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).

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.

In [3]:

```
#Data Reading and Analysis
```

In [4]:

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

In [10]:

```
#Reading data
df=pd.read_csv('Data file.csv')
df.head()
```

Out[10]:

Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech

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

5 rows × 37 columns

In [11]:
```
df.drop('Unnamed: 0',axis=1,inplace=True)
```

In [12]:
```
df.head()
```

Out[12]:

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

5 rows × 36 columns

In [13]:
```
#let's dive into depth
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 36 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   label                 209593 non-null  int64  
 1   msisdn                209593 non-null  object 
 2   aon                   209593 non-null  float64
 3   daily_decr30          209593 non-null  float64
 4   daily_decr90          209593 non-null  float64
 5   rental30              209593 non-null  float64
 6   rental90              209593 non-null  float64
 7   last_rech_date_ma     209593 non-null  float64
 8   last_rech_date_da     209593 non-null  float64
 9   last_rech_amt_ma      209593 non-null  int64  
 10  cnt_ma_rech30         209593 non-null  int64  
```

```
 11  fr_ma_rech30          209593 non-null  float64
 12  sumamnt_ma_rech30     209593 non-null  float64
 13  medianamnt_ma_rech30  209593 non-null  float64
 14  medianmarechprebal30  209593 non-null  float64
 15  cnt_ma_rech90         209593 non-null  int64
 16  fr_ma_rech90          209593 non-null  int64
 17  sumamnt_ma_rech90     209593 non-null  int64
 18  medianamnt_ma_rech90  209593 non-null  float64
 19  medianmarechprebal90  209593 non-null  float64
 20  cnt_da_rech30         209593 non-null  float64
 21  fr_da_rech30          209593 non-null  float64
 22  cnt_da_rech90         209593 non-null  int64
 23  fr_da_rech90          209593 non-null  int64
 24  cnt_loans30           209593 non-null  int64
 25  amnt_loans30          209593 non-null  int64
 26  maxamnt_loans30       209593 non-null  float64
 27  medianamnt_loans30    209593 non-null  float64
 28  cnt_loans90           209593 non-null  float64
 29  amnt_loans90          209593 non-null  int64
 30  maxamnt_loans90       209593 non-null  int64
 31  medianamnt_loans90    209593 non-null  float64
 32  payback30             209593 non-null  float64
 33  payback90             209593 non-null  float64
 34  pcircle               209593 non-null  object
 35  pdate                 209593 non-null  object
dtypes: float64(21), int64(12), object(3)
memory usage: 57.6+ MB
```

In [14]:

```
# let's check null values
df.isnull().sum()
```

Out[14]:

```
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
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
```

```
pcircle                0
pdate                  0
dtype: int64
```

In [15]:
```
print("shape of data set is ",df.shape)
```

```
shape of data set is  (209593, 36)
```

Data Preprocessing

Remove columns where number of unique value is only 1.

Let's look at no of unique values for each column.We will remove all columns where number of unique value is only 1 because that will not make any sense in the analysis

In [18]:
```
unique = df.nunique()
unique = unique[unique.values == 1]
```

In [19]:
```
df.drop(labels = list(unique.index), axis =1, inplace=True)
print("After removing we are left with",df.shape ,"rows & columns.")
```

```
After removing we are left with (209593, 35) rows & columns.
```

In [20]:
```
df.head()
```

Out[20]:

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

5 rows × 35 columns

In [21]:
```
df.describe().transpose()
```

Out[21]:

	count	mean	std	min	25%	50%	75%	
label	209593.0	0.875177	0.330519	0.000000	1.000	1.000000	1.00	
aon	209593.0	8112.343445	75696.082531	-48.000000	246.000	527.000000	982.00	99986
daily_decr30	209593.0	5381.402289	9220.623400	-93.012667	42.440	1469.175667	7244.00	26592
daily_decr90	209593.0	6082.515068	10918.812767	-93.012667	42.692	1500.000000	7802.79	32063
rental30	209593.0	2692.581910	4308.586781	-23737.140000	280.420	1083.570000	3356.94	19892
rental90	209593.0	3483.406534	5770.461279	-24720.580000	300.260	1334.000000	4201.79	20014

