A
Data science
Project
on

“Micro Credit Defaulter Project”

Submitted by:

Santosh Arvind Dharam

ACKNOWLEDGMENT

I feel great pleasure to present the Project entitled “Micro Credit Defaulter Project”. But it would be unfair on our part if I do not acknowledge efforts of some of the people without the support of whom, this Project would not have been a success. First and for most I am very much thankful to my respected SME ‘shrishti Maan’ for his leading guidance in this Project. Also he has been persistent source of inspiration to me. I would like to express my sincere thanks and appreciation to ‘flip robo’ for their valuable support. Most importantly I would like to express our sincere gratitude towards my Friend & Family for always being there when I needed them most.

Mr. Santosh Arvind Dharam

INTRODUCTION

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 be6 (in Indonesian

Rupiah), while, for the loan amount of 10(in Indonesian Rupiah), the payback amount should be 12(in Indonesian Rupiah).

The sample data is provided to us from our client database. It is hereby given to you for this exercise. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

PROBLEM STATEMENT

- In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.
- 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. In this case, Label ‘1’ indicates that the loan has been payed i.e. Non- defaulter, while, Label ‘0’ indicates that the loan has not been payed i.e. defaulter.

Analytical Problem Framing

EDA steps:

Following EDA steps will use for analytical problem framing

1) import necessary libraries:

first we will import all the necessary libraries which will be usefull for analysis of data

```
In [1]: #import all libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')
        from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
        from sklearn.linear_model import LogisticRegression,Lasso,LinearRegression
        from sklearn.neighbors import KNeighborsRegressor
        from sklearn.svm import SVR
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.ensemble import AdaBoostRegressor,GradientBoostingRegressor
        from sklearn.preprocessing import LabelEncoder,StandardScaler
        from sklearn.model_selection import train_test_split,GridSearchCV
        from sklearn.decomposition import PCA
        from scipy.stats import zscore
        from sklearn.model_selection import cross_val_score
```

in this case we have to import all the necessary library that are useful for data analysis in jupyter notebook so we have imported all the necessary libraries

2)extract the dataset in jupyter notebook:

now will extract the data from csv file by pandas library

```
In [2]: data=pd.read_csv("C:\\Users\\SAI BABA\\Desktop\\micro credit defaulter project\\Data file.csv")
        data.head()
```

Out[2]:

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	medianamr
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	

5 rows × 37 columns

```
In [3]: data.shape
```

Out[3]: (209593, 37)

so total we have 209593 rows and 37 columns

Data is extracted for further analysis in jupyter notebook

3)Data Description:

In this case data is described in detail which helping us for detail analysis

```
In [6]: data.describe()
```

Out[6]:

	Unnamed: 0	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_r
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	20
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	999880.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	5

8 rows × 34 columns

this describe the whole data set with its total count ,mean ,std along with minimum to maximum values present in the dataset of perticular column

it gives the detail description of data with its mean,std,mini to max values present in the data column

4)Encoding the dataset:

In this case as our data contains some categorical column having object type of data it is necessary to convert it into numerical form by LabelEncoder

lets will do some label encoding of object type column for further analysis by labelencoder

DATA ENCODING

In [12]:
```
from sklearn.preprocessing import LabelEncoder
```

In [13]:
```
le=LabelEncoder()
label=le.fit_transform(data["msisdn"])
label
data=data.drop("msisdn",axis='columns')
data["msisdn"]=label
```

In [14]:
```
le=LabelEncoder()
label=le.fit_transform(data["pcircle"])
label
data=data.drop("pcircle",axis='columns')
data["pcircle"]=label
```

In [15]:
```
le=LabelEncoder()
label=le.fit_transform(data["pdate"])
label
data=data.drop("pdate",axis='columns')
data["pdate"]=label
data
```

Out[15]:

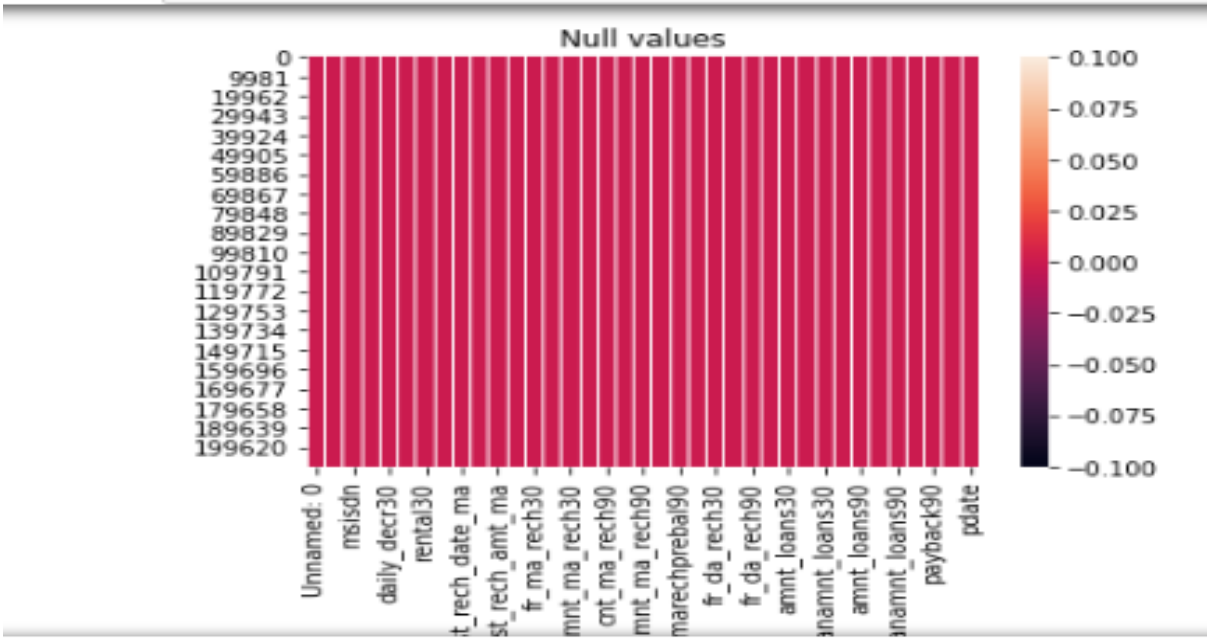
	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_rech30	...	medianamnt_
0	0	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	2	...	
1	1	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	1	...	
2	1	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	1	...	
3	1	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	0	...	
4	1	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	7	...	
...	...	...	...	...	...	...	...	...	...	...	...	
209588	1	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	4048	3	...	
209589	1	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	773	4	...	

