MICRO-CREDIT-DEFAULTER

Submitted by:

TEJENDRA SONI

ACKNOWLEDGMENT

I would like to express my thanks of gratitude to SME SWATI MAHASETH as well as Flip Robo who gave me the golden opportunity to do this wonderful project on the topic Micro Credit Defaulter, which also helped me in doing a lot of research and I came to know about so many new things. I am really very thankful to them.

I am making this project to increase my knowledge.

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 the 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).

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.

Objective:

Building 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 paid i.e. Non- defaulter, while, Label '0' indicates that the loan has not been paid i.e. defaulter.

Analytical Problem Framing

Firstly, we will start by importing required libraries and databases.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

import joblib
import warnings
warnings.filterwarnings('ignore')
```

```
data=pd.read_csv("micro credit data file.csv")
data
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	mec
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	
209588	209589	1	22758I85348	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	...	6.0	
209589	209590	1	95583I84455	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	...	6.0	
209590	209591	1	28556I85350	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	0.0	...	12.0	
209591	209592	1	59712I82733	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	38.0	...	12.0	
209592	209593	1	65061I85339	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	0.0	...	12.0	

209593 rows × 37 columns

Here is the list of all columns and size of dataset:

```
data.columns
```

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

```
data.shape
```

```
(209593, 37)
```

Let’s check the datatype of all columns:

```
data.dtypes
```

```
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
```

Other than msisdn, pcircle and pcircle all other columns are numerical i.e., Float and integer.

Also we can drop column Unnamed: 0 as it contains serial number only, mobile number and pdate is also not required for further analysis.

```
data.drop(["Unnamed: 0",'msisdn','pdate'], axis=1, inplace=True)
```

Then with the help of describe we will take a glimpse of data:

```
data.describe(include='all').T
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
label	209593	NaN	NaN	NaN	0.875177	0.330519	0	1	1	1	1
aon	209593	NaN	NaN	NaN	8112.34	75696.1	-48	246	527	982	999861
daily_decr30	209593	NaN	NaN	NaN	5381.4	9220.62	-93.0127	42.44	1469.18	7244	265926
daily_decr90	209593	NaN	NaN	NaN	6082.52	10918.8	-93.0127	42.692	1500	7802.79	320630
rental30	209593	NaN	NaN	NaN	2692.58	4308.59	-23737.1	280.42	1083.57	3356.94	198926
rental90	209593	NaN	NaN	NaN	3483.41	5770.46	-24720.6	300.26	1334	4201.79	200148
last_rech_date_ma	209593	NaN	NaN	NaN	3755.85	53905.9	-29	1	3	7	998650
last_rech_date_da	209593	NaN	NaN	NaN	3712.2	53374.8	-29	0	0	0	999172
last_rech_amt_ma	209593	NaN	NaN	NaN	2064.45	2370.79	0	770	1539	2309	55000
cnt_ma_rech30	209593	NaN	NaN	NaN	3.97806	4.25609	0	1	3	5	203
fr_ma_rech30	209593	NaN	NaN	NaN	3737.36	53643.6	0	0	2	6	999606
sumamnt_ma_rech30	209593	NaN	NaN	NaN	7704.5	10139.6	0	1540	4628	10010	810096
medianamnt_ma_rech30	209593	NaN	NaN	NaN	1812.82	2070.86	0	770	1539	1924	55000
medianmarechprebal30	209593	NaN	NaN	NaN	3851.93	54006.4	-200	11	33.9	83	999479
cnt_ma_rech90	209593	NaN	NaN	NaN	6.31543	7.19347	0	2	4	8	336
fr_ma_rech90	209593	NaN	NaN	NaN	7.71678	12.5903	0	0	2	8	88
sumamnt_ma_rech90	209593	NaN	NaN	NaN	12396.2	16857.8	0	2317	7226	16000	953036
medianamnt_ma_rech90	209593	NaN	NaN	NaN	1864.6	2081.68	0	773	1539	1924	55000
medianmarechprebal90	209593	NaN	NaN	NaN	92.0255	369.216	-200	14.6	36	79.31	41456.5
cnt_da_rech30	209593	NaN	NaN	NaN	262.578	4183.9	0	0	0	0	99914.4
fr_da_rech30	209593	NaN	NaN	NaN	3749.49	53885.4	0	0	0	0	999809
cnt_da_rech90	209593	NaN	NaN	NaN	0.0414947	0.397556	0	0	0	0	38
fr_da_rech90	209593	NaN	NaN	NaN	0.0457124	0.951386	0	0	0	0	64
cnt_loans30	209593	NaN	NaN	NaN	2.75898	2.5545	0	1	2	4	50
amnt_loans30	209593	NaN	NaN	NaN	17.952	17.3797	0	6	12	24	306
maxamnt_loans30	209593	NaN	NaN	NaN	274.659	4245.26	0	6	6	6	99864.6
medianamnt_loans30	209593	NaN	NaN	NaN	0.0540285	0.218039	0	0	0	0	3
cnt_loans90	209593	NaN	NaN	NaN	18.5209	224.797	0	1	2	5	4997.52
amnt_loans90	209593	NaN	NaN	NaN	23.6454	26.4699	0	6	12	30	438
maxamnt_loans90	209593	NaN	NaN	NaN	6.70313	2.10386	0	6	6	6	12
medianamnt_loans90	209593	NaN	NaN	NaN	0.0460774	0.200692	0	0	0	0	3
payback30	209593	NaN	NaN	NaN	3.39883	8.81373	0	0	0	3.75	171.5
payback90	209593	NaN	NaN	NaN	4.32149	10.3081	0	0	1.66667	4.5	171.5
pcircle	209593	1	UPW	209593	NaN	NaN	NaN	NaN	NaN	NaN	NaN

We can see we have only one unique value in pcircle hence we can drop that column.

```
data.drop("pcircle", axis=1, inplace=True)
```

We can see count of all columns is same still, we will take a look at null values.

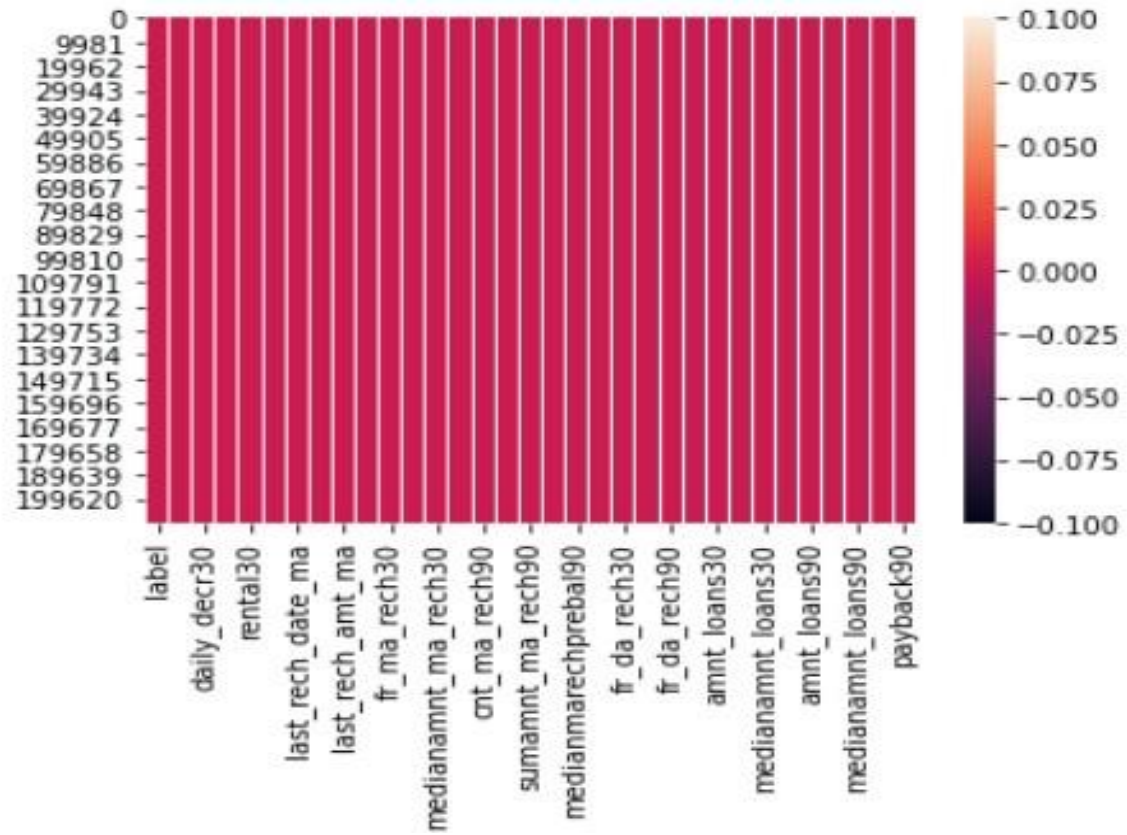
```
data.isnull().sum()
```

```
label                   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
cnt_loans30             0
amnt_loans30            0
maxamnt_loans30         0
medianamnt_loans30      0
cnt_loans90             0
amnt_loans90            0
maxamnt_loans90         0
medianamnt_loans90      0
payback30               0
payback90               0
dtype: int64
```

We don’t have null values in our database. Let's see the same thing with help of heatmap:

```
sns.heatmap(data.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a8d360dc40>
```



In our database we have some columns with data of 30 days as well as 90 days. Last 90 days data obviously include data of 30 days, Hence I am dropping columns with 30 days data.

```
data = data[data.columns.drop(list(data.filter(regex='30')))]
```

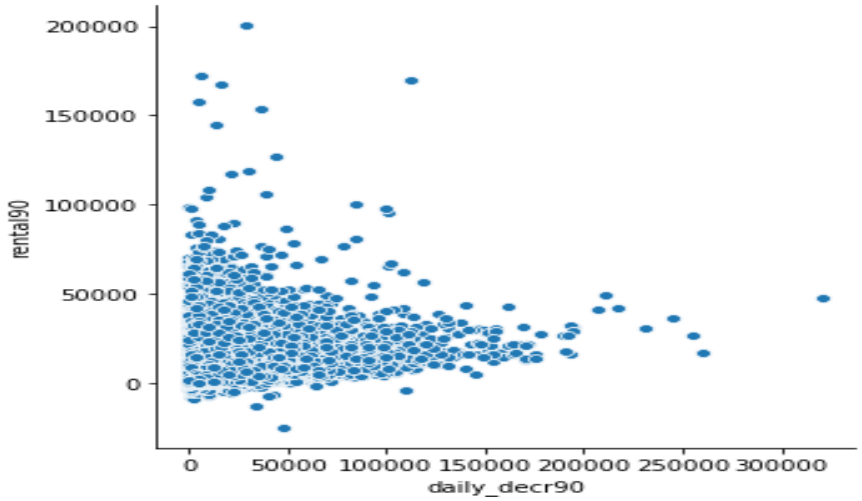
```
data.columns
```

```
Index(['label', 'aon', 'daily_decr90', 'rental90', 'last_rech_date_ma',
       'last_rech_date_da', 'last_rech_amt_ma', 'cnt_ma_rech90',
       'fr_ma_rech90', 'sumamnt_ma_rech90', 'medianamnt_ma_rech90',
       'medianmarechprebal90', 'cnt_da_rech90', 'fr_da_rech90', 'cnt_loans90',
       'amnt_loans90', 'maxamnt_loans90', 'medianamnt_loans90', 'payback90'],
      dtype='object')
```

Let's look at some **visualization** now:

```
sns.relplot(x='daily_decr90',y='rental90',data=data,kind='scatter')
```

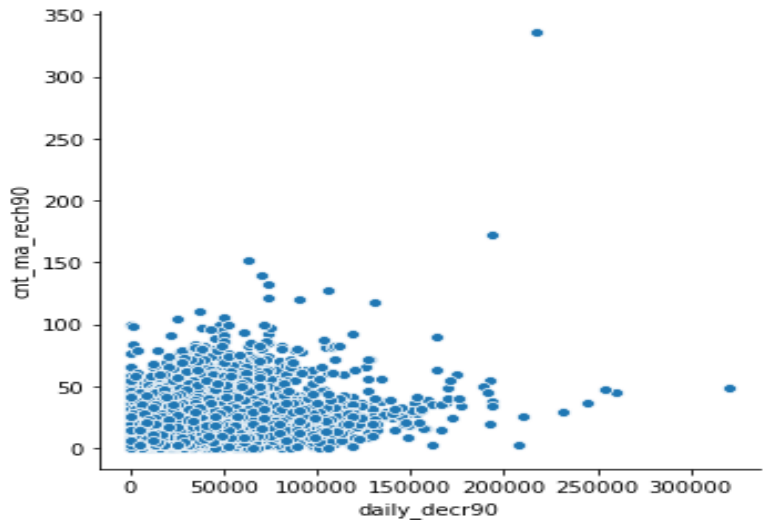
```
<seaborn.axisgrid.FacetGrid at 0x1fb0cf1b2b0>
```



We can see a positive relation between daily_decr90 and rental90. Maximum data for daily_decr90 is in range till 150000 and for rental90 its till 100000.

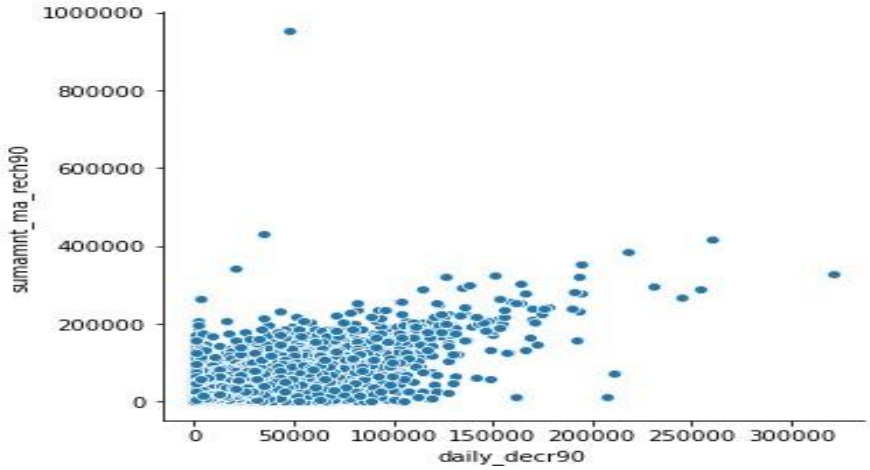
```
sns.relplot(x='daily_decr90',y='cnt_ma_rech90',data=data,kind='scatter')
```

```
<seaborn.axisgrid.FacetGrid at 0x1fb1441bf10>
```



The Relationship between daily_decr90 and cnt_ma_rech90 is also somewhat positive. Maximum amount for daily decr90 is ranging till 13000 and for cnt_ma_rech90 count is 100.

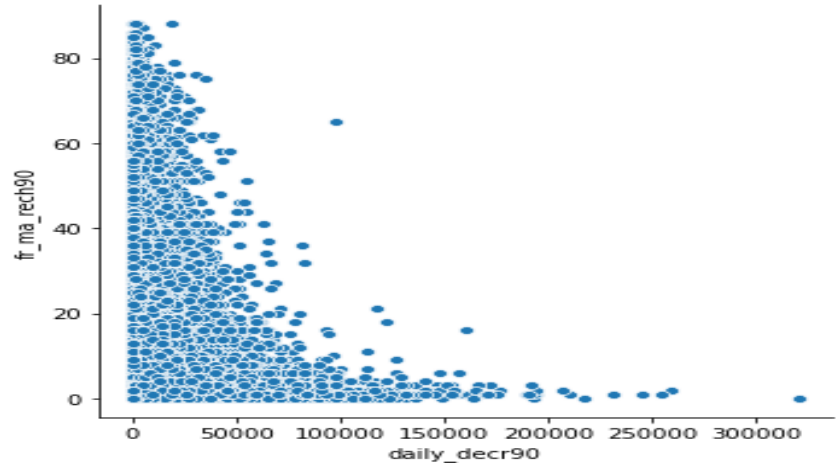
```
sns.relplot(x='daily_decr90',y='sumamnt_ma_rech90',data=data,kind='scatter')
plt.gcf().axes[0].yaxis.get_major_formatter().set_scientific(False)
```



The Relationship between daily_decr90 and sumamnt_ma_rech0 is also positive. Maximum amounts for daily decr are ranging till 15000 and for sumamnt_ma_rech30 it's till 20000.