	count	mean	std	min	25%	50%	75%	
last_rech_date_ma	209593.0	3755.847800	53905.892230	-29.000000	1.000	3.000000	7.00	99865
last_rech_date_da	209593.0	3712.202921	53374.833430	-29.000000	0.000	0.000000	0.00	99917
last_rech_amt_ma	209593.0	2064.452797	2370.786034	0.000000	770.000	1539.000000	2309.00	5500
cnt_ma_rech30	209593.0	3.978057	4.256090	0.000000	1.000	3.000000	5.00	20
fr_ma_rech30	209593.0	3737.355121	53643.625172	0.000000	0.000	2.000000	6.00	99960
sumamnt_ma_rech30	209593.0	7704.501157	10139.621714	0.000000	1540.000	4628.000000	10010.00	81009
medianamnt_ma_rech30	209593.0	1812.817952	2070.864620	0.000000	770.000	1539.000000	1924.00	5500
medianmarechprebal30	209593.0	3851.927942	54006.374433	-200.000000	11.000	33.900000	83.00	99947
cnt_ma_rech90	209593.0	6.315430	7.193470	0.000000	2.000	4.000000	8.00	33
fr_ma_rech90	209593.0	7.716780	12.590251	0.000000	0.000	2.000000	8.00	8
sumamnt_ma_rech90	209593.0	12396.218352	16857.793882	0.000000	2317.000	7226.000000	16000.00	95303
medianamnt_ma_rech90	209593.0	1864.595821	2081.680664	0.000000	773.000	1539.000000	1924.00	5500
medianmarechprebal90	209593.0	92.025541	369.215658	-200.000000	14.600	36.000000	79.31	4145
cnt_da_rech30	209593.0	262.578110	4183.897978	0.000000	0.000	0.000000	0.00	9991
fr_da_rech30	209593.0	3749.494447	53885.414979	0.000000	0.000	0.000000	0.00	99980
cnt_da_rech90	209593.0	0.041495	0.397556	0.000000	0.000	0.000000	0.00	3
fr_da_rech90	209593.0	0.045712	0.951386	0.000000	0.000	0.000000	0.00	6
cnt_loans30	209593.0	2.758981	2.554502	0.000000	1.000	2.000000	4.00	5
amnt_loans30	209593.0	17.952021	17.379741	0.000000	6.000	12.000000	24.00	30
maxamnt_loans30	209593.0	274.658747	4245.264648	0.000000	6.000	6.000000	6.00	9986
medianamnt_loans30	209593.0	0.054029	0.218039	0.000000	0.000	0.000000	0.00	
cnt_loans90	209593.0	18.520919	224.797423	0.000000	1.000	2.000000	5.00	499
amnt_loans90	209593.0	23.645398	26.469861	0.000000	6.000	12.000000	30.00	43
maxamnt_loans90	209593.0	6.703134	2.103864	0.000000	6.000	6.000000	6.00	1
medianamnt_loans90	209593.0	0.046077	0.200692	0.000000	0.000	0.000000	0.00	
payback30	209593.0	3.398826	8.813729	0.000000	0.000	0.000000	3.75	17
payback90	209593.0	4.321485	10.308108	0.000000	0.000	1.666667	4.50	17

In [22]:

```
#Here we check the summary of object and datetime columns
df.describe(include=['object','datetime']).transpose()
```

Out[22]:

	count	unique	top	freq
msisdn	209593	186243	04581I85330	7
pdate	209593	82	2016-07-04	3150

Observation:

Summary statistics shows all the statistics of our dataset i.e. mean, median and other calculation. Mean is greater than median in all the columns so aur data is right skewed. The difference between 75% and maximum is higher that's why outliers are removed which needs to be removed. The pdate column tells the date when the data is collect. It contains only three month data. msidn is a mobile number of user and mobile number is unique for every customers. There are only 186243 unique number out of 209593 so rest of the data is duplicates entry so we have to remove those entry.

In [24]:

```
df1=df.copy()
```

In [25]:

```
#Deleting the duplicates entry in msidn column
df = df.drop_duplicates(subset = 'msisdn',keep='first')
df.shape
```

