\_rech30 0

fr\_ma\_rech30 0

sumamnt\_ma\_rech30 0

medianamnt\_ma\_rech30 0

medianmarechprebal30 0

cnt\_ma\_rech90 0

fr\_ma\_rech90 0

sumamnt\_ma\_rech90 0

medianamnt\_ma\_rech90 0

medianmarechprebal90 0

cnt\_da\_rech30 0

fr\_da\_rech30 0

cnt\_da\_rech90 0

fr\_da\_rech90 0

[show more (open the raw output data in a text editor) ...](vscode-file://vscode-app/c:/Users/bisue/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

medianamnt\_loans90 0 payback30 0 payback90 0 pcircle 0 pdate 0 dtype: int64

So there is no null value on dataset.

**Checking Datatypes**

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 int64

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

[show more (open the raw output data in a text editor) ...](vscode-file://vscode-app/c:/Users/bisue/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

medianamnt\_loans90 float64 payback30 float64 payback90 float64 pcircle object pdate datetime64[ns] dtype: object

so we have mixed data set of float,int and object.

# Exploratory Data Analysis

# Now that the tedious — but necessary — step of data cleaning is complete, we can move on to exploring our data! [Exploratory Data Analysis](https://en.wikipedia.org/wiki/Exploratory_data_analysis)(EDA) is an open-ended process where we calculate statistics and make figures to find trends, anomalies, patterns, or relationships within the data.

In short, the goal of EDA is to learn what our data can tell us. It generally starts out with a high level overview, then narrows in to specific areas as we find interesting parts of the data. The findings may be interesting in their own right, or they can be used to inform our modelling choices, such as by helping us decide which features to use.

**Eda for Nominal Data**

ax = sns.countplot(x="pcircle", data=df\_visualization\_nominal)

print(df\_visualization\_nominal["pcircle"].value\_counts())

UPW 209593

Name: pcircle, dtype: int64

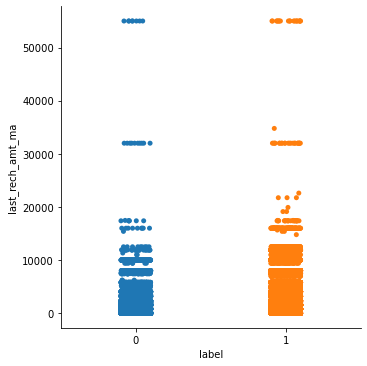


As per this eda nominal value pcircle has upw and count as axis a dataflow is normal.

# Eda for ordinal data.

sns.catplot(x="label", y="last\_rech\_amt\_ma", data=df)

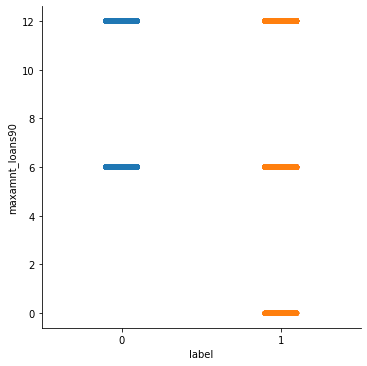
<seaborn.axisgrid.FacetGrid at 0x1bbc1b405b0>



The values is showing as 0 and 1 and the column is looks like similer.

sns.catplot(x="label", y="maxamnt\_loans90", data=df)

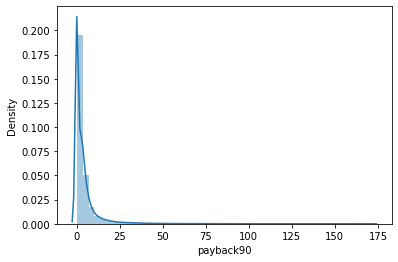
<seaborn.axisgrid.FacetGrid at 0x1bbc26a56d0>



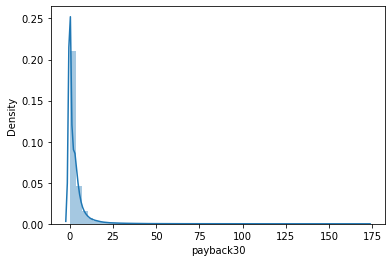
In this Eda analysis we saw little difference in 0’s and 1’s

**Eda for float data**

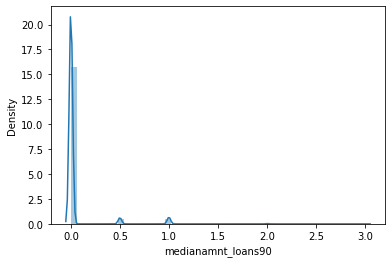
sns.distplot(df\_visualization\_continuous['payback90'], kde=True)

****

sns.distplot(df\_visualization\_continuous['payback30'], kde=True)

****

sns.distplot(df\_visualization\_continuous['medianamnt\_loans90'], kde=True)

****

Eda is showing graph is little disturbing we need work on that, before this we need to change the datatype it should not be object type.

import sklearn

from sklearn.preprocessing import OrdinalEncoder

enc=OrdinalEncoder()

for i in df.columns:

    if df[i].dtypes=="object":

        df[i]=enc.fit\_transform(df[i].values.reshape(-1,1))

bye this code datatype is successfully changed.