Data contains 37 columns and 209593 rows

5)checking null values:

In this case we have to find out the null values present in our data set if yes it is required to remove it. in our data set it does not have any null values it is also shown by heatmap

```
In [8]: sns.heatmap(data.isnull())
        plt.title("Null values")
        plt.show()
```



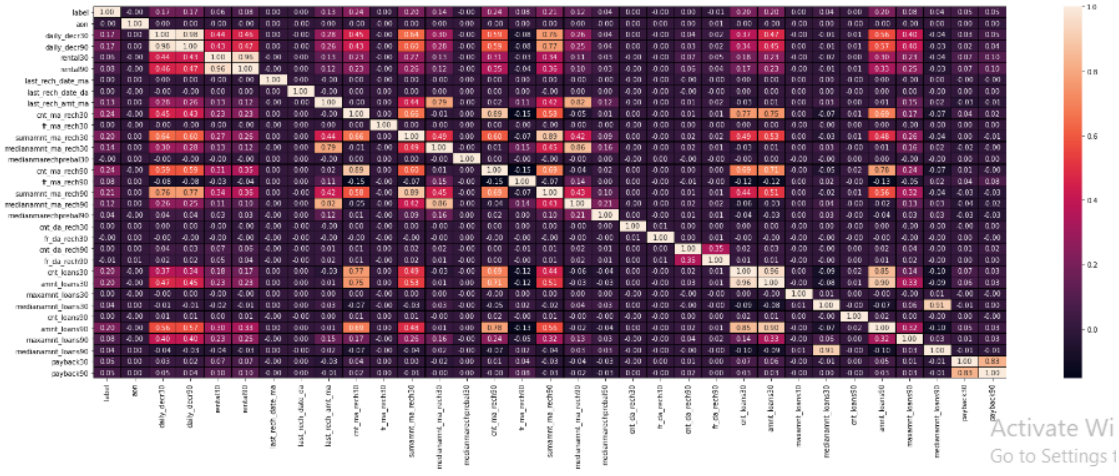
6)Data correlation:

It gives the correlation between each column with the target variable

we can also represent this correlation by heat map

```
In [11]: #heat map
         plt.figure(figsize=(30,10))
         sns.heatmap(data.corr(),annot=True,linewidths=0.2,linecolor='black',fmt='0.2f')
```

```
Out[11]: <AxesSubplot:>
```



it also gives the positive negative correlation of data with respective one another

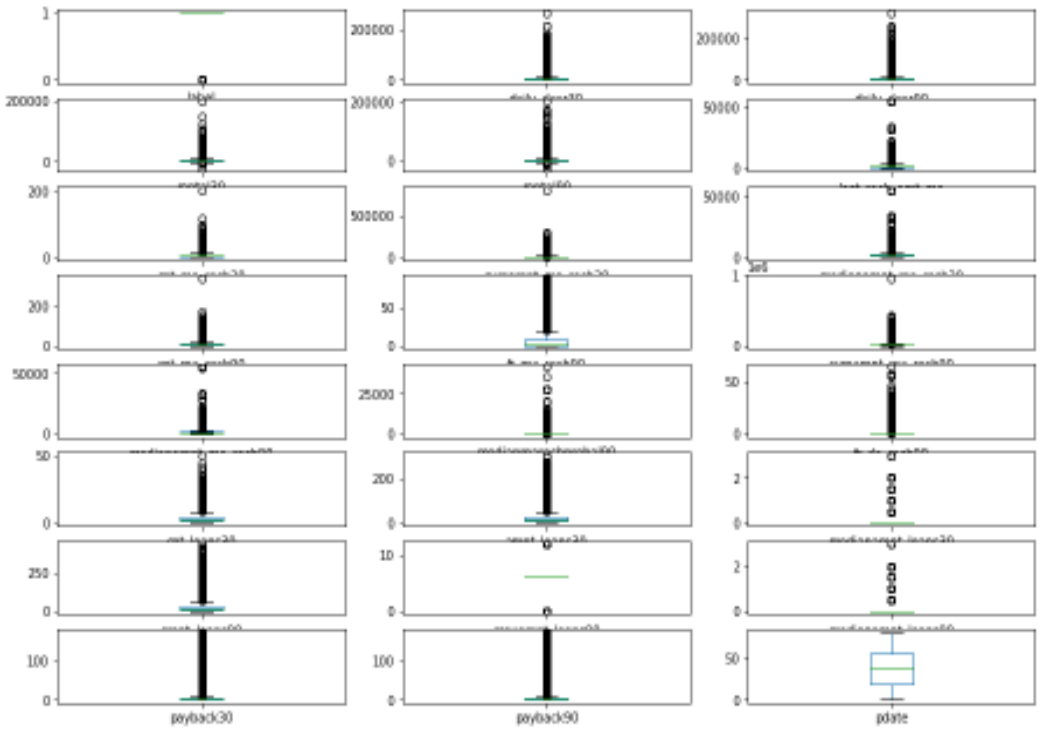
7)visualization:

a) Detection of outliers:

lets will check the outliers present in our dataset by box plot

```
In [21]: #plotting the boxplot of each column to check the outliers
         data.plot(kind='box',subplots = True,layout=(8,3),figsize = (15,18))
```

```
Out[21]: label                    AxesSubplot(0.125,0.799681;0.227941x0.0883191)
         daily_decr30          AxesSubplot(0.398529,0.799681;0.227941x0.0883191)
         daily_decr90          AxesSubplot(0.672059,0.799681;0.227941x0.0883191)
         rental30                 AxesSubplot(0.125,0.703298;0.227941x0.0883191)
         rental90              AxesSubplot(0.398529,0.703298;0.227941x0.0883191)
         last_rech_amt_ma      AxesSubplot(0.672059,0.703298;0.227941x0.0883191)
         cnt_ma_rech30            AxesSubplot(0.125,0.606915;0.227941x0.0883191)
         sumamt_ma_rech30      AxesSubplot(0.398529,0.606915;0.227941x0.0883191)
         medianamnt_ma_rech30  AxesSubplot(0.672059,0.606915;0.227941x0.0883191)
         cnt_ma_rech90            AxesSubplot(0.125,0.510532;0.227941x0.0883191)
         fr_ma_rech90          AxesSubplot(0.398529,0.510532;0.227941x0.0883191)
         sumamt_ma_rech90      AxesSubplot(0.672059,0.510532;0.227941x0.0883191)
         medianamnt_ma_rech90     AxesSubplot(0.125,0.414149;0.227941x0.0883191)
         medianmarechprebal90  AxesSubplot(0.398529,0.414149;0.227941x0.0883191)
         fr_da_rech90          AxesSubplot(0.672059,0.414149;0.227941x0.0883191)
         cnt_loans30              AxesSubplot(0.125,0.317766;0.227941x0.0883191)
         amnt_loans30          AxesSubplot(0.398529,0.317766;0.227941x0.0883191)
         medianamnt_loans30    AxesSubplot(0.672059,0.317766;0.227941x0.0883191)
         amnt_loans90             AxesSubplot(0.125,0.221383;0.227941x0.0883191)
         maxamnt_loans90       AxesSubplot(0.398529,0.221383;0.227941x0.0883191)
         medianamnt_loans90    AxesSubplot(0.672059,0.221383;0.227941x0.0883191)
         payback30                AxesSubplot(0.125,0.125;0.227941x0.0883191)
         payback90             AxesSubplot(0.398529,0.125;0.227941x0.0883191)
         pdate                 AxesSubplot(0.672059,0.125;0.227941x0.0883191)
         dtype: object
```



so outliers are present in our dataset lets will remove it

so data contains outliers present in it let will remove it in further steps can be removed with zscore test and by selecting appropriate threshold values

8)Outliers Removal:

So lets we will remove outliers present in the dataset

so outliers are present in our dataset lets will remove it

```
In [22]: #calculate the zscore
         z = np.abs(zscore(data))
         print(z)
```

```
[[2.64789583 0.25229941 0.27634619 ... 2.9046997  2.39409346 0.52239995]
 [0.37765836 0.73103667 0.5533797  ... 0.38562959 0.41923266 1.47739785]
 [0.37765836 0.43201111 0.42903256 ... 0.38562959 0.41923266 1.88668266]
 ...
 [0.37765836 0.70079045 0.53319431 ... 0.06820893 0.04735622 0.93168476]
 [0.37765836 0.77075515 0.59455827 ... 0.38562959 0.59938541 0.7497804 ]
 [0.37765836 0.09674426 0.14174607 ... 0.38562959 0.41923266 0.06878922]]
```

```
In [23]: threshold=3
         print(np.where(z<3))
         print(data.shape)
```

```
(array([     0,      0,      0, ..., 209592, 209592, 209592], dtype=int64), array([ 0,  1,  2, ..., 21, 22, 23], dtype=int64))
(209593, 24)
```