```
sns.relplot(x='daily_decr90',y='fr_ma_rech90',data=data,kind='scatter')
```

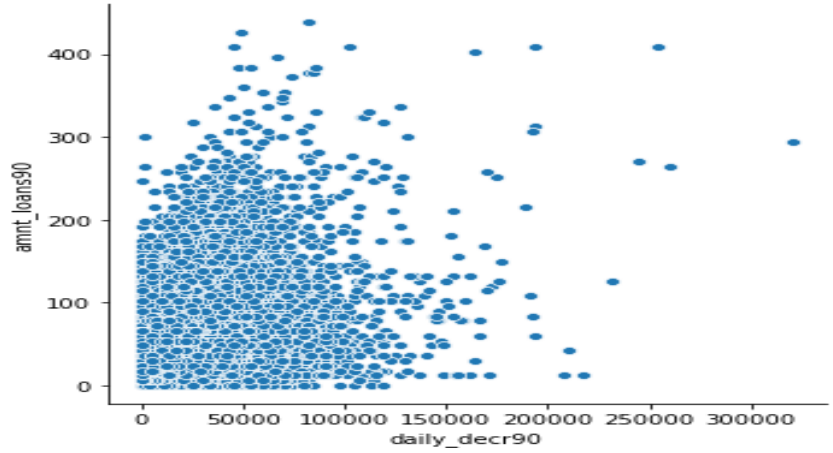
```
<seaborn.axisgrid.FacetGrid at 0x1fb0cf04ca0>
```



The Relationship between daily_decr90 and fr_ma_rech90 is also somewhat negative. Maximum amounts for daily decr90 is ranging till 150000 and for fr_ma_rech90 counts are spread over the entire range.

```
sns.relplot(x='daily_decr90',y='amnt_loans90',data=data,kind='scatter')
```

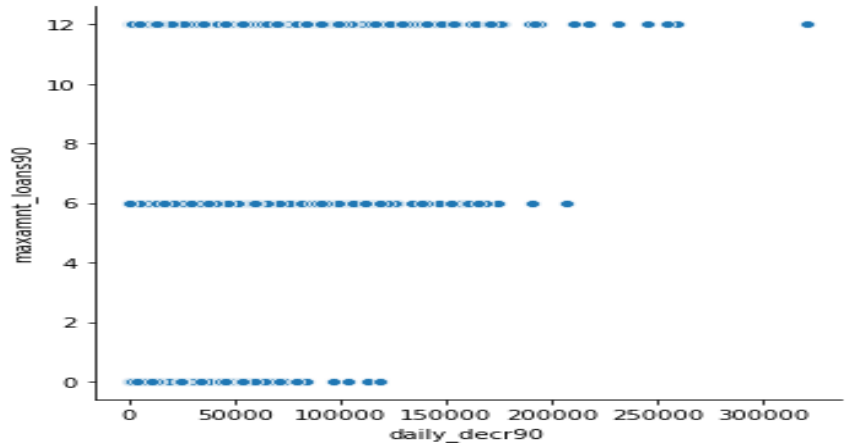
```
<seaborn.axisgrid.FacetGrid at 0x1fb12414eb0>
```



The Relationship between daily_decr90 and amnt_loans90 is also somewhat positive. Maximum amounts for daily decr90 is ranging till 15000 and for amnt_loans90 it’s till 300 counts.

```
sns.relplot(x='daily_decr90',y='maxamnt_loans90',data=data,kind='scatter')
```

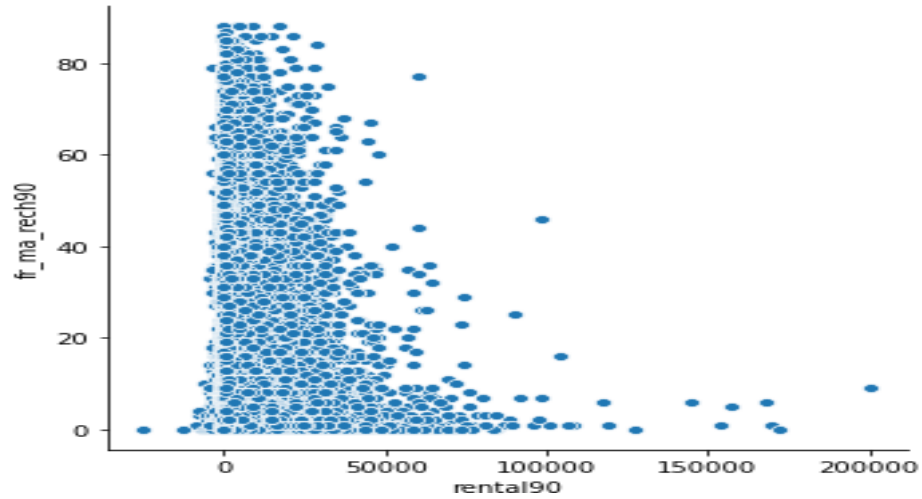
```
<seaborn.axisgrid.FacetGrid at 0x1fb123e18e0>
```



When pmonth is 6 then daily_decr90 ranging at 0, when pmonth is 7 then daily_decr90 is ranging till 180000 and when pmonth is 8 then daily_decr90 is ranging till 250000.

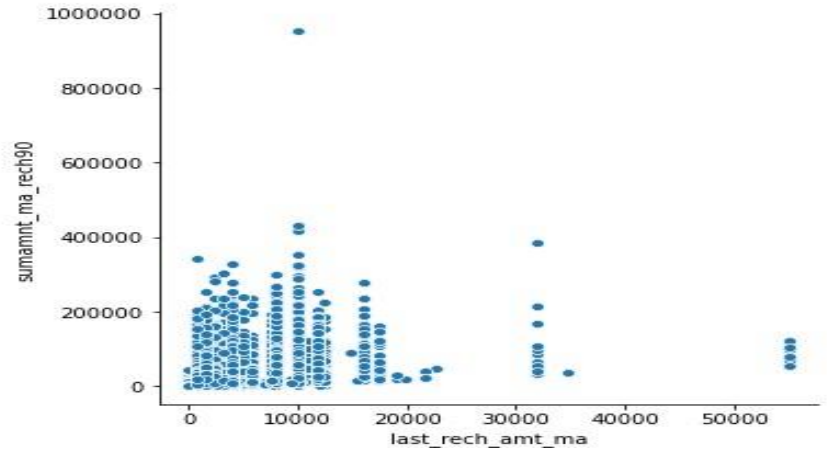
```
sns.relplot(x='rental90',y='fr_ma_rech90',data=data,kind='scatter')
```

```
<seaborn.axisgrid.FacetGrid at 0x1fb124890a0>
```



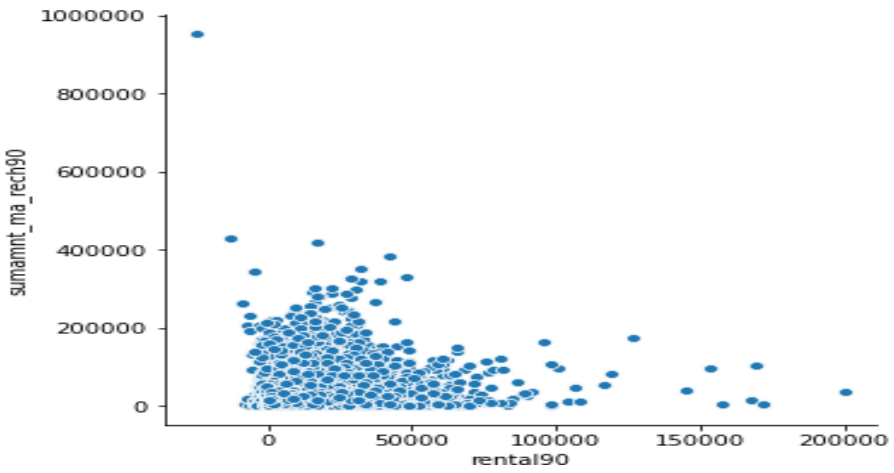
The Relationship between rental90 and fr_ma_rech90 is negative. Data for fr_ma_rech90 is spread over the entire range.

```
sns.relplot(x='last_rech_amt_ma',y='sumamnt_ma_rech90',data=data,kind='scatter')
plt.gcf().axes[0].yaxis.get_major_formatter().set_scientific(False)
```



The Relationship between last_rech_amt_ma and last_rech_amt_ma is positive. Maximum amount of data for last_rech_amt_ma is ranging till 20000

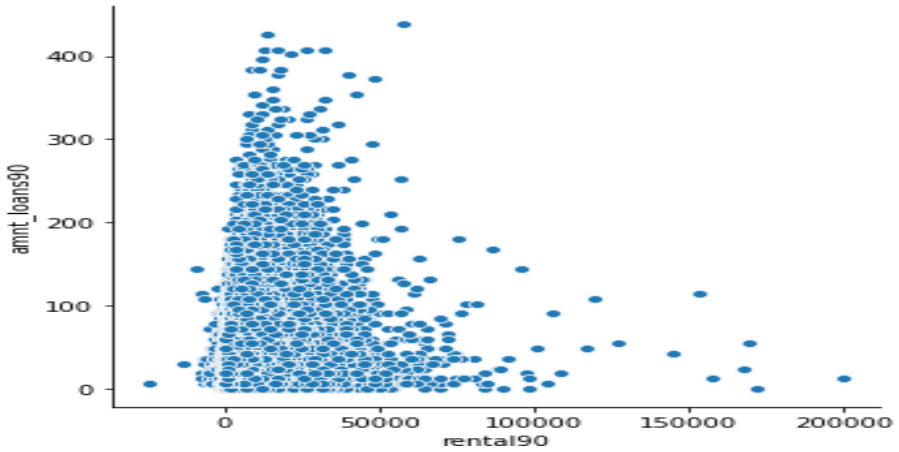
```
sns.relplot(x='rental90',y='sumamnt_ma_rech90',data=data,kind='scatter')
plt.gcf().axes[0].yaxis.get_major_formatter().set_scientific(False)
```



For rental90 data is maximum number of data is ranging till 80000 and sumamnt_ma_rech30 its till 300000

```
sns.relplot(x='rental90',y='amnt_loans90',data=data,kind='scatter')
```

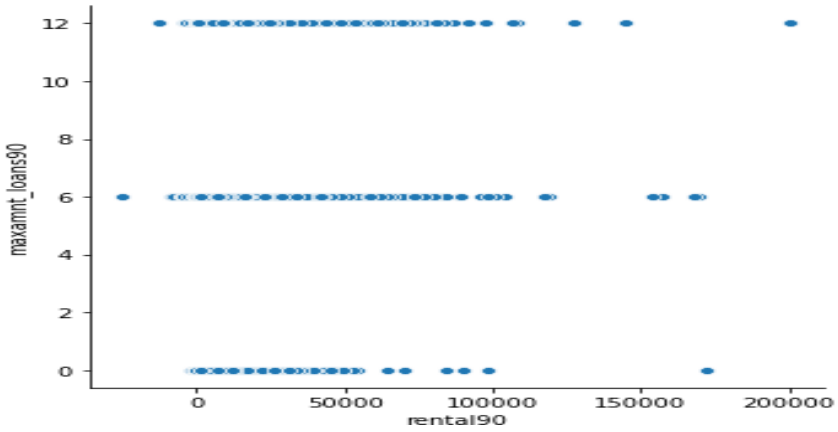
```
<seaborn.axisgrid.FacetGrid at 0x1fb1256e0a0>
```



For amnt_loans90 data is spread over the entire range and for rental90 maximum data is ranging till 80000.

```
sns.relplot(x='rental90',y='maxamnt_loans90',data=data,kind='scatter')
```

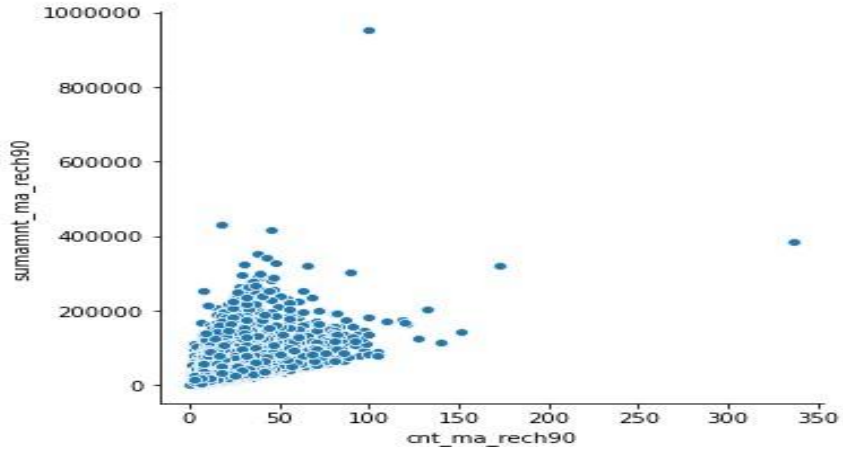
```
<seaborn.axisgrid.FacetGrid at 0x1fb125e8d90>
```



When maxamnt_loans90 is 0 then maximum amount of rental90 is ranging till 100000, When maxamnt_loans90 is 6 then maximum amount rental90 is ranging till 120000 and When

maxamnt_loans90 is 12 then maximum amount of rental90 is ranging till 120000.

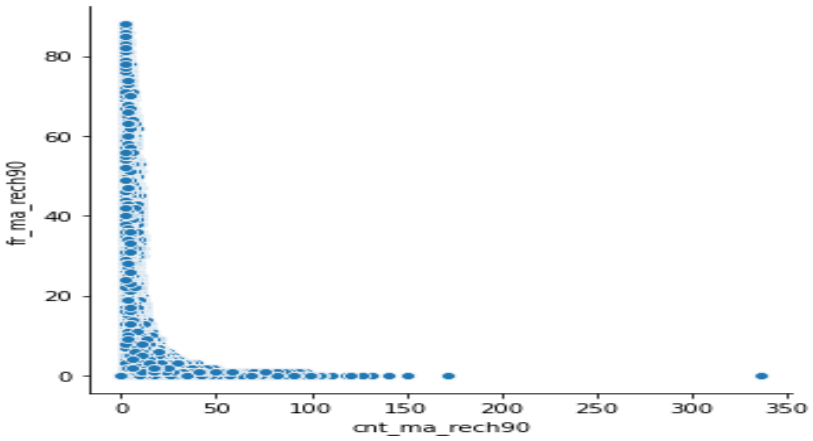
```
sns.relplot(x='cnt_ma_rech90',y='sumamnt_ma_rech90',data=data,kind='scatter')
plt.gcf().axes[0].yaxis.get_major_formatter().set_scientific(False)
```



For cnt_ma_rech90 maximum data is ranging till 100 and for sumamnt_ma_rech90 maximum data is ranging till 300000. We can a see positive relation between cnt_ma_rech90 and sumamnt_ma_rech90.