Out[25]:

```
(186243, 35)
```

Data Exploration

In [26]:

```
#Printing the object datatypes and their unique values.

for column in df.columns:
    if df[column].dtypes == object:
        print(str(column) + ' : ' + str(df[column].unique()))
        print('*************************************************************************
        print('\n')
```

```
msisdn : ['21408I70789' '76462I70374' '17943I70372' ... '22758I85348' '59712I82733'
 '65061I85339']
*******************************************************************************************
****************


pdate : ['2016-07-20' '2016-08-10' '2016-08-19' '2016-06-06' '2016-06-22'
 '2016-07-02' '2016-07-05' '2016-08-05' '2016-06-15' '2016-06-08'
 '2016-06-12' '2016-06-20' '2016-06-29' '2016-06-16' '2016-08-03'
 '2016-06-24' '2016-07-04' '2016-07-03' '2016-07-01' '2016-08-08'
 '2016-06-26' '2016-06-23' '2016-07-06' '2016-07-09' '2016-06-10'
 '2016-06-07' '2016-06-27' '2016-08-11' '2016-06-30' '2016-06-19'
 '2016-07-26' '2016-08-14' '2016-06-14' '2016-06-21' '2016-06-25'
 '2016-06-28' '2016-06-11' '2016-07-27' '2016-07-23' '2016-08-16'
 '2016-08-15' '2016-06-02' '2016-06-05' '2016-08-02' '2016-07-28'
 '2016-07-18' '2016-08-18' '2016-07-16' '2016-07-29' '2016-07-21'
 '2016-06-03' '2016-06-13' '2016-08-01' '2016-07-13' '2016-07-10'
 '2016-06-09' '2016-07-15' '2016-07-11' '2016-08-09' '2016-08-12'
 '2016-07-22' '2016-06-04' '2016-07-24' '2016-06-18' '2016-08-13'
 '2016-06-17' '2016-08-07' '2016-07-12' '2016-08-06' '2016-07-19'
 '2016-08-21' '2016-08-04' '2016-07-25' '2016-07-30' '2016-08-17'
 '2016-07-08' '2016-07-14' '2016-06-01' '2016-07-07' '2016-07-17'
 '2016-07-31' '2016-08-20']
*******************************************************************************************
****************
```

Observation:

contains only one circle area data. So it have not any impact in our model if we drop this feature.

In [27]:
```python
#Printing the float datatype columns and number of unique values in the particular columns

for column in df.columns:
    if df[column].dtype==np.number:
        print(str(column) + ' : ' + str(df[column].nunique()))
        print(df[column].nunique())
        print('/////*************************************************************************
```

```
C:\Users\Arun\AppData\Local\Temp/ipykernel_12624/319834993.py:4: DeprecationWarning: Conve
rting `np.inexact` or `np.floating` to a dtype is deprecated. The current result is `float
64` which is not strictly correct.
  if df[column].dtype==np.number:
aon : 4282
4282
/////*******************************************************************************
*///////
daily_decr30 : 130323
130323
/////*******************************************************************************
*///////
daily_decr90 : 139842
139842
/////*******************************************************************************
*///////
rental30 : 117881
117881
/////*******************************************************************************
*///////
rental90 : 125595
125595
/////*******************************************************************************
*///////
last_rech_date_ma : 1061
1061
/////*******************************************************************************
*///////
last_rech_date_da : 1061
1061
/////*******************************************************************************
*///////
fr_ma_rech30 : 961
961
/////*******************************************************************************
*///////
sumamnt_ma_rech30 : 13130
13130
/////*******************************************************************************
*///////
medianamnt_ma_rech30 : 501
501
/////*******************************************************************************
*///////
medianmarechprebal30 : 28486
28486
/////*******************************************************************************
*///////
medianamnt_ma_rech90 : 602
602
/////*******************************************************************************
*///////
medianmarechprebal90 : 28064
28064
/////*******************************************************************************
*///////
cnt_da_rech30 : 949
```

```
949
//////***********************************************************************
*///////
fr_da_rech30 : 960
960
//////***********************************************************************
*///////
maxamnt_loans30 : 924
924
//////***********************************************************************
*///////
medianamnt_loans30 : 6
6
//////***********************************************************************
*///////
cnt_loans90 : 968
968
//////***********************************************************************
*///////
medianamnt_loans90 : 6
6
//////***********************************************************************
*///////
payback30 : 1249
1249
//////***********************************************************************
*///////
payback90 : 2128
2128
//////***********************************************************************
*///////
```

In [28]:

```
#Checking the number of number of defaulter and non defaulter customers.
df['label'].value_counts()
```

Out[28]:

```
1    160383
0     25860
Name: label, dtype: int64
```

In [29]:

```
#Checking the defaulter customers percentage wise.
df['label'].value_counts(normalize=True) *100
```

Out[29]:

```
1    86.114914
0    13.885086
Name: label, dtype: float64
```

Observation:

After seeing the label column which is also our target feature for this dataset it is clearly shown that 86.11% of data is label 1 and only 13.8% of data is label 0 so our dataset is implanced. So before making the ML model first we have to do sampling to get rid off imblance dataset.