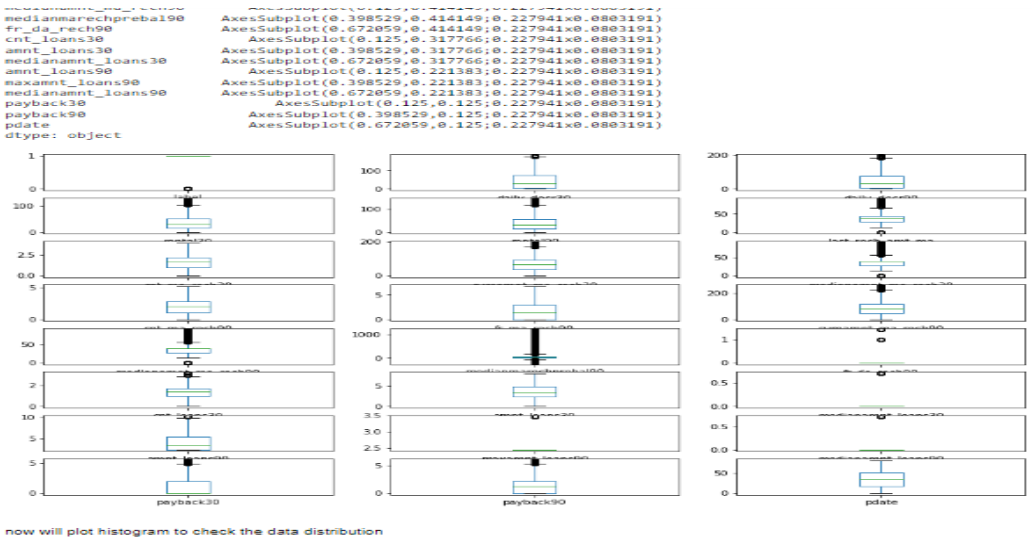
```
In [24]: data_new=data[(z<3).all(axis = 1)]
```

```
In [25]: print(data.shape)
         print(data_new.shape)
         data = data_new
         print('Shape after removing outlires',data.shape)
```

```
(209593, 24)
(169911, 24)
Shape after removing outlires (169911, 24)
```

So we have removed outliers present in the dataset so after removing the outliers we have 169911 rows and 37 columns left

9)data representation after removal outliers:



now will plot histogram to check the data distribution

So amount of outliers present in the dataset are removed which is also shown in the figure

10) Checking Skewness: in this case we have checked whether any skewness present in data or not some column has skewness in it we have remove it,will remove it with the help of sqrt method so,

lets will check skewness present in our dataset

```
In [26]: data.skew()
```

```
Out[26]: label                    -2.089439
daily_decr30              1.960544
daily_decr90              2.074719
rental30                  2.194306
rental90                  2.242482
last_rech_amt_ma          2.125557
cnt_ma_rech30             1.173387
sumamnt_ma_rech30         1.631699
medianamnt_ma_rech30      2.325874
cnt_ma_rech90             1.318110
fr_ma_rech90              1.983354
sumamnt_ma_rech90         1.703959
medianamnt_ma_rech90      2.373303
medianmarechprebal90      3.697595
fr_da_rech90             48.498884
cnt_loans30               1.465197
amnt_loans30              1.439681
medianamnt_loans30        5.346029
amnt_loans90              1.692563
maxamnt_loans90           2.680282
medianamnt_loans90        6.104944
payback30                 2.607510
payback90                 2.523879
pdate                     0.249516
dtype: float64
```

so lets will remove the skewness present in dataset by sqrt method

```
In [27]: #remove skewness
data['daily_decr30']=np.sqrt(data['daily_decr30'])
data['daily_decr90']=np.sqrt(data['daily_decr90'])
data['rental30']=np.sqrt(data['rental30'])
data['rental90']=np.sqrt(data['rental90'])
data['last_rech_amt_ma']=np.sqrt(data['last_rech_amt_ma'])
data['cnt_ma_rech30']=np.sqrt(data['cnt_ma_rech30'])
data['sumamnt_ma_rech30']=np.sqrt(data['sumamnt_ma_rech30'])
data['medianamnt_ma_rech30']=np.sqrt(data['medianamnt_ma_rech30'])
data['cnt_ma_rech90']=np.sqrt(data['cnt_ma_rech90'])
data['fr_ma_rech90']=np.sqrt(data['fr_ma_rech90'])
data['sumamnt_ma_rech90']=np.sqrt(data['sumamnt_ma_rech90'])
data['medianamnt_ma_rech90']=np.sqrt(data['medianamnt_ma_rech90'])
data['fr_da_rech90']=np.sqrt(data['fr_da_rech90'])
data['cnt_loans30']=np.sqrt(data['cnt_loans30'])
data['amnt_loans30']=np.sqrt(data['amnt_loans30'])
```

So after removing the skewness lets we will again check it as

```
In [28]: data.skew()
```

```
Out[28]: label                    -2.089439
         daily_decr30              0.832184
         daily_decr90              0.901643
         rental30                  0.933124
         rental90                  0.956804
         last_rech_amt_ma          0.400870
         cnt_ma_rech30            -0.112755
         sumamnt_ma_rech30         0.169014
         medianamnt_ma_rech30      0.252149
         cnt_ma_rech90             0.073673
         fr_ma_rech90              0.907526
         sumamnt_ma_rech90         0.312670
         medianamnt_ma_rech90      0.346892
         medianmarechprebal90      3.697595
         fr_da_rech90             44.674423
         cnt_loans30               0.778962
         amnt_loans30              0.753777
         medianamnt_loans30        5.346029
         amnt_loans90              0.956368
```