```
sns.relplot(x='cnt_ma_rech90',y='fr_ma_rech90',data=data,kind='scatter')
```

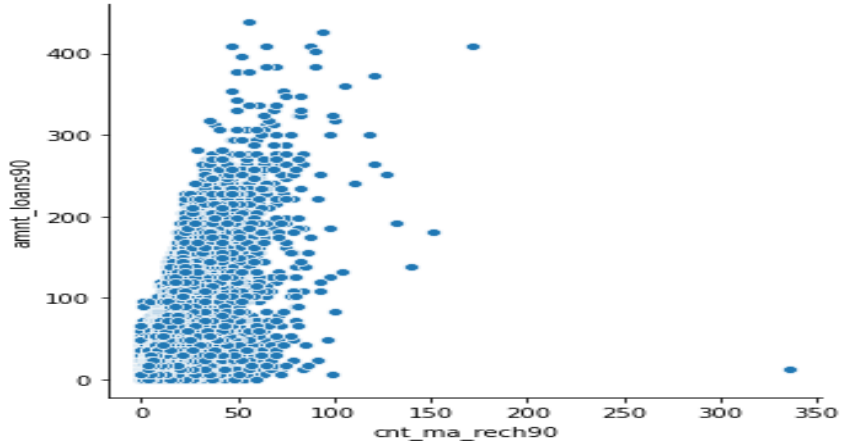
```
<seaborn.axisgrid.FacetGrid at 0x1fb12430670>
```



For fr_ma_rech90 data is spread over the entire range and for cnt_ma_rech90 maximum data is ranging till 150. The Relationship between cnt_ma_rech90 and fr_ma_rech90 is somewhat negative.

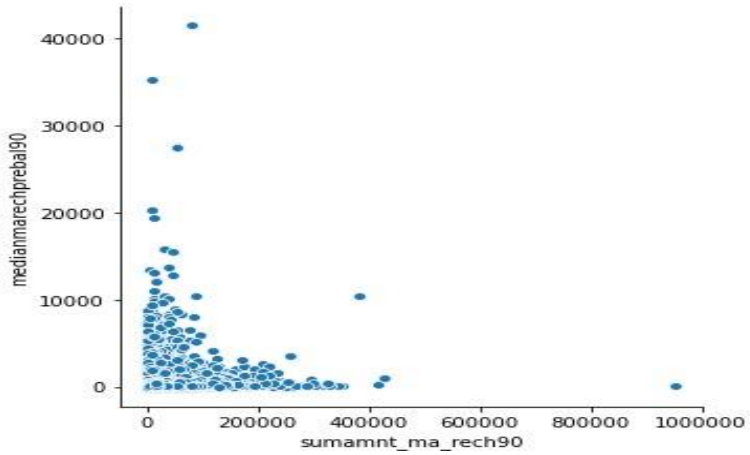
```
sns.relplot(x='cnt_ma_rech90',y='amnt_loans90',data=data,kind='scatter')
```

```
<seaborn.axisgrid.FacetGrid at 0x1fb24ebab80>
```



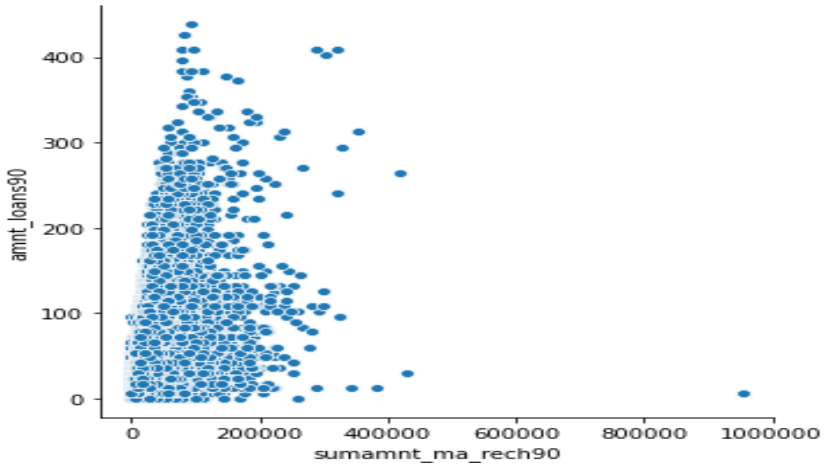
For amnt_loans90, data is spread over the entire range and for amnt_loans90 maximum data is till range 100.

```
sns.relplot(x='sumamnt_ma_rech90',y='medianmarechprebal90',data=data,kind='scatter')
plt.gcf().axes[0].xaxis.get_major_formatter().set_scientific(False)
```



For sumamnt_ma_rech90 maximum data is ranging till 380000 and for medianmarechprebal90 maximum data is ranging till 15000.

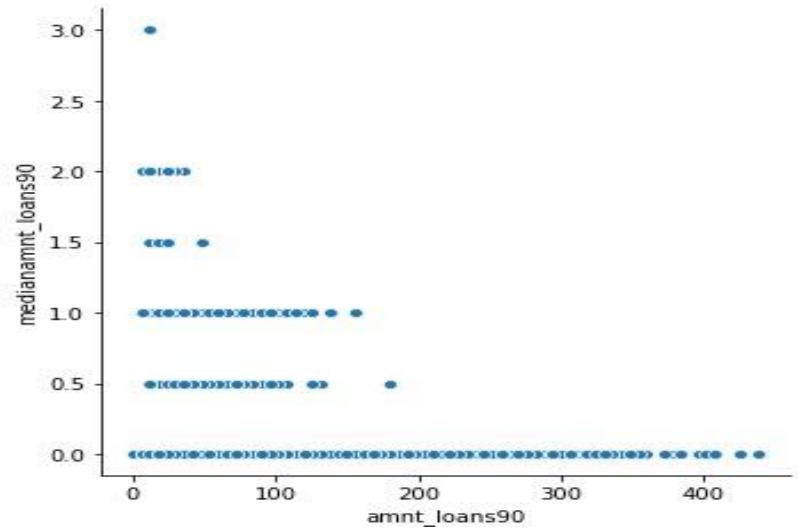
```
sns.relplot(x='sumamnt_ma_rech90',y='amnt_loans90',data=data,kind='scatter')
plt.gcf().axes[0].xaxis.get_major_formatter().set_scientific(False)
```



For sumamnt_ma_rech90 maximum data is ranging till 300000 and for amnt_loans90 data is spread over the entire range.

```
sns.relplot(x='amnt_loans90',y='medianamnt_loans90',data=data,kind='scatter')
```

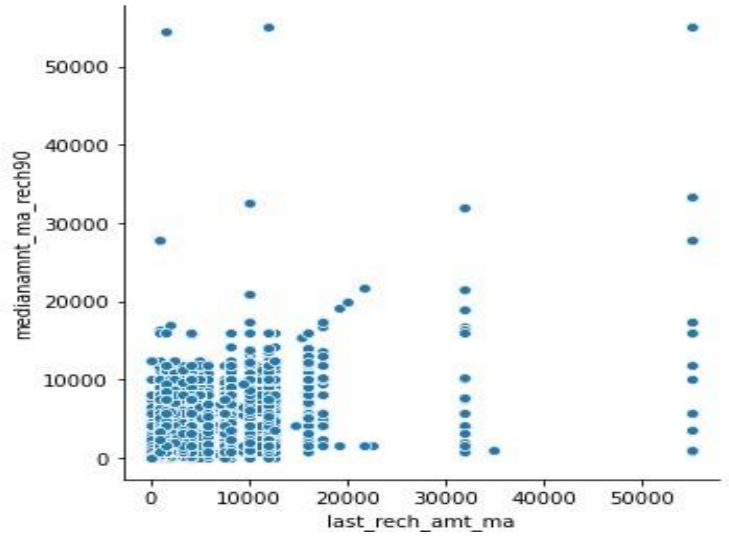
```
<seaborn.axisgrid.FacetGrid at 0x1fb585ee400>
```



When medianamnt_loans90 is 0 then amnt_loans90 is spread over the entire range, When medianamnt_loans90 is 0.5 then amnt_loans90 is ranging till 130, When medianamnt_loans90 is 1.0 then amnt_loans90 is ranging till 180, When medianamnt_loans90 is 1.5 then amnt_loans90 is ranging till 50, When medianamnt_loans90 is 2.0 then amnt_loans90 is ranging till 50, and When medianamnt_loans90 is 3.0 then amnt_loans90 is ranging till 10.

```
sns.relplot(x='last_rech_amt_ma',y='medianamnt_ma_rech90',data=data,kind='scatter')
```

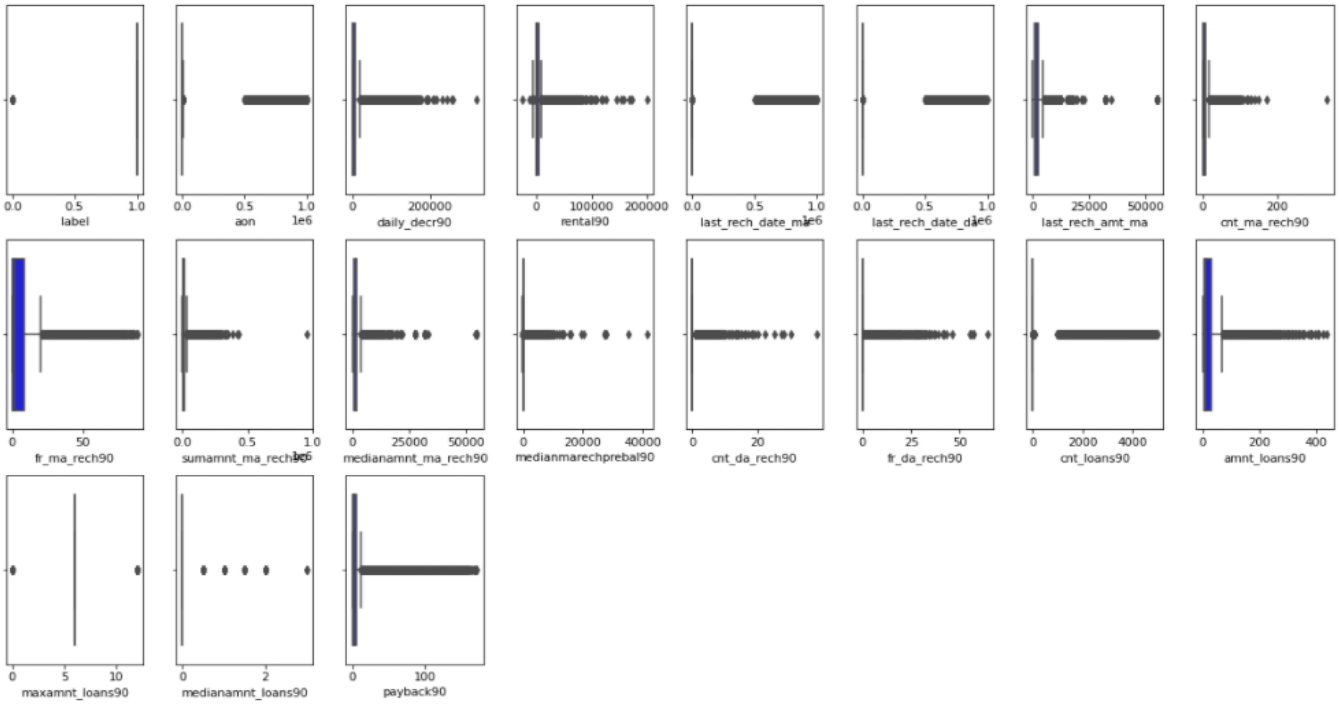
```
<seaborn.axisgrid.FacetGrid at 0x1fb045da880>
```



For last_rech_amt_ma maximum data till the range 20000 and medianamnt_ma_rech90 its till 20000

Let’s check outliers with the help of boxplot:

```
collist=data.columns.values
ncol=8
nrows=8
plt.figure(figsize=(2*ncol,3*ncol))
for i in range (0,len(collist)):
    if data.dtypes[i] != 'object':
        plt.subplot(nrows,ncol,i+1)
        sns.boxplot(data[collist[i]],color='Blue',orient='v')
        plt.tight_layout()
```



We can see outliers in many columns, Let's check individually and then work on the same.

Some functions for further analysis:

```
#below function will to detect outliers with the help IQR when feature has less/low skewness

def outlier_IQR_l(data_frame, feature_name):
    IQR=data_frame[feature_name].quantile(0.75) - data_frame[feature_name].quantile(0.25)
    lower_limit=data_frame[feature_name].quantile(0.25) - (IQR*1.5)
    upper_limit=data_frame[feature_name].quantile(0.75) + (IQR*1.5)
    return(lower_limit,upper_limit)
```

```
#below function will to detect outliers with the help IQR when feature has high skewness

def outlier_IQR_h(data_frame, feature_name):
    IQR=data_frame[feature_name].quantile(0.75) - data_frame[feature_name].quantile(0.25)
    lower_limit=data_frame[feature_name].quantile(0.25) - (IQR*3)
    upper_limit=data_frame[feature_name].quantile(0.75) + (IQR*3)
    return(lower_limit,upper_limit)
```

```
#below function will to detect outliers with the help of Zscore when feature data is distributed near to normal curve

def normaloutlier(data_frame, feature_name):
    lower_limit=data_frame[feature_name].mean() - 3 * data_frame[feature_name].std()
    upper_bridge=data_frame[feature_name].mean() + 3 * data_frame[feature_name].std()
    return(lower_limit,upper_limit)
```

```
# below function will help us to check skewness and distribution

def distribution(data_frame,feature_name):
    plt.figure(figsize=(10,5))
    plt.subplot(1,2,1)
    stats.probplot(data_frame[feature_name],dist='norm',plot=pylab)
    plt.subplot(1,2,2)
    data_frame[feature_name].hist()
    plt.title("Distribution")
```

Functions for transformation:

```
#Log transformation

def log_transform(data_frame,feature_name):
    data_frame_copy=data_frame.copy()
    if 0 in data_frame_copy[feature_name].unique():
        pass
    else:
        data_frame_copy[feature_name]=np.log(data_frame_copy[feature_name])
        plt.figure(figsize=(10,5))
        plt.subplot(1,2,1)
        stats.probplot(data_frame_copy[feature_name],dist='norm',plot=pylab)
        plt.subplot(1,2,2)
        data_frame_copy[feature_name].hist()
        plt.title('log transformation')
```

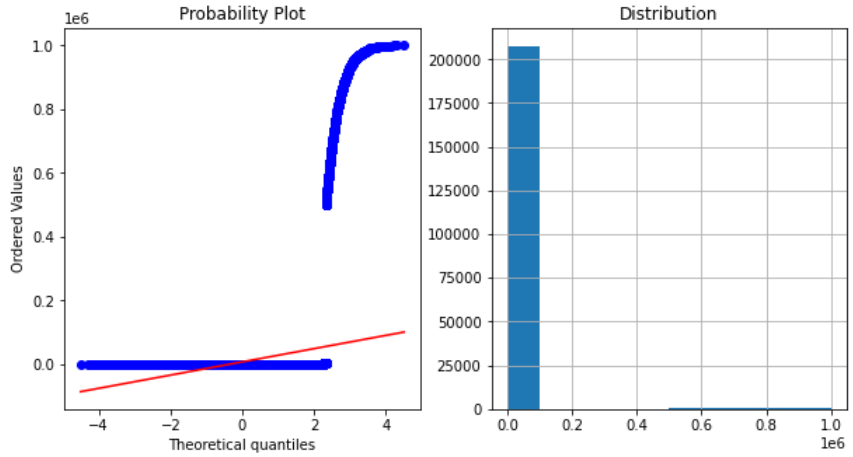
```
#Reciprocal transformation

def reciprocal_transform(data_frame,feature_name):
    data_frame_copy=data_frame.copy()
    if 0 in data_frame_copy[feature_name].unique():
        pass
    else:
        data_frame_copy[feature_name]=1/data_frame_copy[feature_name]
        plt.figure(figsize=(10,5))
        plt.subplot(1,2,1)
        stats.probplot(data_frame_copy[feature_name],dist='norm',plot=pylab)
        plt.subplot(1,2,2)
        data_frame_copy[feature_name].hist()
        plt.title('reciprocal transformation')
```

```
def sqaureroot_transform(data_frame.feature_name):
    data_frame_copy=data frame.copy()
    if 0  in data_frame_copy[feature_name].unique():
        pas  s

        data_frame_copy[feature_name]=data_frame_copy[feature_name]**(1/Z)
        pit.figure(figsize=(10,5))
        pit.subplot(1,Z,1)
        stats. probplot(data_frame_copy[feature_name].dist='norm',plot=pylab)
        pit.subplot(1,2,2)
        data_frame_copy[feature_name].hist()
        pit.title('Squareroot transformation')

#exponential transformation

def exponential transform(data frame,feature_name):
    data_frame_copy=data_frame.copy()
    if 0  in data_frame_copy[feature_name].unique():

    else:
                                        =a
        pit.figure(figsize=(£0,5))
        pit.subplot(1,2,1)
        stats. probplot(data_frame copy{feature names.dist='norm',plot=pylab)
        pit.subplot(1,2,2)
        data_frame_copy[feature_namej.hist()
        pit.title(' exponential transformation' )



def  boxcox transform(data_frame,feature name):
    data_frame_copy=data_frame.copy()
    #data_frame_copy = data_frame_copy[data_frame_copy >= 0]
    if 0  in data_frame_copy[feature_name].unique():

    else:

        p  .fibu  e  flg  i  e=(10,5))
                l
        stats. probplot(data_frame_copy $feature_name].dist='norm',plot=pylab)
        pit.subplot(1.Z,2)
        data_frame_copy[feature_name].list()
        pit.title('Boxcox transformation ' }
```

Let's Work on features individually:

aon:

```
distribution(data,"aon")
plt.gcf().axes[0].xaxis.get_major_formatter().set_scientific(False)
```

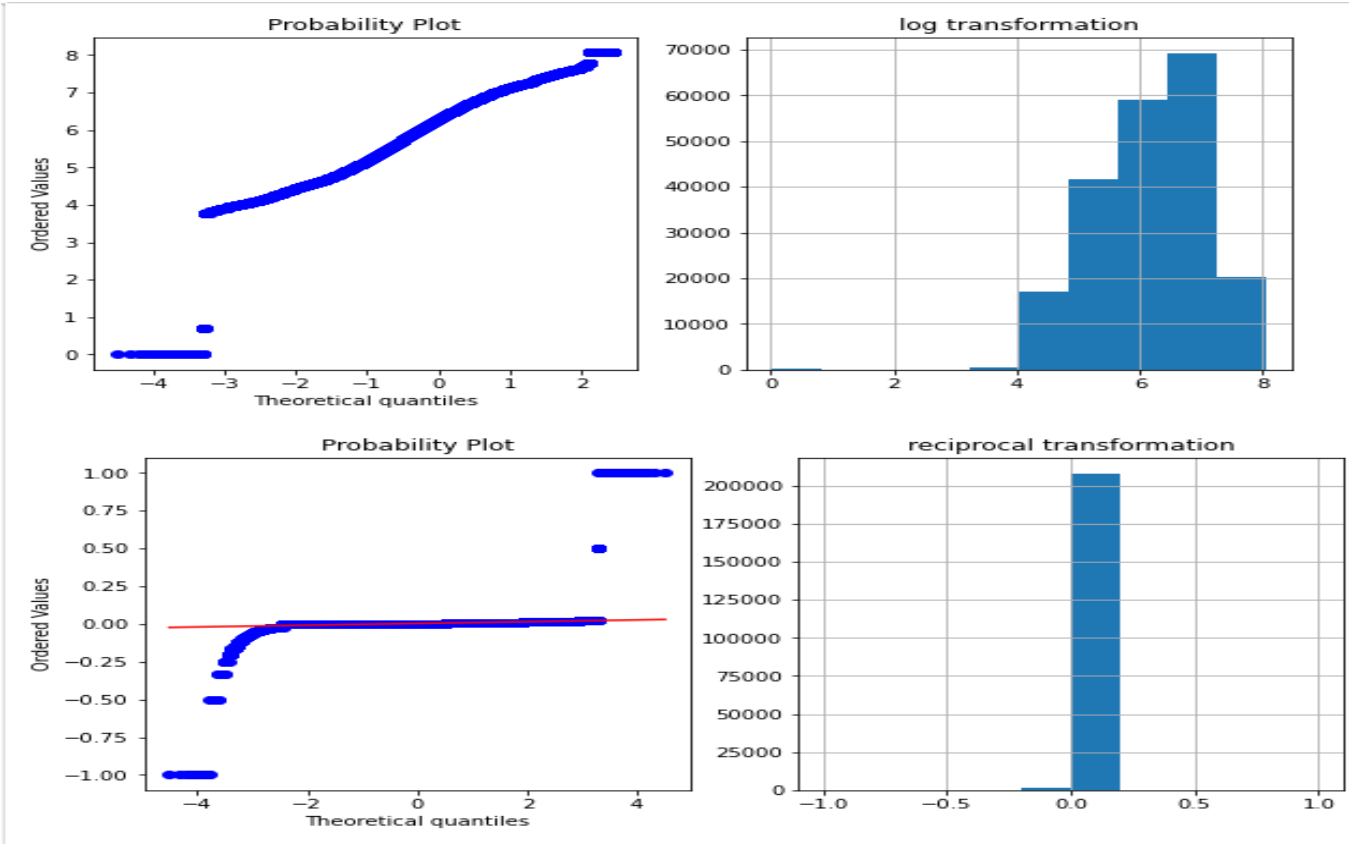


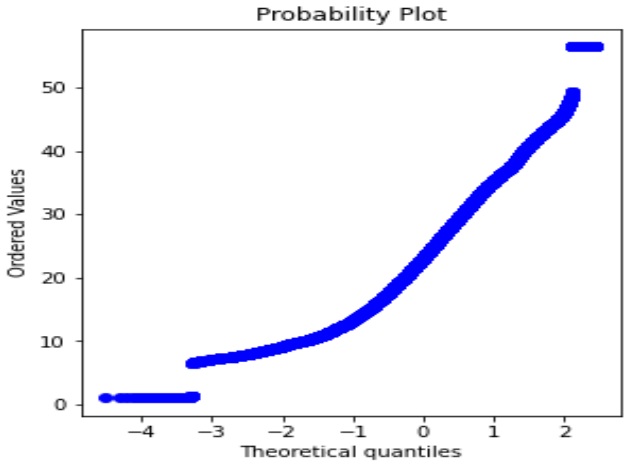

```
#We can see aon is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"aon")
```

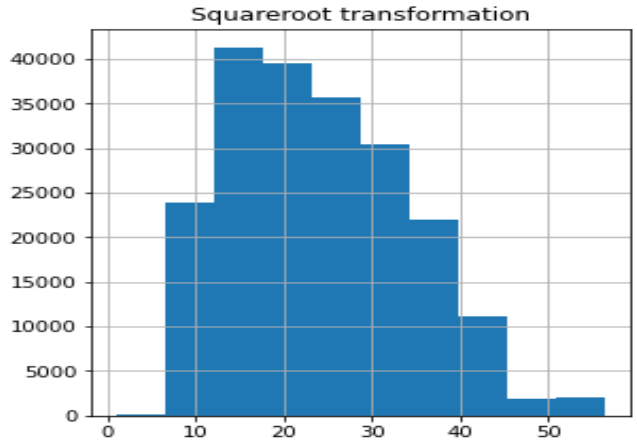
```
(-1962.0, 3190.0)
```

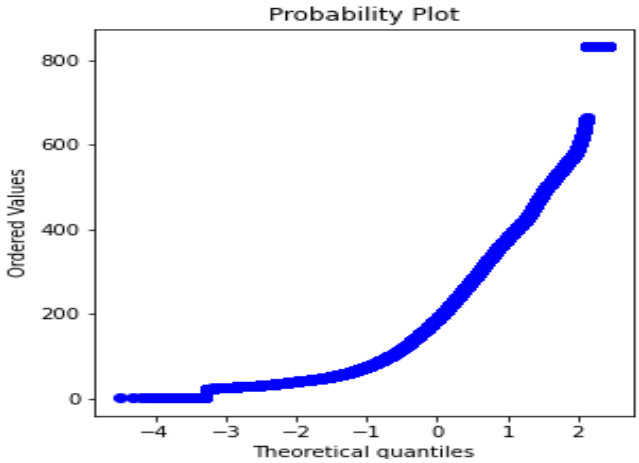
```
data.loc[data['aon']>= 3190.0,'aon']=3190.0
```

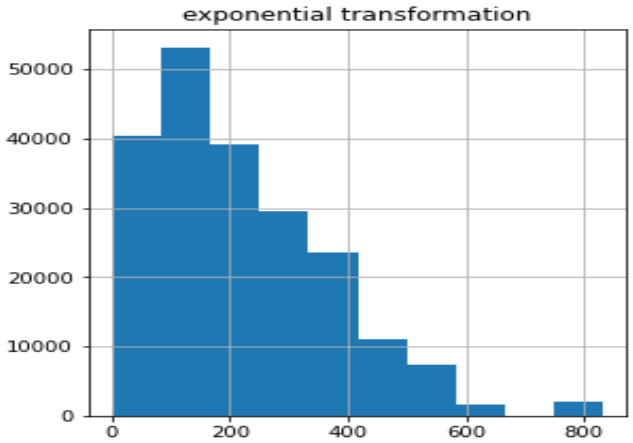
```
#Apply some transformation technique on aon for make distribution as normally distribution and check distribution by QQ plot

log_transform(data,"aon")
reciprocal_transform(data,"aon")
sqaureroot_transform(data,"aon")
exponential_transform(data,"aon")
for i in data.aon:
    if i > 0:
        break
    else:
        boxcox_transform(data,"aon")
```


Probability Plot
Ordered Values
Theoretical quantiles
log transformation
Probability Plot
Ordered Values
Theoretical quantiles
reciprocal transformation





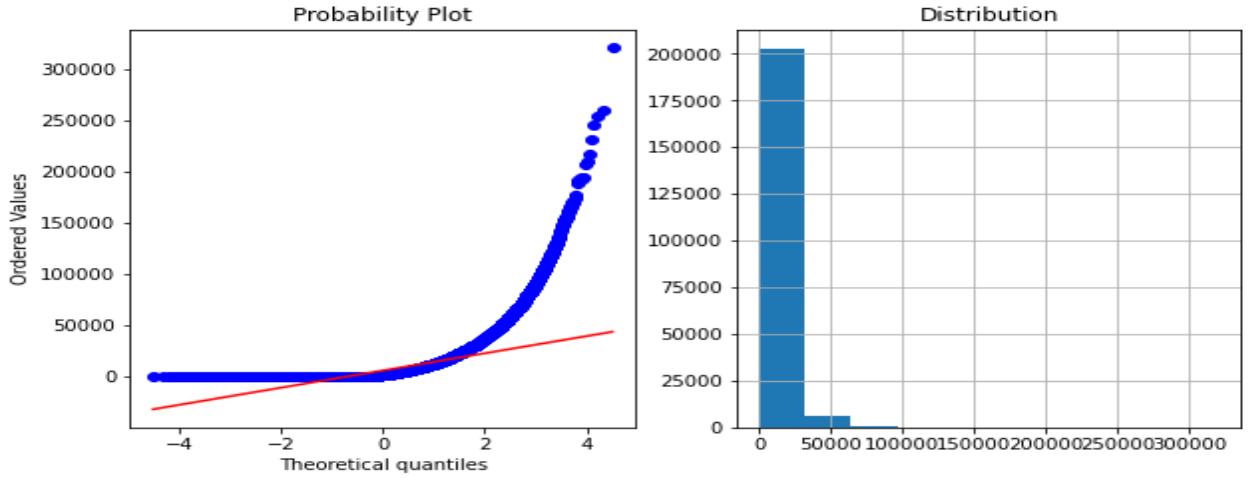





```
# Apply reciprocal transformation on aon feature
if 0 in data['aon'].unique():
    pass
else:
    data['aon']=1/data['aon']
```

daily_decr90:

```
distribution(data,"daily_decr90")
```



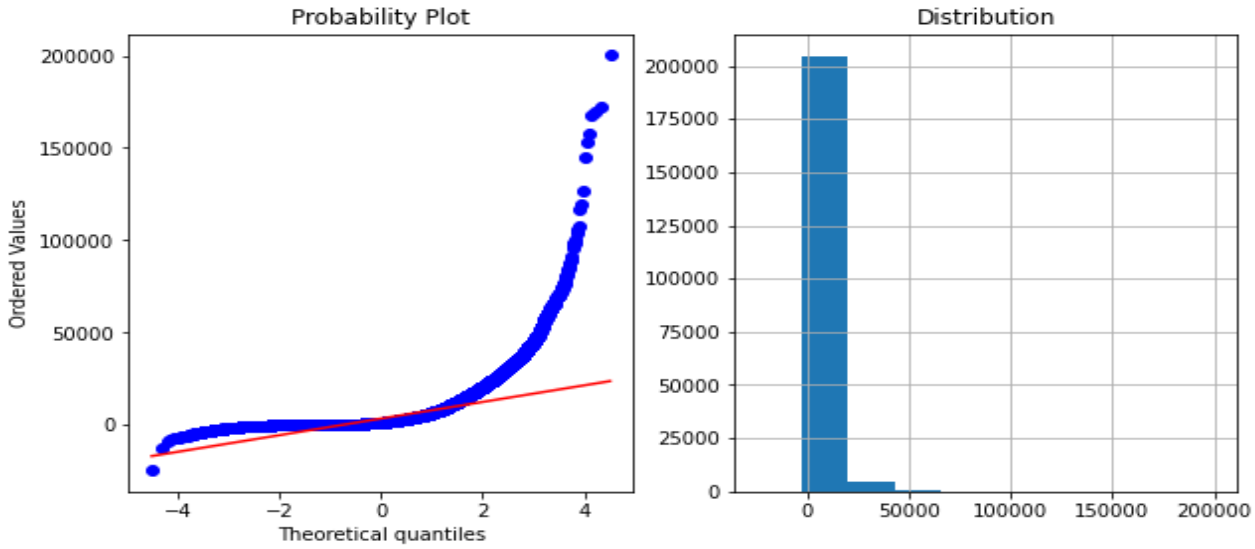

```
#We can see daily_decr90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"daily_decr90")
```

```
(-23237.60200000003, 31083.08400000004)
```

```
data.loc[data['daily_decr90']>= 31083.08400000004,'daily_decr90']=31083.08400000004
```

rental90:

```
distribution(data,"rental90")
```



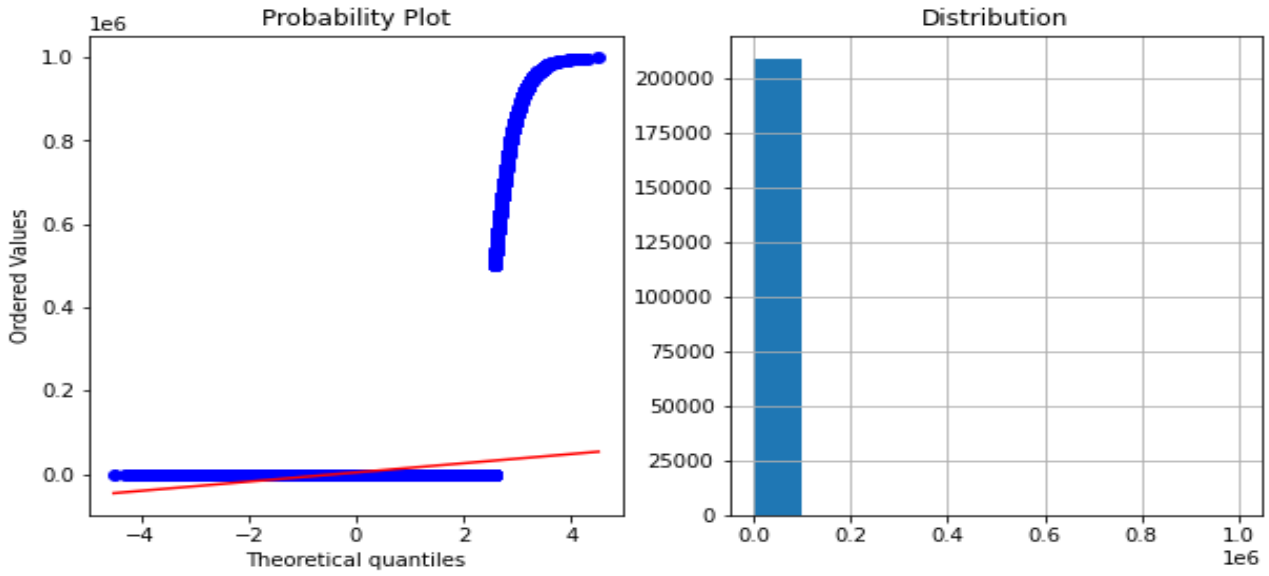

```
#We can see rental90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"rental90")
```

```
(-11404.33, 15906.380000000001)
```

```
data.loc[data['rental90']>= 15906.380000000001,'rental90']=15906.380000000001
```

last_rech_date_ma:

```
distribution(data,"last_rech_date_ma")
```



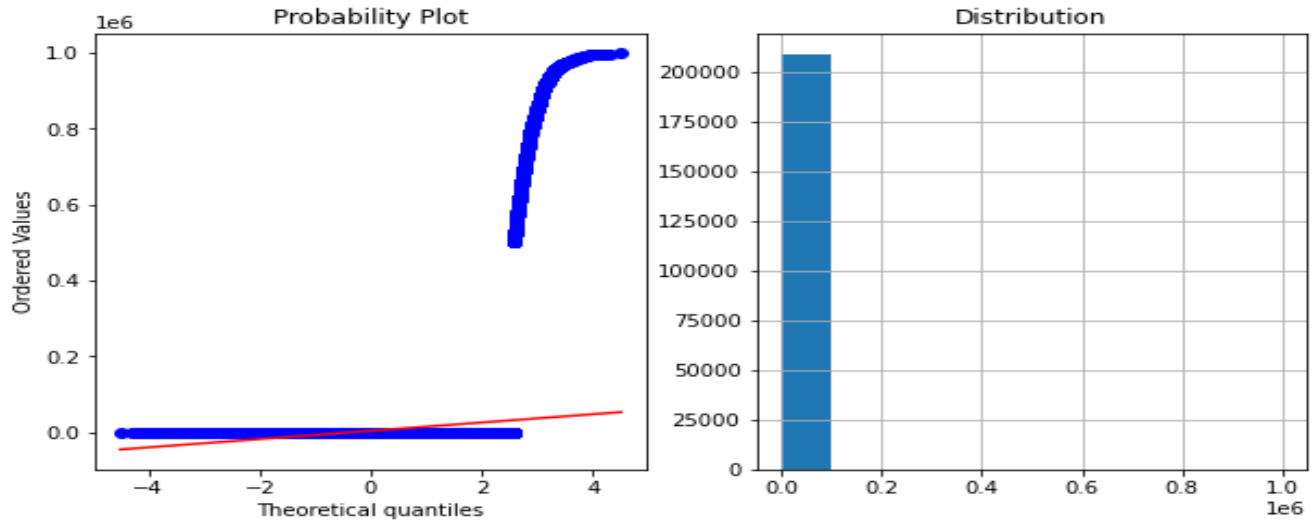

```
#We can see last_rech_date_ma is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"last_rech_date_ma")
```

```
(-17.0, 25.0)
```

```
data.loc[data['last_rech_date_ma']>= 25.0,'last_rech_date_ma']=25.0
```

last_rech_date_da:

```
distribution(data,"last_rech_date_da")
```




```
#We can see last_rech_date_da is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"last_rech_date_da")
```

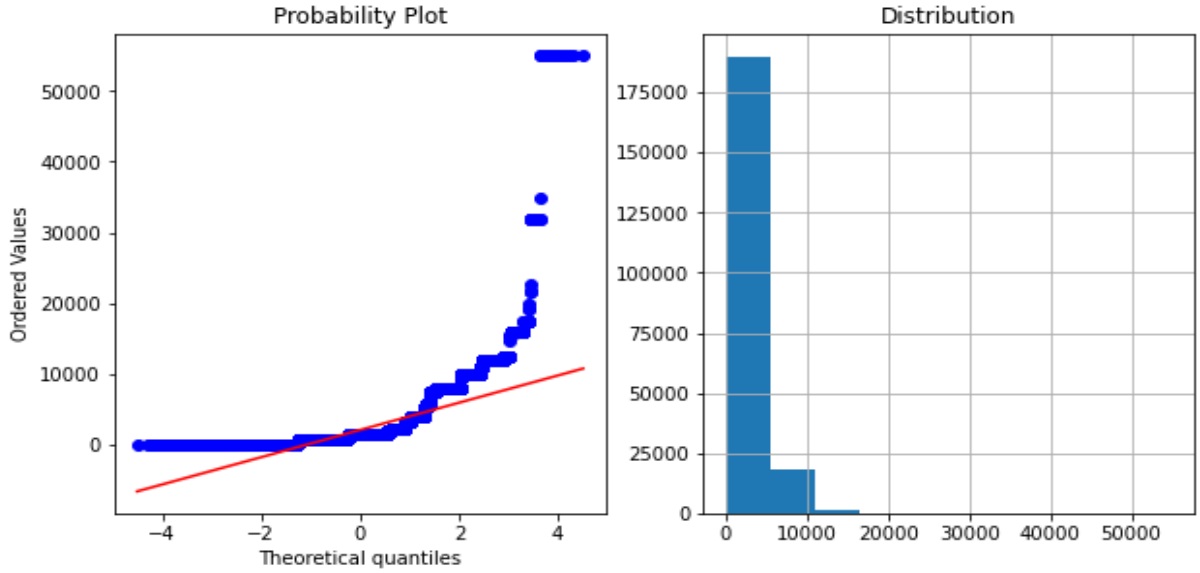
```
(0.0, 0.0)
```

we can see other than 0 all other values in feature last_rech_date_da are outlier, if we replace outliers with 0 then there will be only one unique value i.e. 0, hence i consider to drop this column.

```
data.drop("last_rech_date_da", axis=1, inplace=True)
```

last_rech_amt_ma:

```
distribution(data,"last_rech_amt_ma")
```



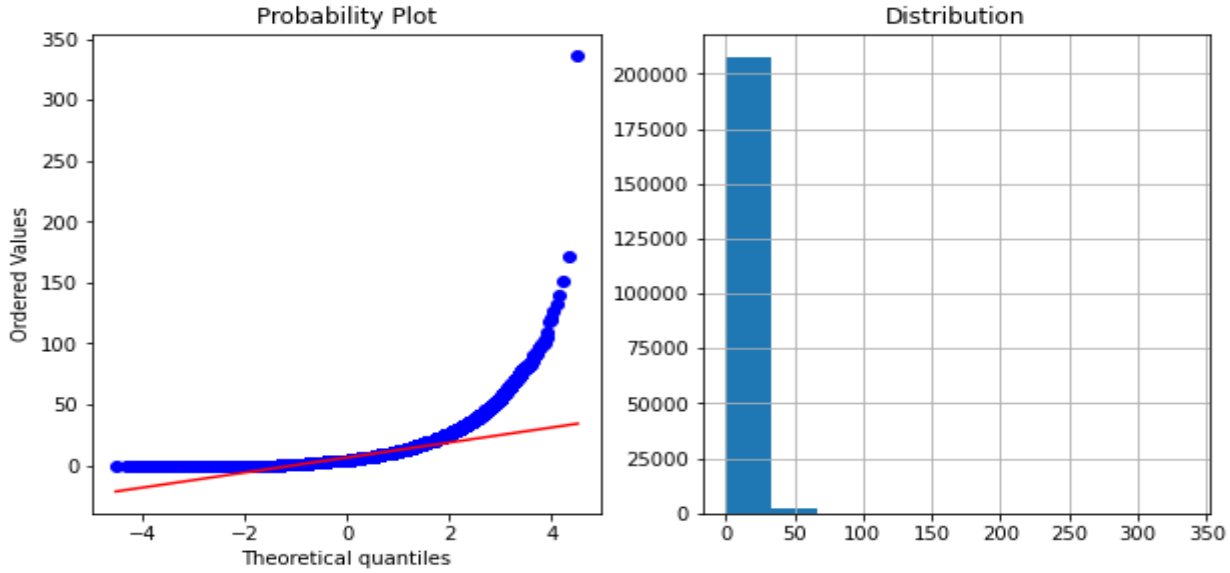

```
#We can see last_rech_amt_ma is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"last_rech_amt_ma")
```

```
(-3847.0, 6926.0)
```

```
data.loc[data['last_rech_amt_ma']>= 6926.0,'last_rech_amt_ma']=6926.0
```

cnt_ma_rech90:

```
distribution(data,"cnt_ma_rech90")
```



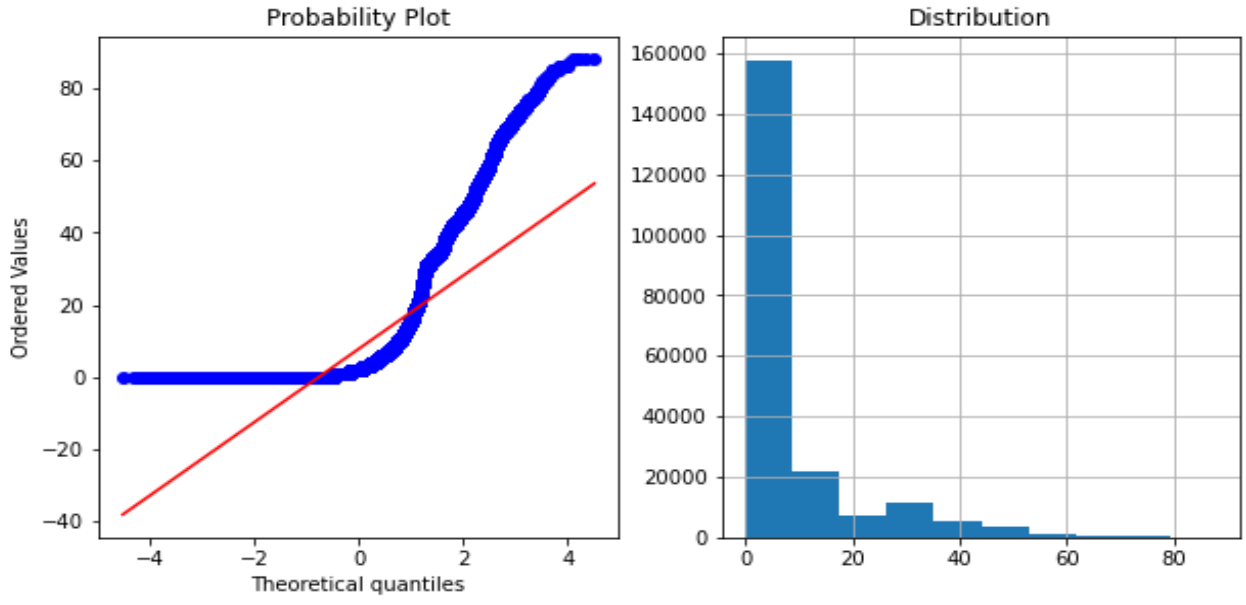

```
#We can see cnt_ma_rech90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"cnt_ma_rech90")
```

```
(-16.0, 26.0)
```

```
data.loc[data['cnt_ma_rech90']>= 26.0,'cnt_ma_rech90']=26.0
```

fr_ma_rech90:

```
distribution(data,"fr_ma_rech90")
```



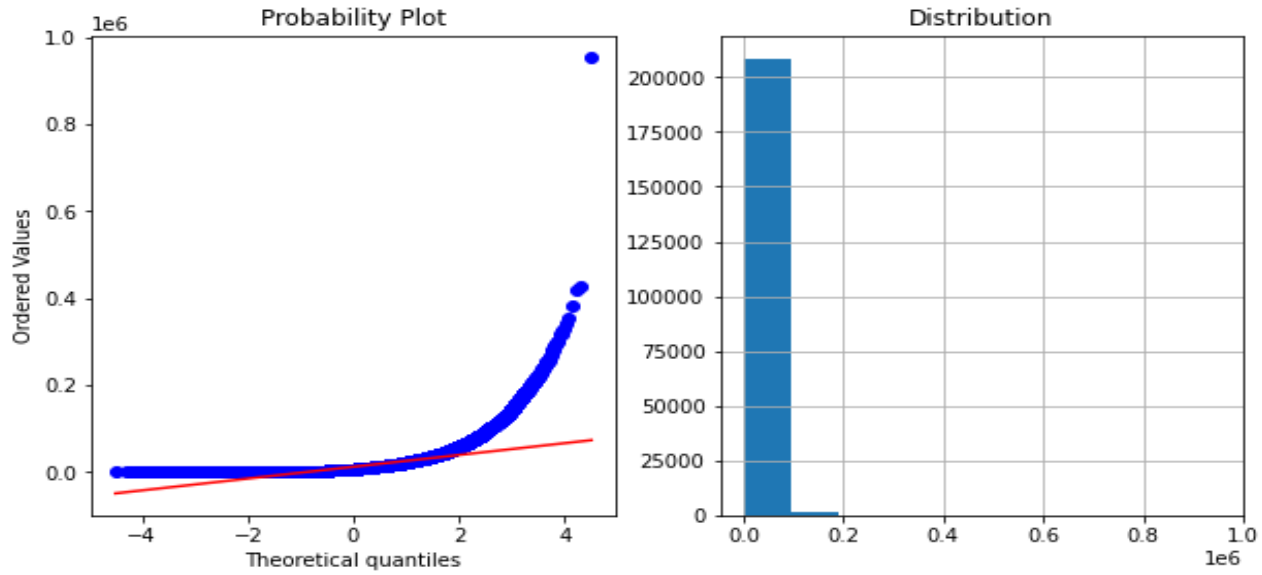

```
#We can see aon is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"fr_ma_rech90")
```

```
(-24.0, 32.0)
```

```
data.loc[data['fr_ma_rech90']>= 32.0,'fr_ma_rech90']=32.0
```

sumamnt_ma_rech90:

```
distribution(data,"sumamnt_ma_rech90")
```



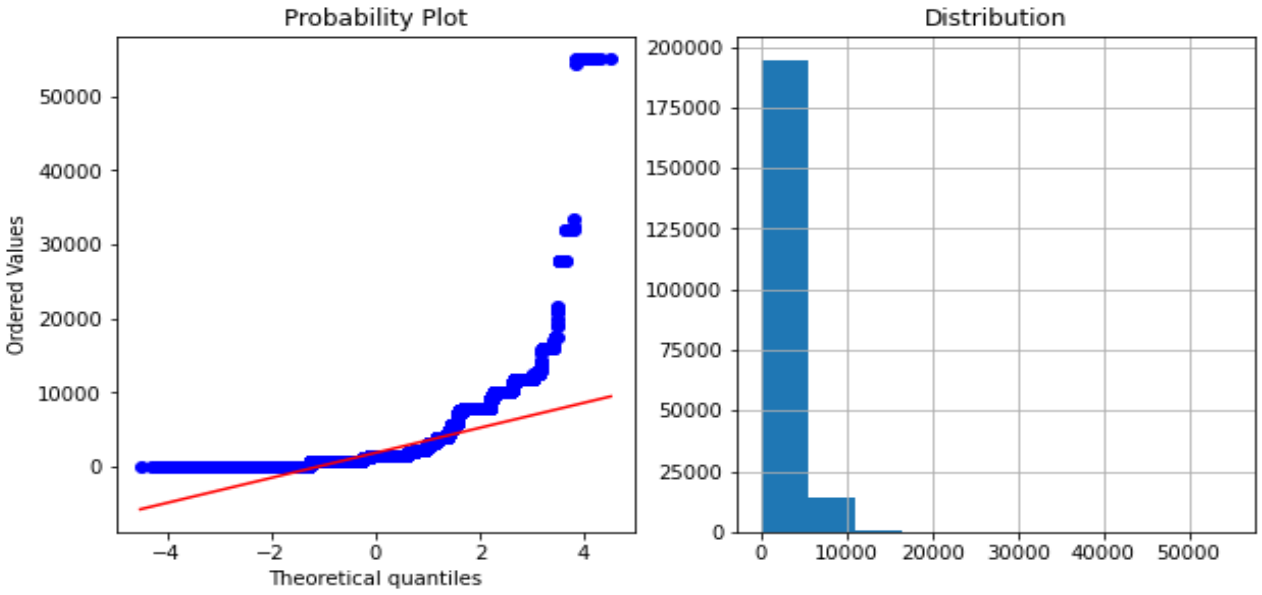

```
#We can see sumamnt_ma_rech90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"sumamnt_ma_rech90")
```

```
(-38732.0, 57049.0)
```

```
data.loc[data['sumamnt_ma_rech90']>= 57049.0,'sumamnt_ma_rech90']=57049.0
```

medianamnt_ma_rech90:

```
distribution(data,"medianamnt_ma_rech90")
```



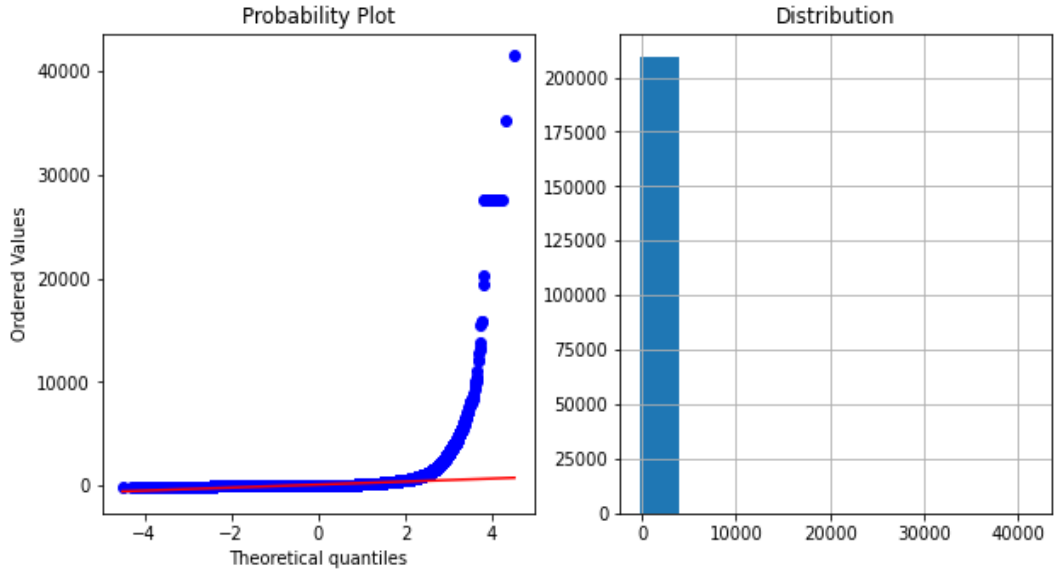