In [30]:

```
#check cor-relation
df_cor = df.corr()
df_cor
```

Out[30]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	las
label	1.000000	-0.004035	0.174901	0.173016	0.057207	0.075869	0.004113	

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	las
aon	-0.004035	1.000000	0.000630	0.000052	-0.002930	-0.002618	0.001853	
daily_decr30	0.174901	0.000630	1.000000	0.977659	0.427503	0.444932	-0.000171	
daily_decr90	0.173016	0.000052	0.977659	1.000000	0.420561	0.457443	0.000058	
rental30	0.057207	-0.002930	0.427503	0.420561	1.000000	0.955233	-0.000949	
rental90	0.075869	-0.002618	0.444932	0.457443	0.955233	1.000000	-0.001758	
last_rech_date_ma	0.004113	0.001853	-0.000171	0.000058	-0.000949	-0.001758	1.000000	
last_rech_date_da	0.001814	-0.001796	-0.001311	-0.001484	0.003294	0.002643	0.002629	
last_rech_amt_ma	0.139969	0.004102	0.287181	0.275195	0.128773	0.123436	-0.000754	
cnt_ma_rech30	0.244728	-0.004315	0.444365	0.419650	0.220472	0.218618	0.006491	
fr_ma_rech30	0.001129	-0.000436	0.000766	0.001091	0.000272	0.001057	-0.001165	
sumamnt_ma_rech30	0.207727	-0.000397	0.630202	0.597542	0.258656	0.246626	0.002544	
medianamnt_ma_rech30	0.149780	0.004446	0.307440	0.294838	0.132083	0.122747	-0.002716	
medianmarechprebal30	-0.004835	0.004221	-0.000854	-0.000688	-0.001112	-0.001047	0.004216	
cnt_ma_rech90	0.245941	-0.003957	0.576787	0.582115	0.295746	0.329330	0.006131	
fr_ma_rech90	0.094709	0.005517	-0.061858	-0.063740	-0.022353	-0.024882	0.000881	
sumamnt_ma_rech90	0.212666	0.000160	0.754042	0.759865	0.324302	0.342772	0.002345	
medianamnt_ma_rech90	0.129527	0.005022	0.269721	0.262627	0.113115	0.106832	-0.001947	
medianmarechprebal90	0.041728	-0.001128	0.042276	0.041210	0.029945	0.032886	-0.001506	
cnt_da_rech30	0.004184	0.002445	0.000312	-0.000128	-0.001286	-0.001307	-0.003344	
fr_da_rech30	-0.000137	0.000806	-0.002442	-0.002189	-0.001917	-0.001997	-0.003469	
cnt_da_rech90	0.003601	0.000868	0.038944	0.031408	0.073169	0.057332	-0.003700	
fr_da_rech90	-0.005779	0.006379	0.019874	0.015944	0.047579	0.037829	-0.002232	
cnt_loans30	0.197565	-0.003157	0.346504	0.321006	0.162833	0.154900	0.002308	
amnt_loans30	0.199916	-0.003302	0.454169	0.430940	0.217586	0.216641	0.001031	
maxamnt_loans30	-0.000274	-0.003096	0.001569	0.001283	-0.001525	-0.002189	0.001681	
medianamnt_loans30	0.050067	0.004679	-0.005629	0.000012	-0.013746	-0.006703	0.002430	
cnt_loans90	0.004305	0.000192	0.008865	0.009220	0.003026	0.004301	-0.000216	
amnt_loans90	0.205065	-0.003336	0.542179	0.544854	0.280233	0.307920	0.000664	
maxamnt_loans90	0.086033	-0.000975	0.396803	0.394487	0.225449	0.241772	-0.003097	
medianamnt_loans90	0.041265	0.002346	-0.031485	-0.029046	-0.032555	-0.031045	0.003261	
payback30	0.050892	0.002246	0.033669	0.025432	0.075530	0.069847	-0.002857	
payback90	0.053776	0.002549	0.056822	0.050147	0.099533	0.104731	-0.001787	

33 rows × 33 columns

Observation:

daily_decr30 and daily_decr90 features are highly correlated with each otheer. rental30 and rental90 features are highly correlated with each other. cnt_loans30 and amount_loans30 columns are highly correlated with each other. amount_loans30 is also highly correlated with amount_loans90 column. medianamnt_loans30 and medianamnt_loans90 is highly correlated with each other. We have to drop one of the features which are highly correlated with other feayures. And if we dont do this then our model will face multicolinearity problem.