So skewness is removed

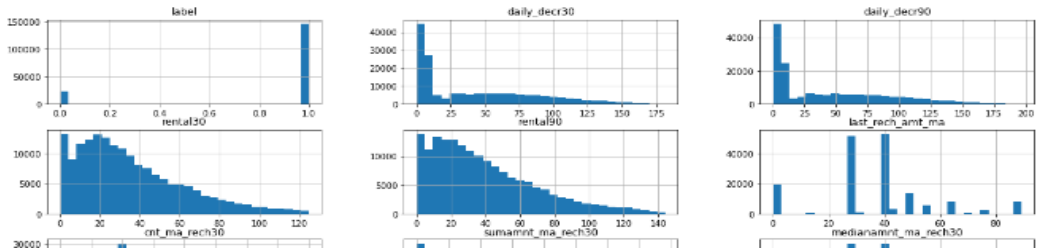
11)Data Distribution by histogram:

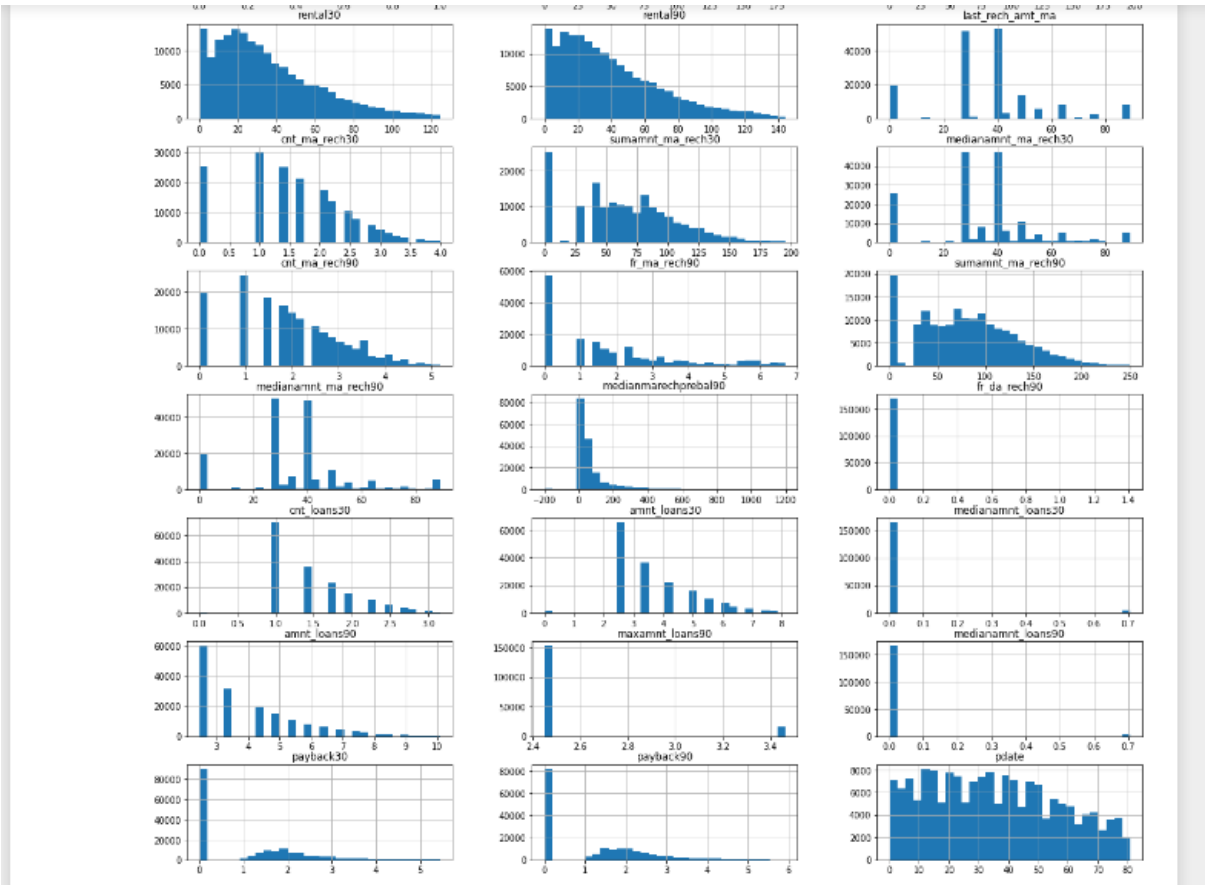
Now we will plot histogram to check how data is distributed