```
#We can see medianamnt_ma_rech90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"medianamnt_ma_rech90")
```

```
(-2680.0, 5377.0)
```

```
data.loc[data['medianamnt_ma_rech90']>= 5377.0,'medianamnt_ma_rech90']=5377.0
```

medianmarechprebal90:

```
distribution(data,"medianmarechprebal90")
```



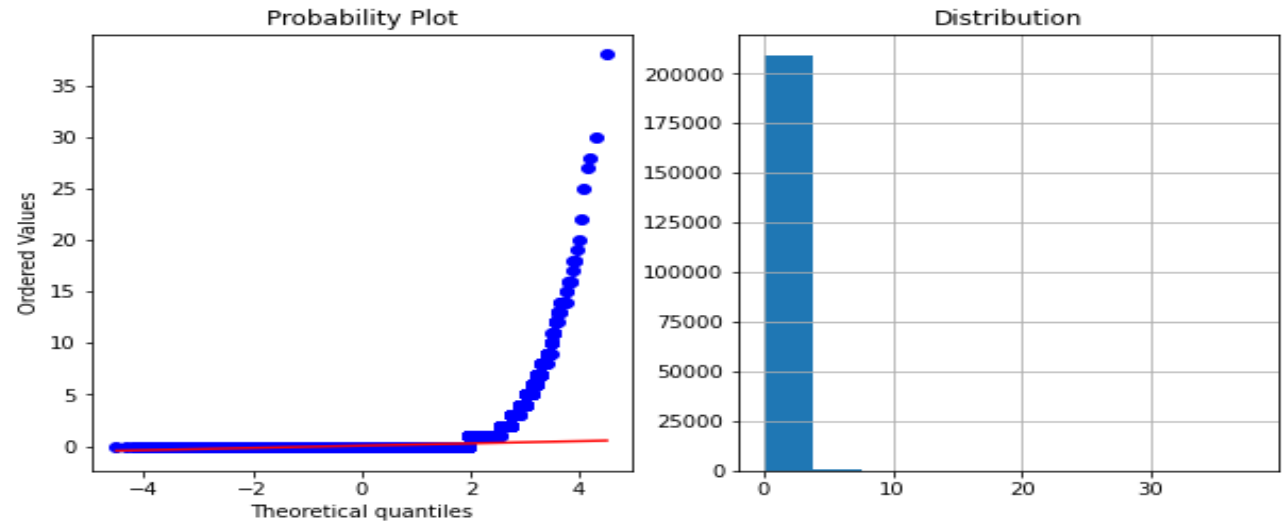

```
#We can see medianmarechprebal90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"medianmarechprebal90")
```

```
(-179.53000000000017, 273.4400000000005)
```

```
data.loc[data['medianmarechprebal90']>= 273.4400000000005,'medianmarechprebal90']=273.4400000000005
```

cnt_da_rech90:

```
distribution(data,"cnt_da_rech90")
```




```
#We can see cnt_da_rech90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"cnt_da_rech90")
```

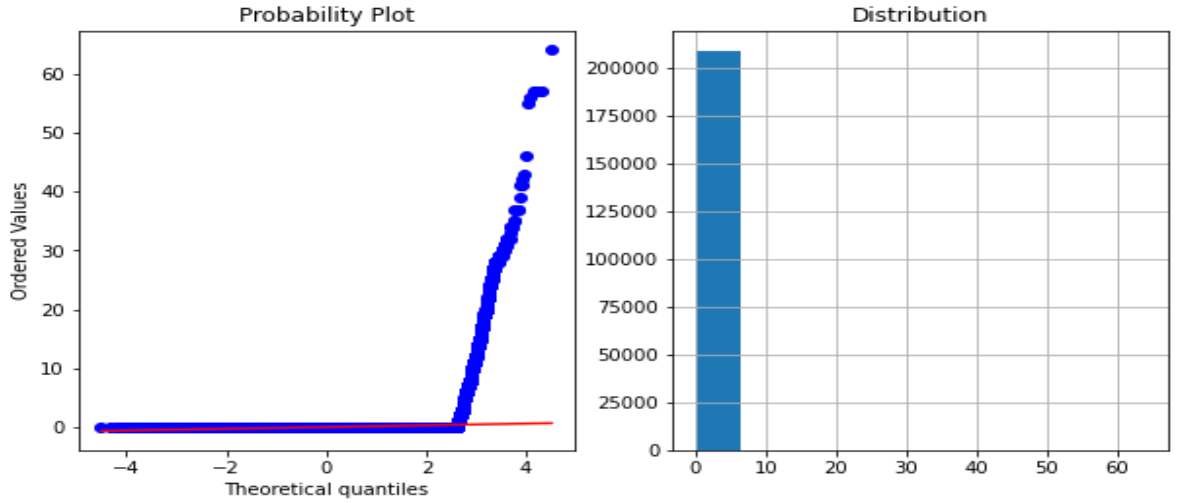
```
(0.0, 0.0)
```

we can see other than 0 all other values in feature cnt_da_rech90 are outliers, if we replace outliers with 0 then there will be only one unique value i.e. 0, hence i consider to drop this column.

```
data.drop("cnt_da_rech90", axis=1, inplace=True)#feature has been dropped
```

fr_da_rech90:

```
distribution(data,"fr_da_rech90")
```




```
#We can see aon is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"fr_da_rech90")
```

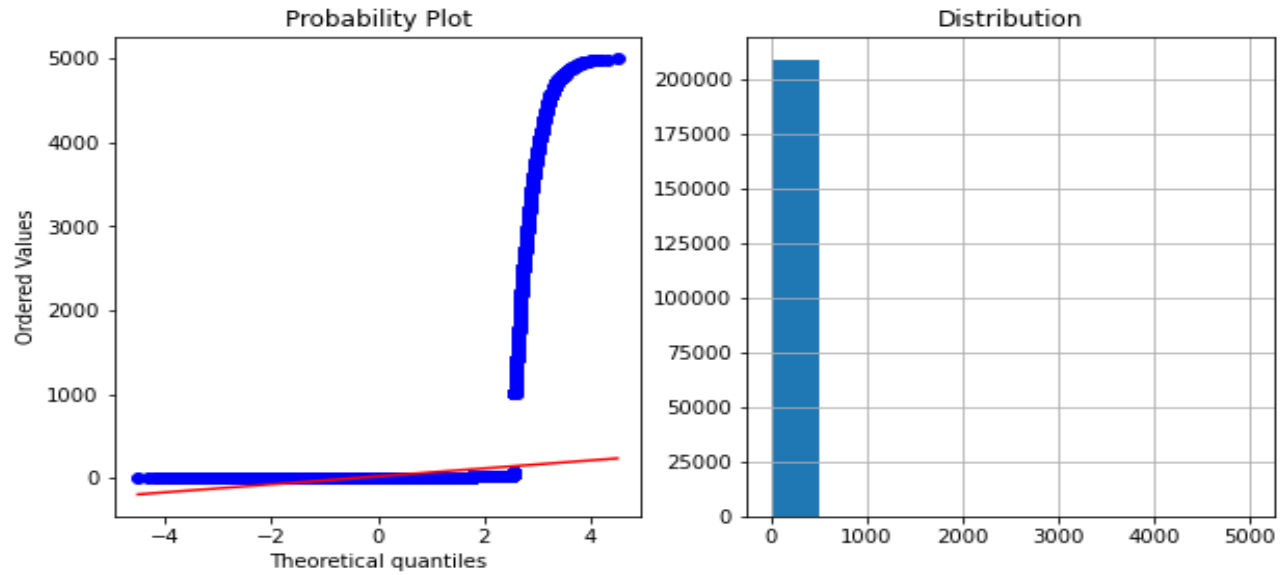
```
(0.0, 0.0)
```

we can see other than 0 all other values in feature fr_da_rech90 are outliers, if we replace outliers with 0 then there will be only one unique value i.e. 0, hence i consider to drop this column.

```
data.drop("fr_da_rech90", axis=1, inplace=True)
```

cnt_loans90:

```
distribution(data,"cnt_loans90")
```



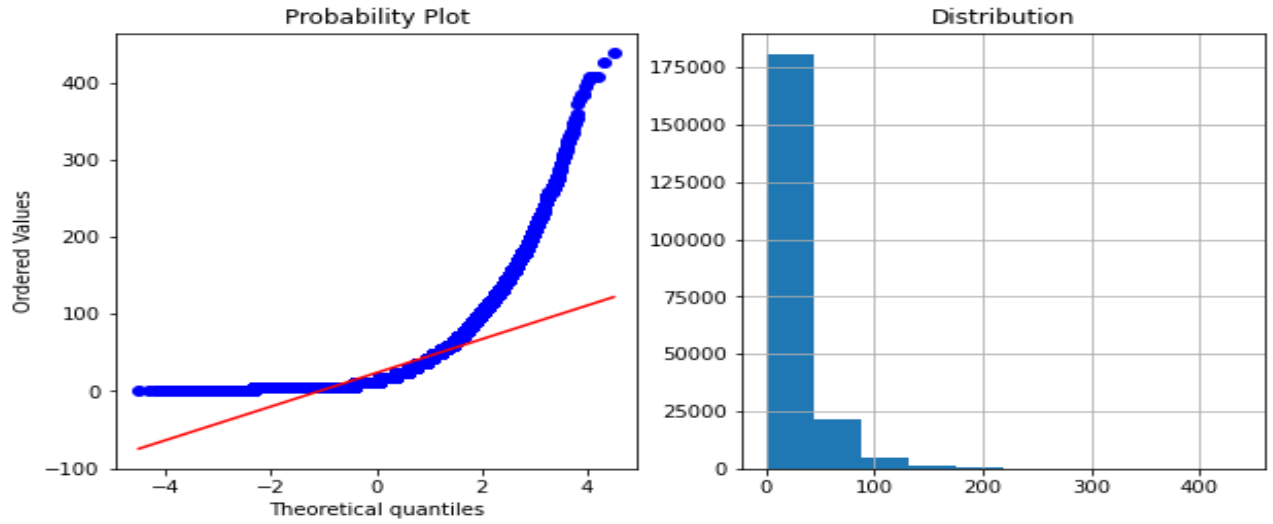

```
#We can see cnt_loans90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"cnt_loans90")
```

```
(-11.0, 17.0)
```

```
data.loc[data['cnt_loans90']>= 17.0,'cnt_loans90']=17.0
```

amnt_loans90:

```
distribution(data,"amnt_loans90")
```



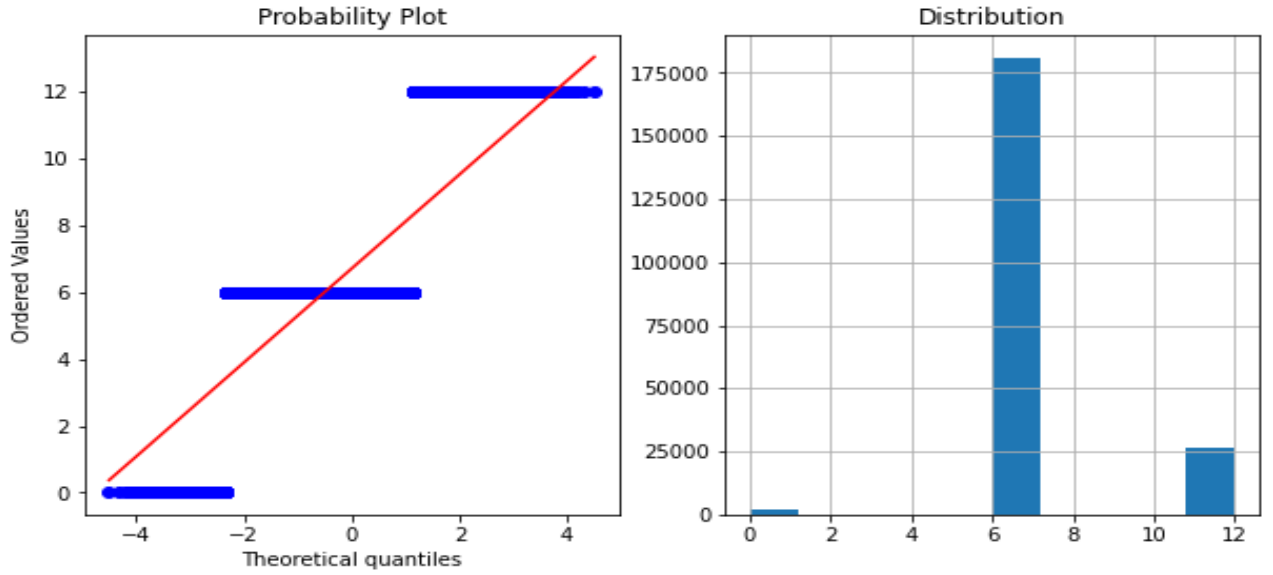

```
#We can see amnt_loans90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"amnt_loans90")
```

```
(-66.0, 102.0)
```

```
data.loc[data['amnt_loans90']>= 102.05,'amnt_loans90']=102.0
```

maxamnt_loans90:

```
distribution(data,"maxamnt_loans90")
```




```
#We can see maxamnt_loans90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"maxamnt_loans90")
```

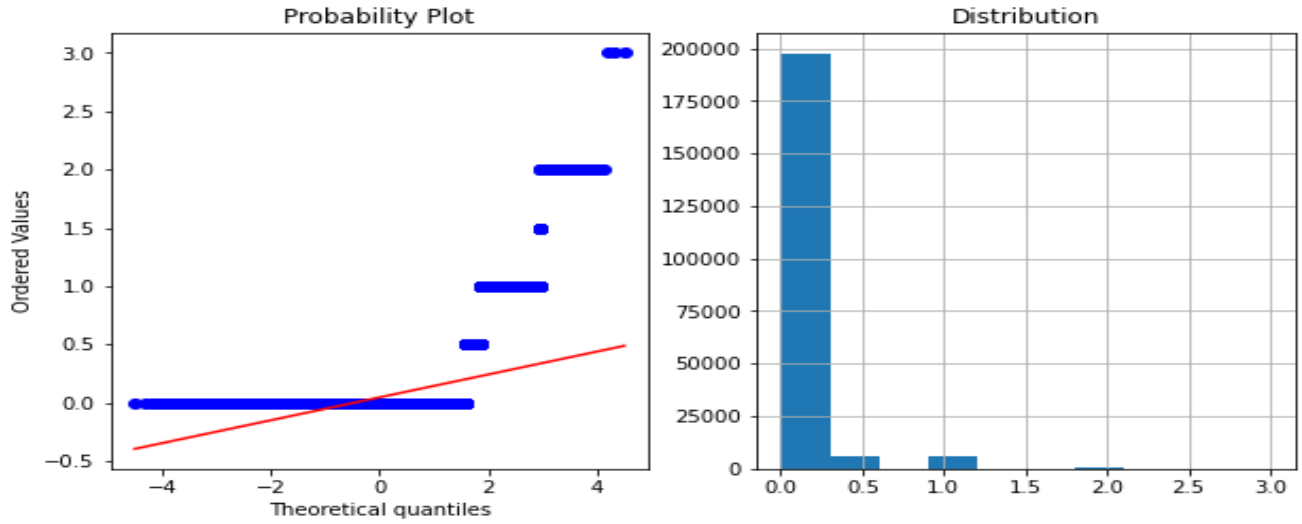
```
(6.0, 6.0)
```

we can see other than 6.0 all other values in feature maxamnt_loans90 are outliers, if we replace outliers with 6.0 then there will be only one unique value i.e. 6.0, hence i consider to drop this column.