In [31]:
```
#Dropping the columns which is highly correlated with each other do avoid multicolinearity
df.drop(columns=['daily_decr30','rental30','amnt_loans30','medianamnt_loans30'], axis=1 ,
```

In [32]:
```
#Now checking the shape
print(df.shape)
#Checking the unique value in pdate column.
df['pdate'].nunique()
```

```
(186243, 31)
```

Out[32]:
```
82
```

In [33]:
```
#Making the new column Day, Month and year from pdate column
df['pDay']=pd.to_datetime(df['pdate'],format='%Y/%m/%d').dt.day
df['pMonth']=pd.to_datetime(df['pdate'],format='%Y/%m/%d').dt.month
df['pYear']=pd.to_datetime(df['pdate'],format='%Y/%m/%d').dt.year
```

In [34]:
```
df.head()
```

Out[34]:

	label	msisdn	aon	daily_decr90	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_
0	0	21408I70789	272.0	3065.150000	260.13	2.0	0.0	1539	
1	1	76462I70374	712.0	12124.750000	3691.26	20.0	0.0	5787	
2	1	17943I70372	535.0	1398.000000	900.13	3.0	0.0	1539	
3	1	55773I70781	241.0	21.228000	159.42	41.0	0.0	947	
4	1	03813I82730	947.0	150.619333	1098.90	4.0	0.0	2309	

5 rows × 34 columns

In [35]:
```
#Checking the number of months
df['pMonth'].unique()
```

Out[35]:
```
array([7, 8, 6], dtype=int64)
```

In [36]:
```
#After fetching the data from pdate column now we are going to drop it because it has not
df.drop(columns=['pdate'],axis=1, inplace = True)
```

In [37]:
```
#Seprate the categorical columns and Numerical columns
cat_df,num_df=[],[]

for i in df.columns:
```

```
    if df[i].dtype==object:
        cat_df.append(i)
    elif (df[i].dtypes=='int64') | (df[i].dtypes=='float64') | (df[i].dtypes=='int32'):
        num_df.append(i)
    else: continue

print('>>> Total Number of Feature::', df.shape[1])
print('>>> Number of categorical features::', len(cat_df))
print('>>> Number of Numerical Feature::', len(num_df))
```

```
>>> Total Number of Feature:: 33
>>> Number of categorical features:: 1
>>> Number of Numerical Feature:: 32
```