now will plot histogram to check the data distribution

```
In [36]: #plotting histogram for univariate analysis and checking the Normal Distribution
         data.hist(figsize=(20,20), grid = True, layout = (8,3), bins = 30)
```

```
Out[36]: array([[<AxesSubplot:title={'center':'label'}>,
                 <AxesSubplot:title={'center':'daily_decr30'}>,
                 <AxesSubplot:title={'center':'daily_decr90'}>],
                [<AxesSubplot:title={'center':'rental30'}>,
                 <AxesSubplot:title={'center':'rental90'}>,
                 <AxesSubplot:title={'center':'last_rech_amt_ma'}>],
                [<AxesSubplot:title={'center':'cnt_ma_rech30'}>,
                 <AxesSubplot:title={'center':'sumamnt_ma_rech30'}>,
                 <AxesSubplot:title={'center':'medianamnt_ma_rech30'}>],
                [<AxesSubplot:title={'center':'cnt_ma_rech90'}>,
                 <AxesSubplot:title={'center':'fr_ma_rech90'}>,
                 <AxesSubplot:title={'center':'sumamnt_ma_rech90'}>],
                [<AxesSubplot:title={'center':'medianamnt_ma_rech90'}>,
                 <AxesSubplot:title={'center':'medianmarechpreba190'}>,
                 <AxesSubplot:title={'center':'fr_da_rech90'}>],
                [<AxesSubplot:title={'center':'cnt_loans30'}>,
                 <AxesSubplot:title={'center':'amnt_loans30'}>,
                 <AxesSubplot:title={'center':'medianamnt_loans30'}>],
                [<AxesSubplot:title={'center':'amnt_loans90'}>,
                 <AxesSubplot:title={'center':'maxamnt_loans90'}>,
                 <AxesSubplot:title={'center':'medianamnt_loans90'}>],
                [<AxesSubplot:title={'center':'payback30'}>,
                 <AxesSubplot:title={'center':'payback90'}>,
                 <AxesSubplot:title={'center':'pdate'}>]], dtype=object)
```

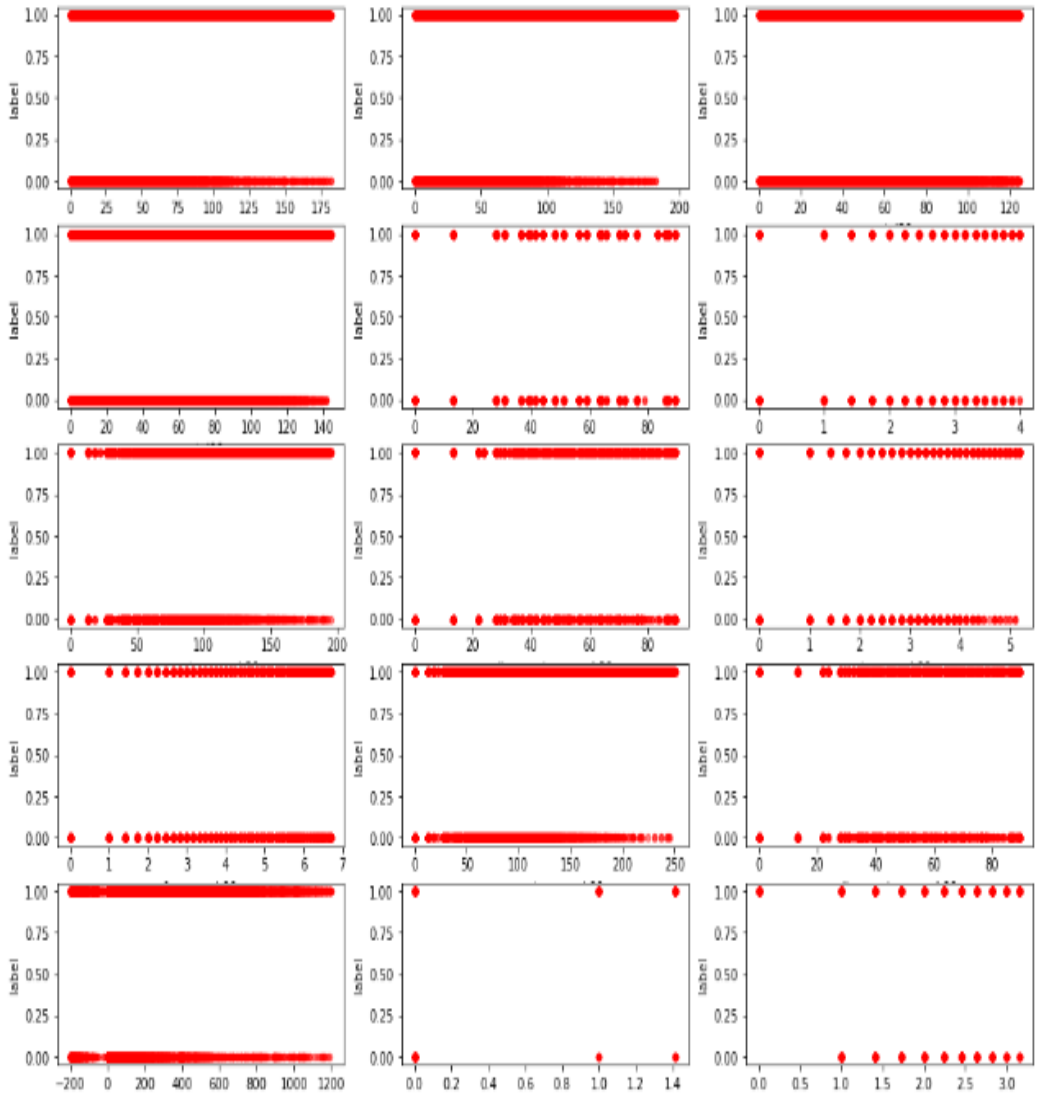




12)Scatter Plot: scatter plot help us how data is distributed with respective to target variable so lets we will plot scatter plot

```
In [37]: # setup figure
fig, axes = plt.subplots(nrows=8, ncols=3, figsize=(16, 20))

# iterate and plot subplots
for xcol, ax in zip(data.columns[1:], [x for v in axes for x in v]):
    data.plot.scatter(x=xcol, y='label', ax=ax, alpha=0.5, color='r')
```