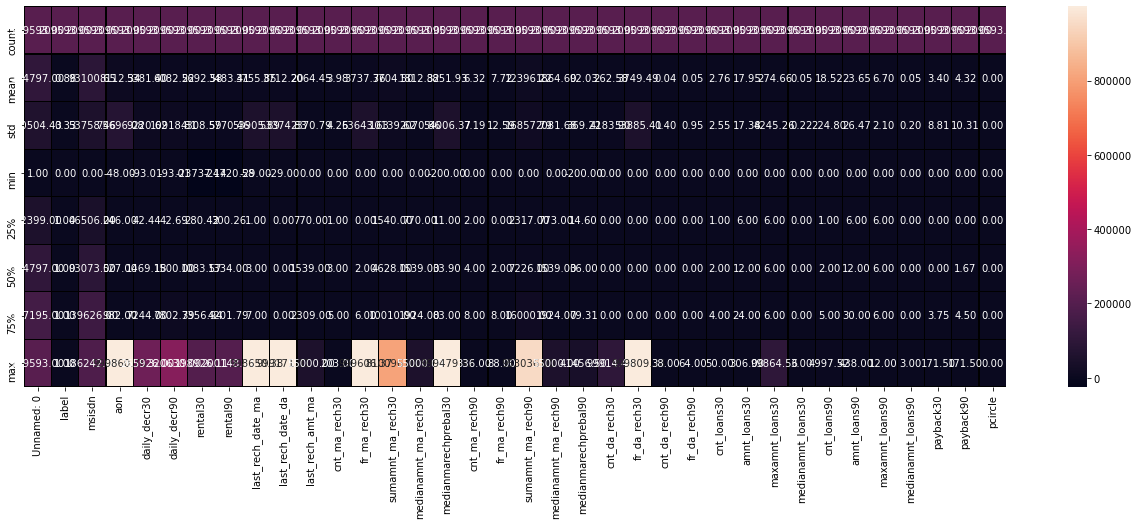
Now we will see heatmap

import matplotlib.pyplot as plt

plt.figure(figsize=(22,7))

sns.heatmap(df.describe(),annot=True,linewidths=0.1,linecolor="black",fmt="0.2f")

<AxesSubplot:>

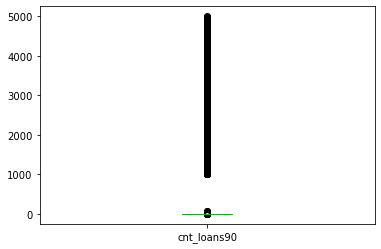


So we done with the visualization

**Now we will remove outlier**

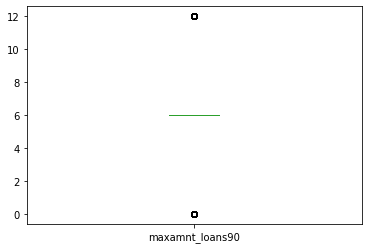
df['cnt\_loans90'].plot.box()

<AxesSubplot:>

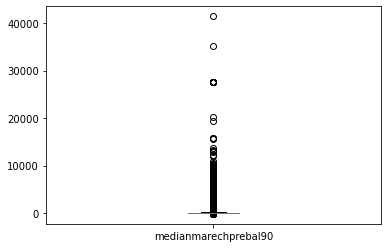


df['maxamnt\_loans90'].plot.box()

<AxesSubplot:>



df['medianmarechprebal90'].plot.box()



# Autometic Outlier Removal[¶](http://localhost:8888/notebooks/Micro-Credit%20Defaulter%20Model.ipynb#Autometic-Outlier-Removal)

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.neighbors import LocalOutlierFactor

from sklearn.metrics import mean\_absolute\_error

data = df.values

X, Y = data[:, :-1], data[:, -1]

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.33, random\_state=1)

print(X\_train.shape, Y\_train.shape)

(140427, 35) (140427,)

lof = LocalOutlierFactor()

yhat = lof.fit\_predict(X\_train)

mask = yhat != -1

X\_train, Y\_train = X\_train[mask, :], Y\_train[mask]

print(X\_train.shape, Y\_train.shape)

(138465, 35) (138465,)

model = LinearRegression()

model.fit(X\_train, Y\_train)

LinearRegression()

yhat = model.predict(X\_test)

mae = mean\_absolute\_error(Y\_test, yhat)

print('MAE: %.3f' % mae)

MAE: 0.000

dataloss=((140427-138465)/140427)\*100

dataloss

1.3971672114336986

data loss is less than 10 so we can go ahead.