```
data.drop("maxamnt_loans90", axis=1, inplace=True)
```

medianamnt_loans90:

```
distribution(data,"medianamnt_loans90")
```




```
#We can see medianamnt_loans90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"medianamnt_loans90")
```

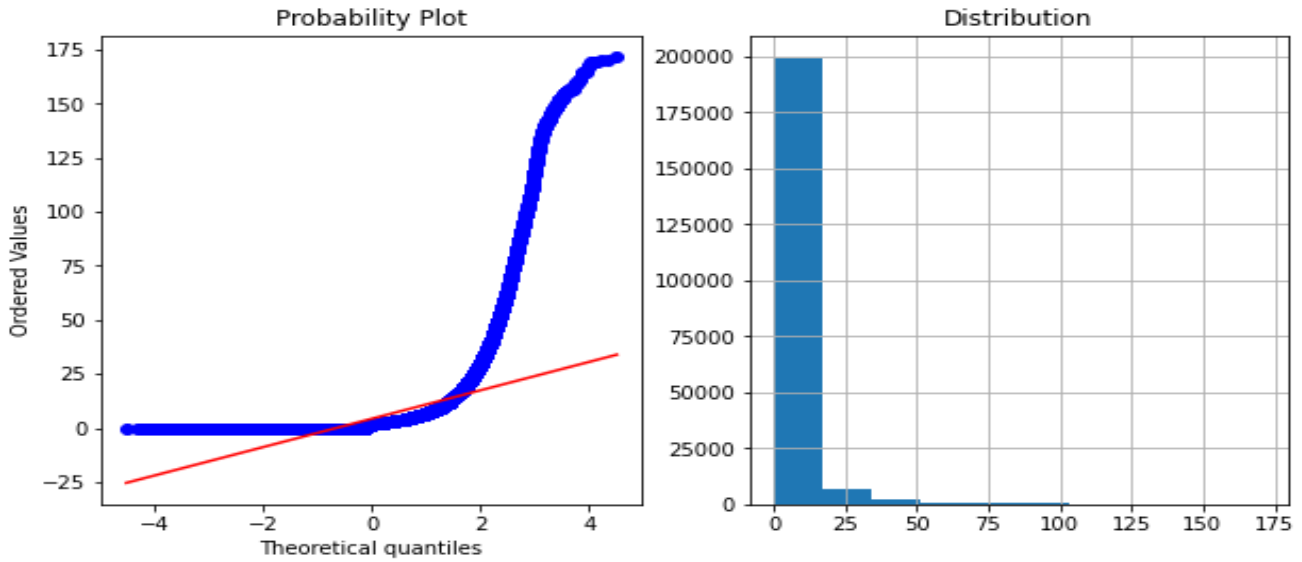
```
(0.0, 0.0)
```

we can see other than 0 all other values in feature medianamnt_loans90 are outliers, if we replace outliers with 0 then there will be only one unique value i.e. 0, hence i consider to drop this column.

```
data.drop('medianamnt_loans90', axis=1, inplace=True)
```

payback90:

```
distribution(data,"payback90")
```



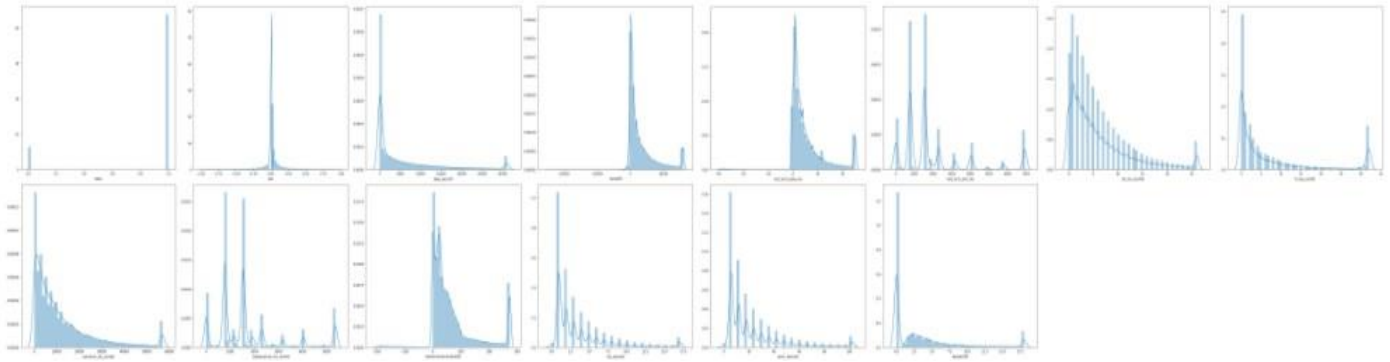

```
#We can see payback90 is highly skewed hence i will go with IQR_h method
outlier_IQR_h(data,"payback90")
```

```
(-13.5, 18.0)
```

```
data.loc[data['payback90']>= 18.0,'payback90']=18.0
```

Let’s check the distribution with the help of distplot:

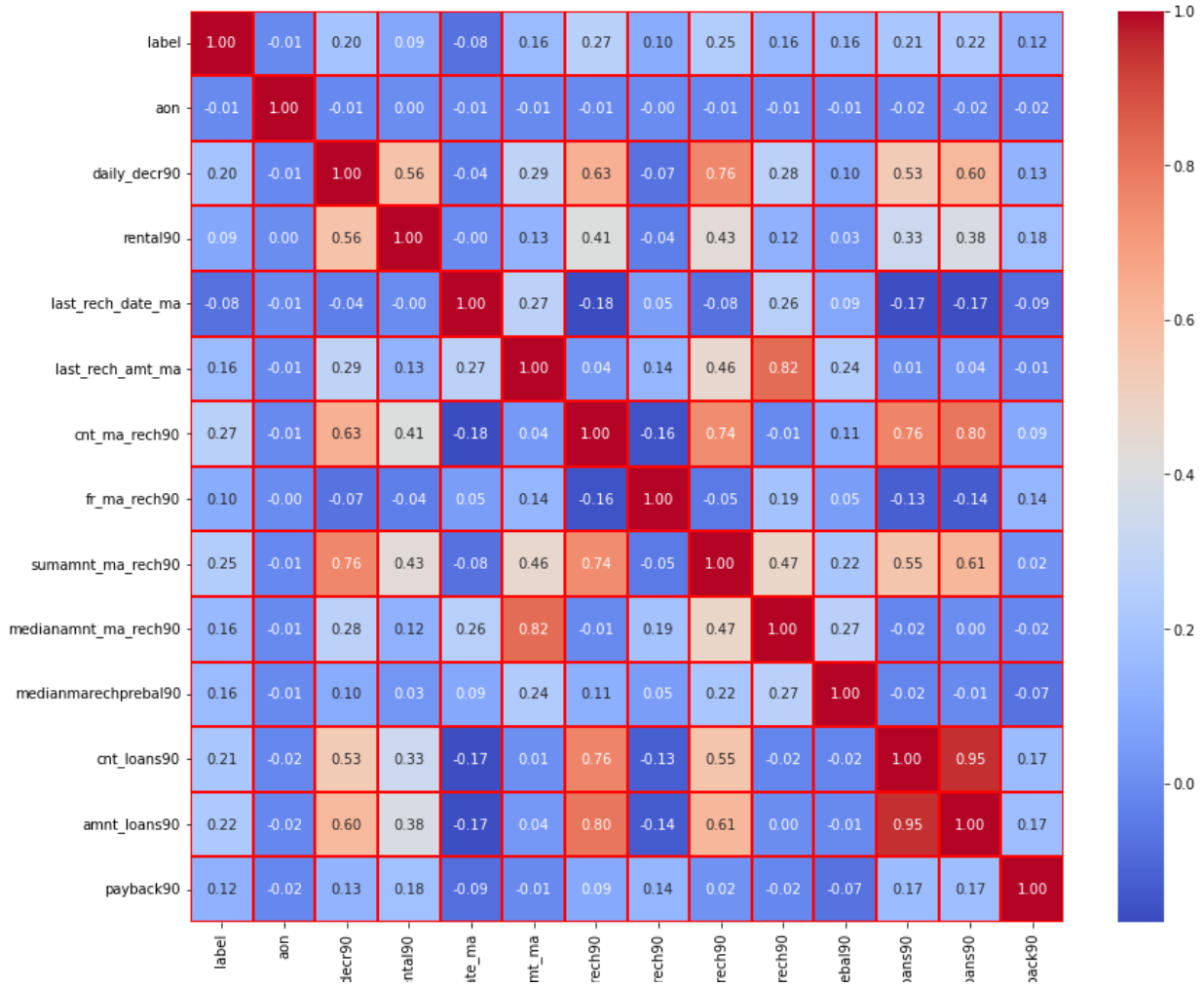
```
collist=data.columns.values
ncol=8
nrows=8
plt.figure(figsize=(8*ncol,7*ncol),edgecolor="black")
for i in range (0,len(collist)):
    if data.dtypes[i] != 'object':
        plt.subplot(nrows,ncol,i+1)
        sns.distplot(data[data.columns[i]])
        plt.tight_layout()
```



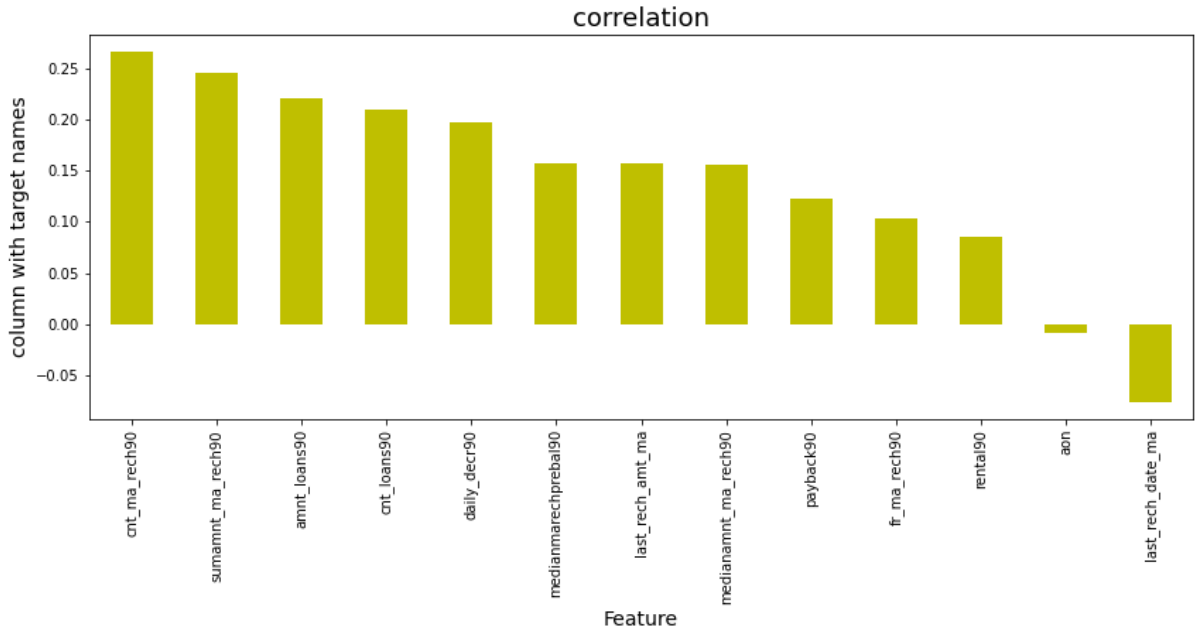
Let’s check the correlation with the help of heatmap:

```
plt.figure(figsize=(14,12))
sns.heatmap(data.corr(),annot=True,linewidths=0.1,linecolor="red",fmt="0.2f",cmap="coolwarm")
```

```
<AxesSubplot:>
```



```
plt.figure(figsize=(14,5))
data.corr()['label'].sort_values(ascending=False).drop(['label']).plot(kind='bar',color='y')
plt.xlabel('Feature',fontsize=14)
plt.ylabel('column with target names',fontsize=14)
plt.title('correlation',fontsize=18)
plt.show()
```

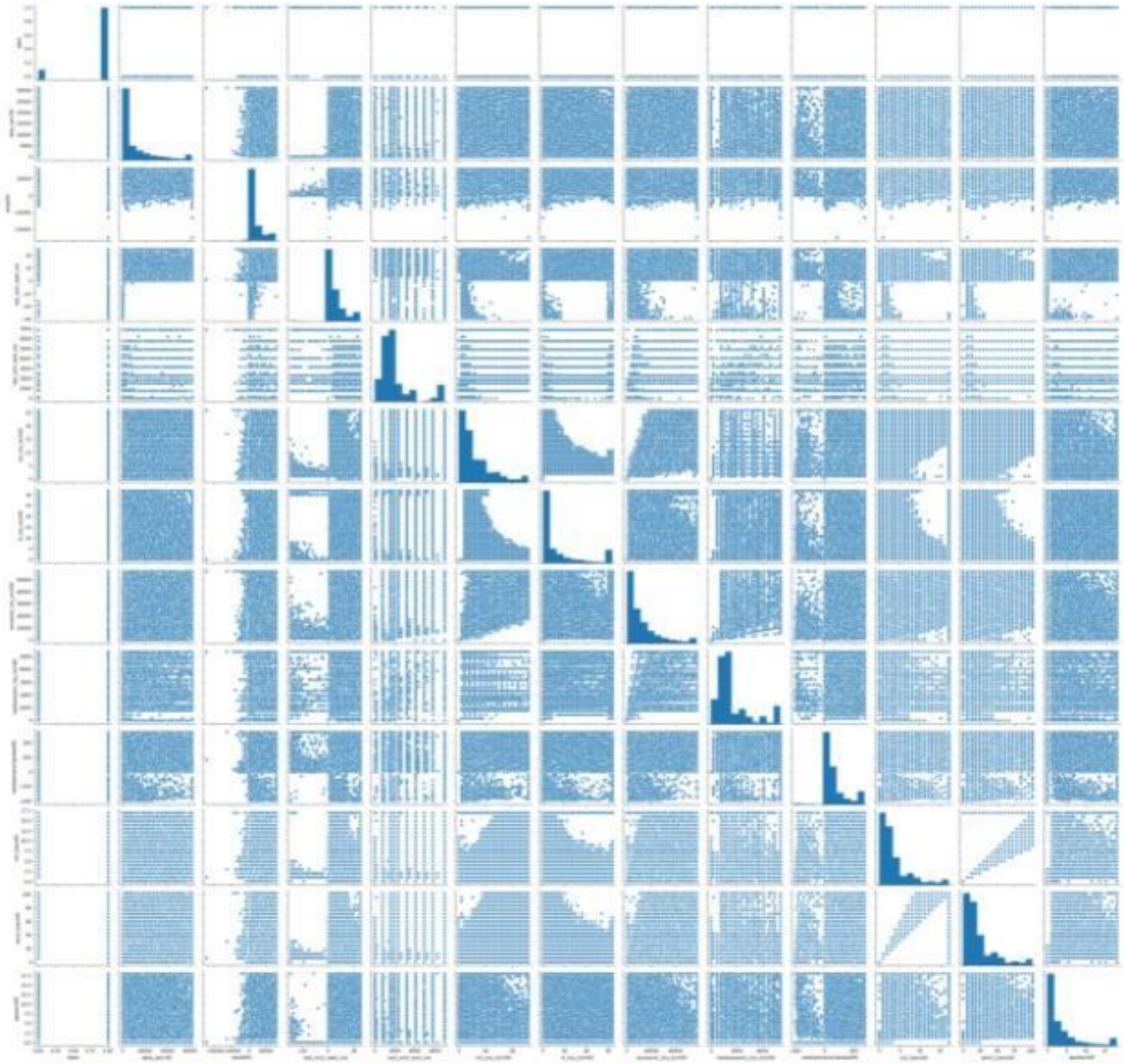


aon columns have very less correlation with our target column, hence we can drop that.