It shows the distribution of data with respective target variable

13)Model Building:

Now we will built a model by dividing the dataset into x and y variables also we use the standard scalar to scale the data

In [38]:
```
#devide data set into feature and label
y=data['label']
x=data.drop(['label'],axis=1)
```

In [39]:
```
from sklearn.preprocessing import LabelEncoder,StandardScaler
```

In [40]:
```
#Standardize the value of x so that mean will 0 and SD will become 1 , and make the data as normal distributed
sc = StandardScaler()
sc.fit_transform(x)
x = pd.DataFrame(x,columns=x.columns)
```

In [41]:
```
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
from sklearn.linear_model import LogisticRegression,Lasso,LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor,GradientBoostingRegressor
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.decomposition import PCA
from scipy.stats import zscore
from sklearn.model_selection import cross_val_score
```

In [42]:
```
from sklearn.preprocessing import StandardScaler
```

In [43]:
```
scalar=StandardScaler()
x_scaled=scalar.fit_transform(x)
x_scaled
```

Out[43]:
```
array([[ 0.26996045,  0.2193973 , -0.77690976, ...,  3.50043557,
         3.14776031,  0.6409707 ],
       [ 1.53487317,  1.40998014,  0.93094037, ..., -0.83974649,
        -0.90571444,  1.61029666],
       [-0.1426069 , -0.17147058, -0.21286805, ..., -0.83974649,
        -0.90571444,  2.02572207],
       ...,
       [ 1.50548344,  1.38811669,  1.51883972, ...,  0.77215666,
         0.56801163,  1.05639611],
       [ 1.57295814,  1.45397421, -0.57395603, ..., -0.83974649,
         1.53334915,  0.87176259],
       [ 0.54058534,  0.47985924, -0.51055785, ..., -0.83974649,
```

So we have divided data into two variables now we will fit our data into various model and among that those who will give good accuracy we will select that model so lets move further

Model Building

now we will building model

In [44]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

In [45]:
```
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.24,random_state=40)
```

In [46]:
```
log_reg=LogisticRegression()
log_reg.fit(x_train, y_train)
preddt=log_reg.predict(x_test)
print(accuracy_score(y_test,preddt))
print(confusion_matrix(y_test,preddt))
print(classification_report(y_test,preddt))
```

```
0.8673827214988107
[[  848  4897]
 [  511 34523]]
              precision    recall  f1-score   support

           0       0.62      0.15      0.24      5745
           1       0.88      0.99      0.93     35034

    accuracy                           0.87     40779
   macro avg       0.75      0.57      0.58     40779
weighted avg       0.84      0.87      0.83     40779
```

we got 86.73% accuracy from logistic regression mode

In [47]:
```
dt=DecisionTreeClassifier()
dt.fit(x_train, y_train)
preddt=dt.predict(x_test)
print(accuracy_score(y_test,preddt))
print(confusion_matrix(y_test,preddt))
print(classification_report(y_test,preddt))
```

```
0.8792025307143383
[[ 3369  2376]
 [ 2550 32484]]
              precision    recall  f1-score   support

           0       0.57      0.59      0.58      5745
           1       0.93      0.93      0.93     35034

    accuracy                           0.88     40779
   macro avg       0.75      0.76      0.75     40779
weighted avg       0.88      0.88      0.88     40779
```

we got 87.93% accuracy from DecisionTreeClassifier model

In [48]:
```
rf=RandomForestClassifier()
rf.fit(x_train, y_train)
preddt=rf.predict(x_test)
print(accuracy_score(y_test,preddt))
print(confusion_matrix(y_test,preddt))
print(classification_report(y_test,preddt))
```

```
0.91527501900488
[[ 3207  2538]
 [  917 34117]]
              precision    recall  f1-score   support

           0       0.78      0.56      0.65      5745
           1       0.93      0.97      0.95     35034

    accuracy                           0.92     40779
   macro avg       0.85      0.77      0.80     40779
weighted avg       0.91      0.92      0.91     40779
```

In [49]:
```
kn=KNeighborsClassifier()
kn.fit(x_train, y_train)
preddt=kn.predict(x_test)
print(accuracy_score(y_test,preddt))
print(confusion_matrix(y_test,preddt))
print(classification_report(y_test,preddt))
```

```
0.8970793790921798
[[ 2895  2850]
 [ 1347 33687]]
              precision    recall  f1-score   support

           0       0.68      0.50      0.58      5745
           1       0.92      0.96      0.94     35034

    accuracy                           0.90     40779
   macro avg       0.80      0.73      0.76     40779
weighted avg       0.89      0.90      0.89     40779
```

we got 89.70 % accuracy from KNeighborsclassifier

so from above four model we got maximum accuracy with randomforestclassifier i.e 91.39% we can also check it with the AUC-ROC Curve