**Target and Features**

features**=**df**.**drop("label",axis**=**1)

target**=**df["label"]

x**=**features

y**=**target

**Classification analysis**

ad**=**AdaBoostClassifier()

ad**.**fit(xtrain,ytrain)

ad\_pred**=**ad**.**predict(xtest)

print(accuracy\_score(ytest,ad\_pred))

print(confusion\_matrix(ytest,ad\_pred))

print(classification\_report(ytest,ad\_pred))

0.9025096218073094

[[ 2254 5568]

[ 562 54494]]

precision recall f1-score support

0 0.80 0.29 0.42 7822

1 0.91 0.99 0.95 55056

accuracy 0.90 62878

macro avg 0.85 0.64 0.69 62878

dtc**=**DecisionTreeClassifier()

dtc**.**fit(xtrain,ytrain)

dtc**.**score(xtrain,ytrain)

preddtc**=**dtc**.**predict(xtest)

print(accuracy\_score(ytest,preddtc))

print(confusion\_matrix(ytest,preddtc))

print(classification\_report(ytest,preddtc))

0.863656604853844

[[ 3788 4034]

[ 4539 50517]]

precision recall f1-score support

0 0.45 0.48 0.47 7822

1 0.93 0.92 0.92 55056

accuracy 0.86 62878

macro avg 0.69 0.70 0.70 62878

weighted avg 0.87 0.86 0.87 62878

knn**=**KNeighborsClassifier(n\_neighbors**=**5)

knn**.**fit(xtrain,ytrain)

knn**.**score(xtrain,ytrain)

predknn**=**knn**.**predict(xtest)

print(accuracy\_score(ytest,predknn))

print(confusion\_matrix(ytest,predknn))

print(classification\_report(ytest,predknn))

0.8624002035688157

[[ 1378 6444]

[ 2208 52848]]

precision recall f1-score support

0 0.38 0.18 0.24 7822

1 0.89 0.96 0.92 55056

accuracy 0.86 62878

macro avg 0.64 0.57 0.58 62878

weighted avg 0.83 0.86 0.84 62878

I created 3 train test model best output giving AdaBoostClassifier()

It is given almost 90% accuracy.

**Loading Model:-**

**for** m **in** model:

m**.**fit(xtrain,ytrain)

m**.**score(xtrain,ytrain)

predm**=**m**.**predict(xtest)

print('Accuracy score of',m,'is:')

print(accuracy\_score(ytest,predm))

print(confusion\_matrix(ytest,predm))

print(classification\_report(ytest,predm))

print('\n')

Accuracy score of DecisionTreeClassifier() is:

0.8635452781577022

[[ 3815 4007]

[ 4573 50483]]

precision recall f1-score support

0 0.45 0.49 0.47 7822

1 0.93 0.92 0.92 55056

accuracy 0.86 62878

macro avg 0.69 0.70 0.70 62878

weighted avg 0.87 0.86 0.87 62878

Accuracy score of SVC() is:

0.8756003689684786

[[ 0 7822]

[ 0 55056]]

precision recall f1-score support

0 0.00 0.00 0.00 7822

1 0.88 1.00 0.93 55056

accuracy 0.88 62878

macro avg 0.44 0.50 0.47 62878

weighted avg 0.77 0.88 0.82 62878

Accuracy score of KNeighborsClassifier() is:

0.8624002035688157

[[ 1378 6444]

[ 2208 52848]]

precision recall f1-score support

0 0.38 0.18 0.24 7822

1 0.89 0.96 0.92 55056

accuracy 0.86 62878

macro avg 0.64 0.57 0.58 62878

weighted avg 0.83 0.86 0.84 62878

Accuracy score of AdaBoostClassifier() is:

0.9025096218073094

[[ 2254 5568]

[ 562 54494]]

precision recall f1-score support

0 0.80 0.29 0.42 7822

1 0.91 0.99 0.95 55056

accuracy 0.90 62878

macro avg 0.85 0.64 0.69 62878

weighted avg 0.89 0.90 0.88 62878

**Loading the model**

**import** joblib

joblib**.**dump(ad,'micro.obj')

ad\_from\_joblib**=**joblib**.**load('micro.obj')

ad\_from\_joblib**.**predict(xtest)

array([1, 1, 1, ..., 1, 1, 0])

**import** pickle

filename **=** 'pickleadfile.pkl'

pickle**.**dump(ad, open(filename, 'wb'))

loaded\_model **=** pickle**.**load(open(filename,'rb'))

loaded\_model**.**predict(xtest)

array([1, 1, 1, ..., 1, 1, 0])

**Checking the model**

score**=**cross\_val\_score(ad,x,y,cv**=**15)

print(score)

print(score**.**mean())

print(score**.**std())

[0.90431547 0.90388607 0.90603306 0.90173907 0.90281257 0.90560366

0.9023116 0.90474487 0.9057468 0.90474487 0.90302727 0.902741

0.902741 0.90344976 0.90438019]

0.9038851491206642

0.0012836628943687421

kfold**=**KFold(5)

print(score)

print(score**.**mean())

print(score**.**std())

[0.90431547 0.90388607 0.90603306 0.90173907 0.90281257 0.90560366

0.9023116 0.90474487 0.9057468 0.90474487 0.90302727 0.902741

0.902741 0.90344976 0.90438019]

0.9038851491206642

0.0012836628943687421

**Conclusion :-**

As I create the model , I can say that model is giving 90% accuracy , which is quite good enough. I done with the Kfold score mean( ) and std() also and as we see the output is 90.43%.

Its good enough and model is good to go.