```
data.drop('aon', axis=1, inplace=True)
```

```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x2a880c523d0>
```



Skewness and outlier's removal:

```
data.skew()
```

```
label                   -2.270254
daily_decr90             1.785188
rental90                 1.765769
last_rech_date_ma        1.086744
last_rech_amt_ma         1.656548
cnt_ma_rech90            1.516680
fr_ma_rech90             1.650263
sumamnt_ma_rech90        1.760046
medianamnt_ma_rech90     1.431250
medianmarechprebal90     1.392096
cnt_loans90              1.911960
amnt_loans90             1.829131
payback90                1.825511
dtype: float64
```

As we have already Dealt with outliers.

```
x=data.drop('label', axis=1)
y=data['label']
print(x.shape)
print(y.shape)
```

```
(209593, 12)
(209593,)
```

```
from sklearn.preprocessing import power_transform
data_new=power_transform(x)

data_new=pd.DataFrame(data_new,columns=x.columns)
```

```
data_new.skew()
```

```
daily_decr90            -6.244673
rental90                -1.046043
last_rech_date_ma        0.015803
last_rech_amt_ma        -0.172648
cnt_ma_rech90           -0.013313
fr_ma_rech90             0.128961
sumamnt_ma_rech90       -0.274652
medianamnt_ma_rech90    -0.243194
medianmarechprebal90     0.400360
cnt_loans90              0.100062
amnt_loans90            -0.002188
payback90                0.182777
dtype: float64
```

Skewness has now been reduced.

Let's perform scaling now:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit_transform(data_new)
```

```
array([[ 0.31182077, -0.66376346, -0.44899921, ..., -0.20174088,
        -0.25940212,  1.68102054],
       [ 1.08730494,  0.22402666,  1.98972434, ..., -1.03840847,
        -0.25940212, -1.02147618],
       [-0.02607081, -0.47076755, -0.29895889, ..., -1.03840847,
        -1.01274484, -1.02147618],
       ...,
       [ 1.07513081,  1.33982491, -0.29895889, ...,  1.05993228,
         1.39349071,  0.7616568 ],
       [ 1.11163267, -0.44719923, -0.44899921, ...,  0.29159119,
         0.50728348,  1.39398029],
       [ 0.50657201, -0.54811168,  1.08388664, ..., -0.20174088,
         0.18909619, -1.02147618]])
```

```
x=data_new
```

Model/s Development and Evaluation

Finding best random state:

```
maxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i)
    LR=LogisticRegression()
    LR.fit(x_train,y_train)
    pred=LR.predict(x_test)
    acc=accuracy_score(y_test,pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is ",maxAccu, " on Random State ",maxRS)
```

```
Best accuracy is  0.8836565757770939  on Random State  81
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=maxRS)
```

```
model=[LogisticRegression(),DecisionTreeClassifier(),SVC(),AdaBoostClassifier(),RandomForestClassifier()]

for m in model:
    m.fit(x_train,y_train)
    m.score(x_train,y_train)
    predm=m.predict(x_test)
    print('Accuracy Score of',m,'is:')
    print(accuracy_score(y_test,predm))
    print(confusion_matrix(y_test,predm))
    print(classification_report(y_test,predm))
    print('\n')
```

```
Accuracy Score of LogisticRegression() is:
0.8836565757770939
[[  500  4603]
 [  274 36542]]
              precision    recall  f1-score   support

           0       0.65      0.10      0.17      5103
           1       0.89      0.99      0.94     36816

    accuracy                           0.88     41919
   macro avg       0.77      0.55      0.55     41919
weighted avg       0.86      0.88      0.84     41919


Accuracy Score of DecisionTreeClassifier() is:
0.865263961449462
[[ 2554  2549]
 [ 3099 33717]]
              precision    recall  f1-score   support

           0       0.45      0.50      0.47      5103
           1       0.93      0.92      0.92     36816

    accuracy                           0.87     41919
   macro avg       0.69      0.71      0.70     41919
weighted avg       0.87      0.87      0.87     41919
```

```
Accuracy Score of SVC() is:
0.8998306257305756
[[ 1405  3698]
 [  501 36315]]
              precision    recall  f1-score   support

           0       0.74      0.28      0.40      5103
           1       0.91      0.99      0.95     36816

    accuracy                           0.90     41919
   macro avg       0.82      0.63      0.67     41919
weighted avg       0.89      0.90      0.88     41919


Accuracy Score of AdaBoostClassifier() is:
0.9067487296929794
[[ 1525  3578]
 [  331 36485]]
              precision    recall  f1-score   support

           0       0.82      0.30      0.44      5103
           1       0.91      0.99      0.95     36816

    accuracy                           0.91     41919
   macro avg       0.87      0.64      0.69     41919
weighted avg       0.90      0.91      0.89     41919
```

```
Accuracy Score of RandomForestClassifier() is:
0.910828025477707
[[ 2215  2888]
 [  850 35966]]
              precision    recall  f1-score   support

           0       0.72      0.43      0.54      5103
           1       0.93      0.98      0.95     36816

    accuracy                           0.91     41919
   macro avg       0.82      0.71      0.75     41919
weighted avg       0.90      0.91      0.90     41919
```

We can see AdaBoostClassifier has 90.67% accuracy with 91 f1 score and RandomForestClassifier has 91.08% accuracy with 91 f1 score hence let's do hyper parameter tunning with these two models.

Hyper parameter tunning:

1. AdaBoostClassifier:

```
from sklearn.model_selection import RandomizedSearchCV
#creating  parameter list to pass in RandomizedSearchCV

parameters={'base_estimator':[None],'n_estimators':[20,40, 50,75,100],
            'learning_rate':[0.1,0.01,1.0,2.0],'algorithm':['SAMME.R','SAMME'],
            'random_state':range(0,20)}
```

```
RSV=RandomizedSearchCV(AdaBoostClassifier(),parameters,cv=5)
```

```
RSV.fit(x_train,y_train)
```

```
RandomizedSearchCV(cv=5, estimator=AdaBoostClassifier(),
                   param_distributions={'algorithm': ['SAMME.R', 'SAMME'],
                                        'base_estimator': [None],
                                        'learning_rate': [0.1, 0.01, 1.0, 2.0],
                                        'n_estimators': [20, 40, 50, 75, 100],
                                        'random_state': range(0, 20)})
```

```
RSV.best_params_
```

```
{'random_state': 16,
 'n_estimators': 40,
 'learning_rate': 1.0,
 'base_estimator': None,
 'algorithm': 'SAMME.R'}
```

```
RSV_pred=RSV.best_estimator_.predict(x_test)
```

```
RSV_pred
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

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

```
0.902763696220046
```

Hyper parameter tunning with adaboost classifier gives us 90.28 % accuracy.

2. RandomForestClassifier

```
parameters={'n_estimators':[100],'criterion':['gini','entropy'],'max_depth':[None],
            'min_samples_split':[2],'min_samples_leaf':[1],'min_weight_fraction_leaf':[0.0],
            'max_features':['auto'],'max_leaf_nodes':[None],'min_impurity_decrease':[0.0],
            'min_impurity_split':[None],'bootstrap':[True,False],'oob_score':[True,False],'n_jobs':[None],
            'random_state':range(0,20),'verbose':[0],'warm_start':[True,False],'class_weight':[None],
            'ccp_alpha':[0.0],'max_samples':[None]}
```

```
RSV=RandomizedSearchCV(RandomForestClassifier(),parameters,cv=5)
```

```
RSV.fit(x_train,y_train)
```

```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(),
                   param_distributions={'bootstrap': [True, False],
                                        'ccp_alpha': [0.0],
                                        'class_weight': [None],
                                        'criterion': ['gini', 'entropy'],
                                        'max_depth': [None],
                                        'max_features': ['auto'],
                                        'max_leaf_nodes': [None],
                                        'max_samples': [None],
                                        'min_impurity_decrease': [0.0],
                                        'min_impurity_split': [None],
                                        'min_samples_leaf': [1],
                                        'min_samples_split': [2],
                                        'min_weight_fraction_leaf': [0.0],
                                        'n_estimators': [100], 'n_jobs': [None],
                                        'oob_score': [True, False],
                                        'random_state': range(0, 20),
                                        'verbose': [0],
                                        'warm_start': [True, False]})
```

```
RSV.best_params_
```

```
{'warm_start': False,
 'verbose': 0,
 'random_state': 0,
 'oob_score': True,
 'n_jobs': None,
 'n_estimators': 100,
 'min_weight_fraction_leaf': 0.0,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'min_impurity_split': None,
 'min_impurity_decrease': 0.0,
 'max_samples': None,
 'max_leaf_nodes': None,
 'max_features': 'auto',
 'max_depth': None,
 'criterion': 'gini',
 'class_weight': None,
 'ccp_alpha': 0.0,
 'bootstrap': True}
```

```
RSV_pred=RSV.best_estimator_.predict(x_test)
```

```
RSV_pred
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

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

```
0.9964514474516025
```

Hyper parameter tunning with RandomForestClassifier gives us 99.65 % accuracy.

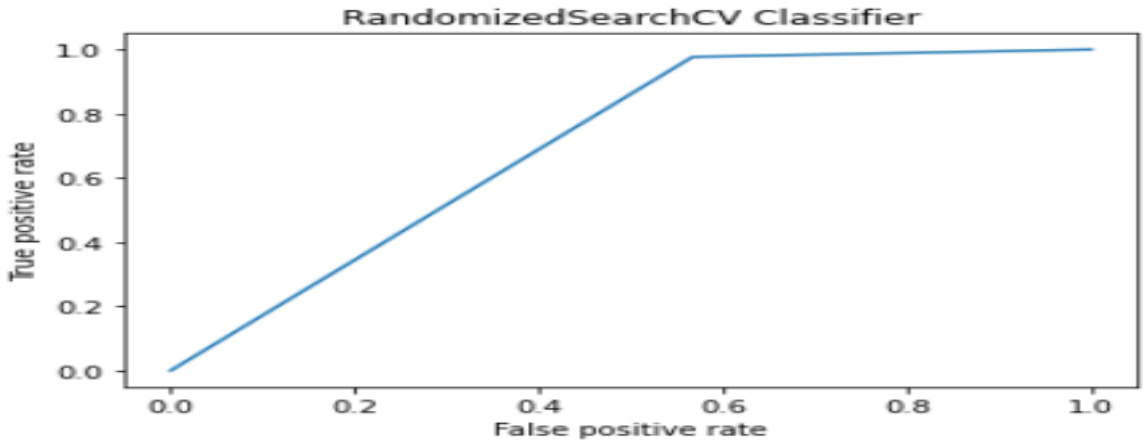
AUC_ROC:

1. RandomizedSearchCV:

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

```
#RandomizedSearchCV
fpr,tpr,thresholds=roc_curve(y_test,RSV_pred)
```

```
plt.plot([0.1],[0.1],'k--')
plt.plot(fpr,tpr,label='RandomizedSearchCV')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('RandomizedSearchCV Classifier')
plt.show()
```

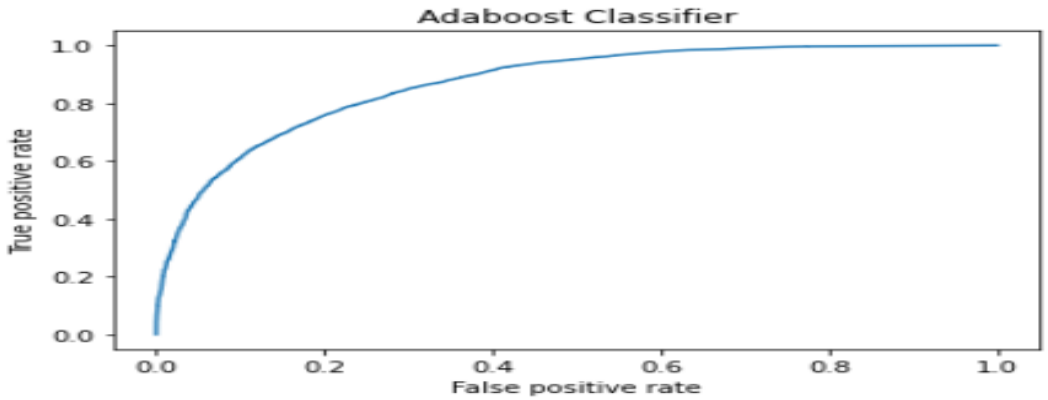


2. Adaboost Classifier:

```
ad=AdaBoostClassifier()
ad.fit(x_train,y_train)
auc_score=roc_auc_score(y_test,ad.predict(x_test))
print(auc_score)
```

```
0.6449265805629585
```

```
#Adaboost classifier curve
y_pred_prob=ad.predict_proba(x_test)[:,1]
fpr,tpr,thresholds=roc_curve(y_test,y_pred_prob)
plt.plot([0.1],[0.1],'k--')
plt.plot(fpr,tpr,label='Adaboost Classifier')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('Adaboost Classifier')
plt.show()
```

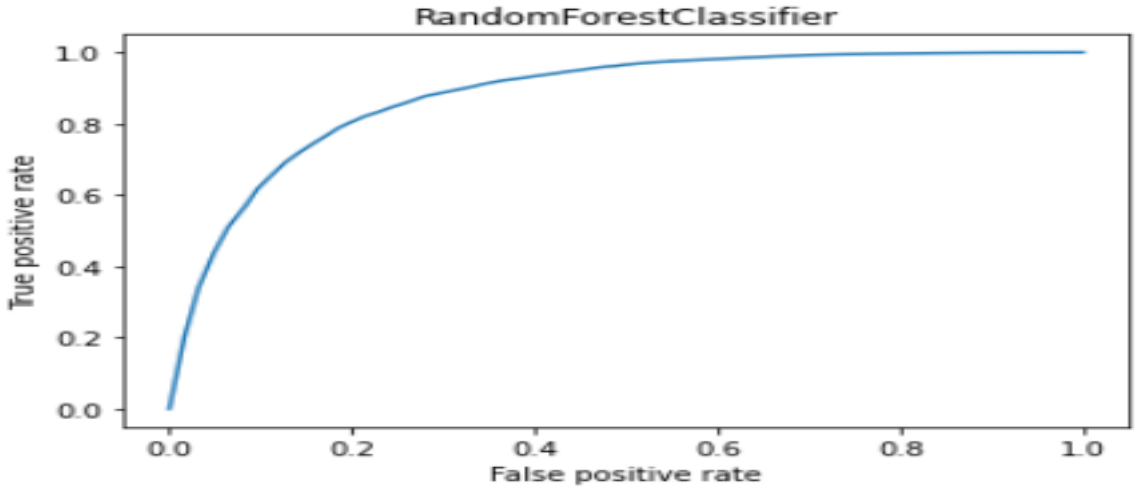


3. RandomForestClassifier:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
auc_score=roc_auc_score(y_test,rfc.predict(x_test))
print(auc_score)
```

```
0.7057598398033111
```

```
#RandomForestClassifier
y_pred_prob=rfc.predict_proba(x_test)[:,1]
fpr,tpr,thresholds=roc_curve(y_test,y_pred_prob)
plt.plot([0.1],[0.1],'k--')
plt.plot(fpr,tpr,label='RandomForestClassifier')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('RandomForestClassifier')
plt.show()
```



As we got best accuracy score with RandomForest Classifier and AUC_roc score is also graph is better than another model hence I am saving that model.

Saving the model:

```
import joblib
joblib.dump(RSV,"RSVMCDP.obj")
```

```
['RSVMCDP.obj']
```

```
RSVfile=joblib.load("RSVMCDP.obj")
RSVfile.predict(x_test)
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

CONCLUSION

A Microfinance Institution (MFI) is an organization that offers financial services to low-income popu lations. Fixed wireless telecommunications network provider collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. Our database consists of 209593 rows and 37 columns in which we have 3 object type columns and all other columns are numeric. First, I have dropped some columns which are not required also I have dropped all columns with last 30 d ays data as we have the same column with last 90 days data that means we will get that info in on e or the other way.

Our database had outliers in many features then I worked on all features individually to check outli ers and replaced them with relevant data OR I took necessary action. Then I have checked correla tion of data and also dropped features which have very less correlation with our target variable. Th en with the help of the power transform method I tried to reduce skewness of the data. With the he lp of standard scalar, I have standardized the data.

Used 5 methods for model building then I found that Adaboost classifier and Random Forest Class ifier have good accuracy and good f1 score as well. With the help of RandomizedSearchCV I have tried to improve accuracy. AUC ROC helped me to make a decision which model is better than I decided to go ahead with Random Forest Classifier and saved model.