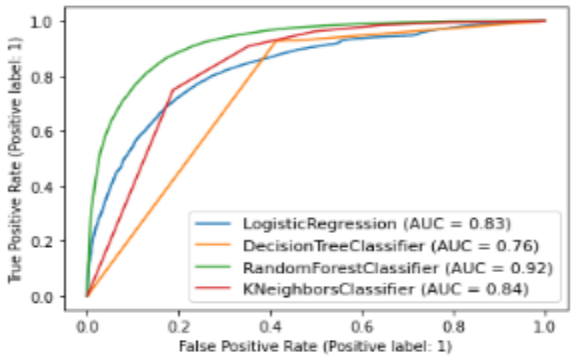
So among the four model we got maximum accuracy gor the RandomForestClassifier 91.39%,we can also use AUC-ROC curve for confirmation of model selection

14) AUC-ROC Curve:

AUC-ROC curve

In [50]:
```
from sklearn.metrics import roc_curve,roc_auc_score,plot_roc_curve
```

In [51]:
```
disp=plot_roc_curve(log_reg,x_test,y_test)
disp=plot_roc_curve(dt,x_test,y_test,ax=disp.ax_)
disp=plot_roc_curve(rf,x_test,y_test,ax=disp.ax_)
disp=plot_roc_curve(kn,x_test,y_test,ax=disp.ax_)
plt.legend(prop={'size':11},loc='lower right')
plt.show()
```



so from AUC-ROC curve it is seen that maximum accuracy is for RandomForestClassifier i.e 92%

so lets will select RandomForestClassifier

So from this also we got maximum accuracy for the RandomForestClassifier

15) Cross-Validation:

Now we will the cross validation of our model

cross validation

In [54]:
```
from sklearn.model_selection import cross_val_score
```

In [55]:
```
for i in range(2,10):
    cv=cross_val_score(rf,x,y,cv=i)
    print(rf,cv.mean())
```

```
RandomForestClassifier() 0.9137195389823827
RandomForestClassifier() 0.9146729758520639
RandomForestClassifier() 0.9150261030692938
RandomForestClassifier() 0.9144846459311104
RandomForestClassifier() 0.9148554299991561
RandomForestClassifier() 0.9153321444756372
RandomForestClassifier() 0.9148554291046272
RandomForestClassifier() 0.9150378727686849
```

we can predict the output as

In [56]:
```
#Lets plot and visualize
y_pred=rf.predict(x_test)
y_pred
```

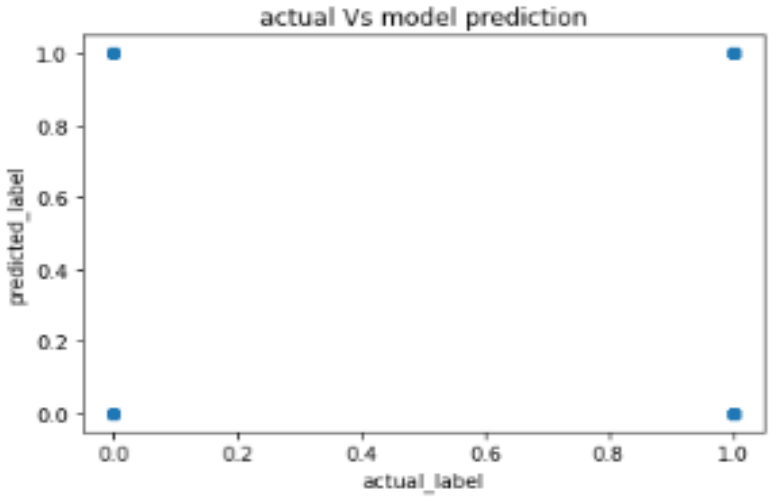
Out[56]:
```
array([1, 1, 1, ..., 1, 1, 0], dtype=int64)
```

So we have also cross validated our data

16) Actual vs predicted output:

Now we plot a actual verses predicted output of dataset

```
In [57]: plt.scatter(y_test,y_pred)
         plt.xlabel('actual_label')
         plt.ylabel('predicted_label')
         plt.title('actual Vs model prediction')
         plt.show()
```



this is the visualization of actual vs predicted label values

17) saving model:

So we have selected RandomForestClassifier model ,so will save it for future data prediction

---

saving model

```
In [58]: #saving model
         import joblib
         joblib.dump(rf,'Micro_credit_defaulter_project')

Out[58]: ['Micro_credit_defaulter_project']
```

18) conclusion:

conclusion

In [59]:
```
loaded_model=joblib.load('Micro_credit_defaulter_project')
result=loaded_model.score(x_test,y_test)
print(result)
```

```
0.9141715098457539
```

so in this we have devoloped model ,saved it and also drawn a conclusion with model accuracy score of 91.41%

So in this way we have developed model by checking its accuracy through various algorithm among that we have selected best model also we have saved the model ,accuracy from saved model is drawn 91.41%.so now we can use this saved model for the future data prediction