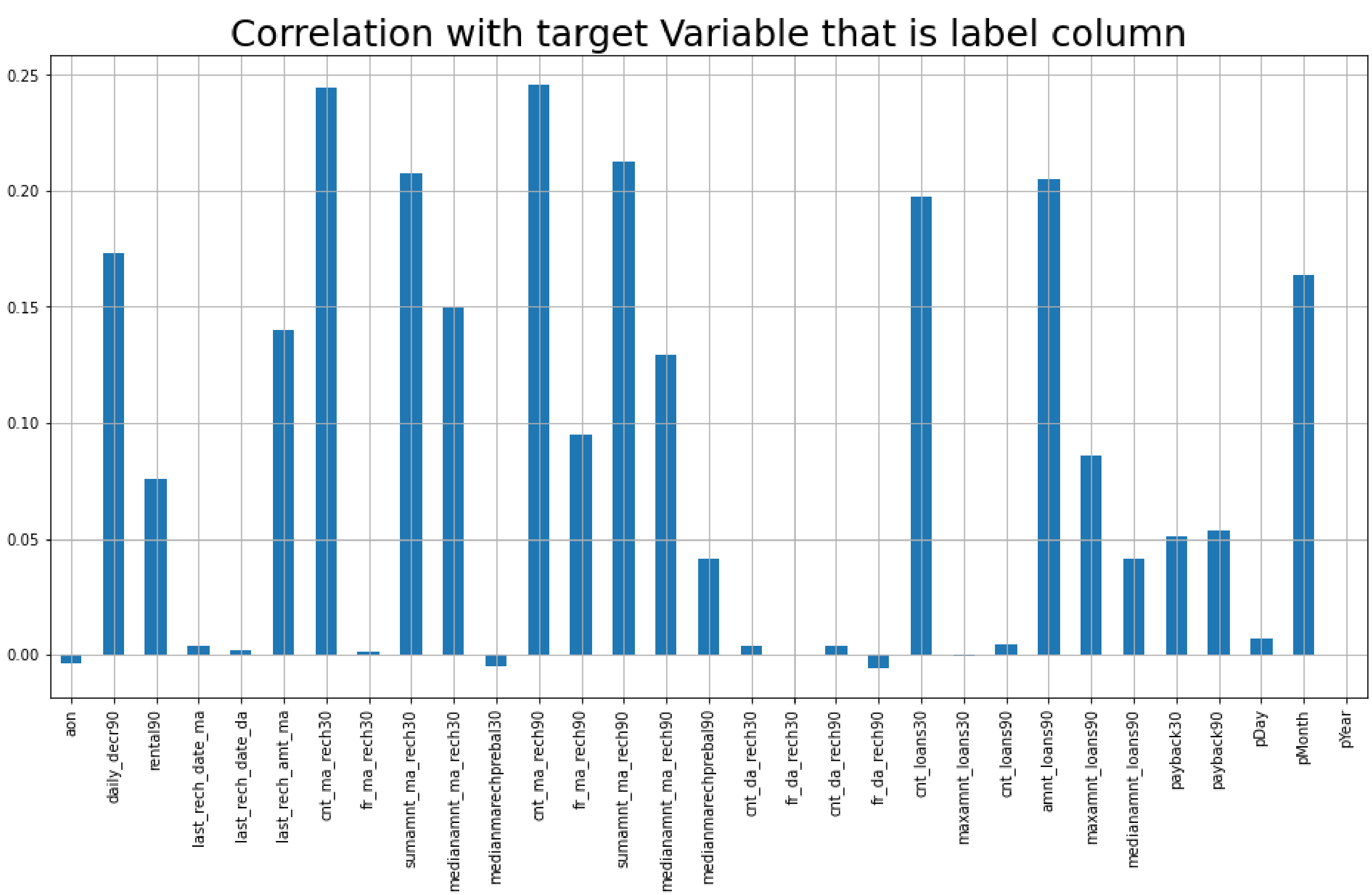
Data Visualization

In [38]:

```
#Checking the correlation with target variable
plt.figure(figsize=(16,8))
df.drop('label', axis=1).corrwith(df['label']).plot(kind='bar',grid=True)
plt.xticks(rotation='vertical')
plt.title("Correlation with target Variable that is label column",fontsize=25)
```

Out[38]:

```
Text(0.5, 1.0, 'Correlation with target Variable that is label column')
```



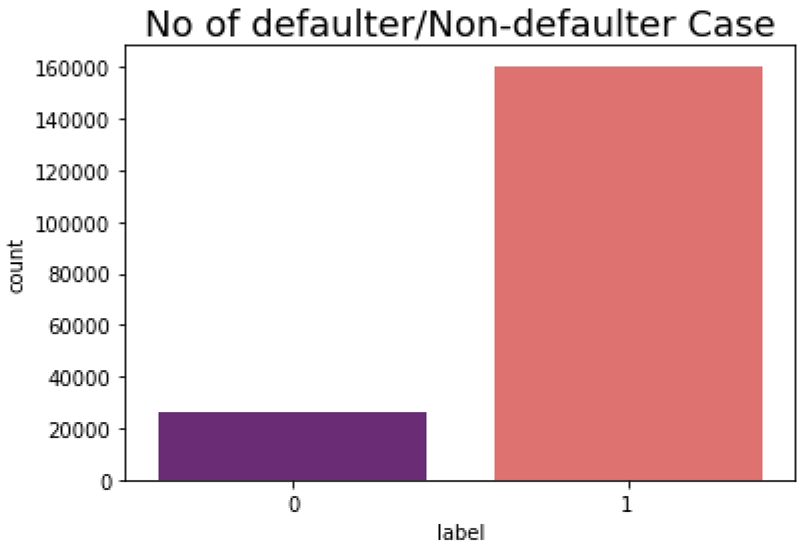
Observation:

Here we see the correlation of the columns with respect to the target column that is label.

In [39]:

```
#Checking the number of Fraud cases.
sns.countplot(x='label', data=df, palette='magma')
plt.title('No of defaulter/Non-defaulter Case',fontsize=18)
```

```
plt.show()

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




```
1    160383
0     25860
Name: label, dtype: int64
```

Observation:

Label 1 indicates loan has been payed i.e Non-Defaulter and label 0 indicates indicates that the loan has not beenpayed i.e. defaulter.

In [40]:

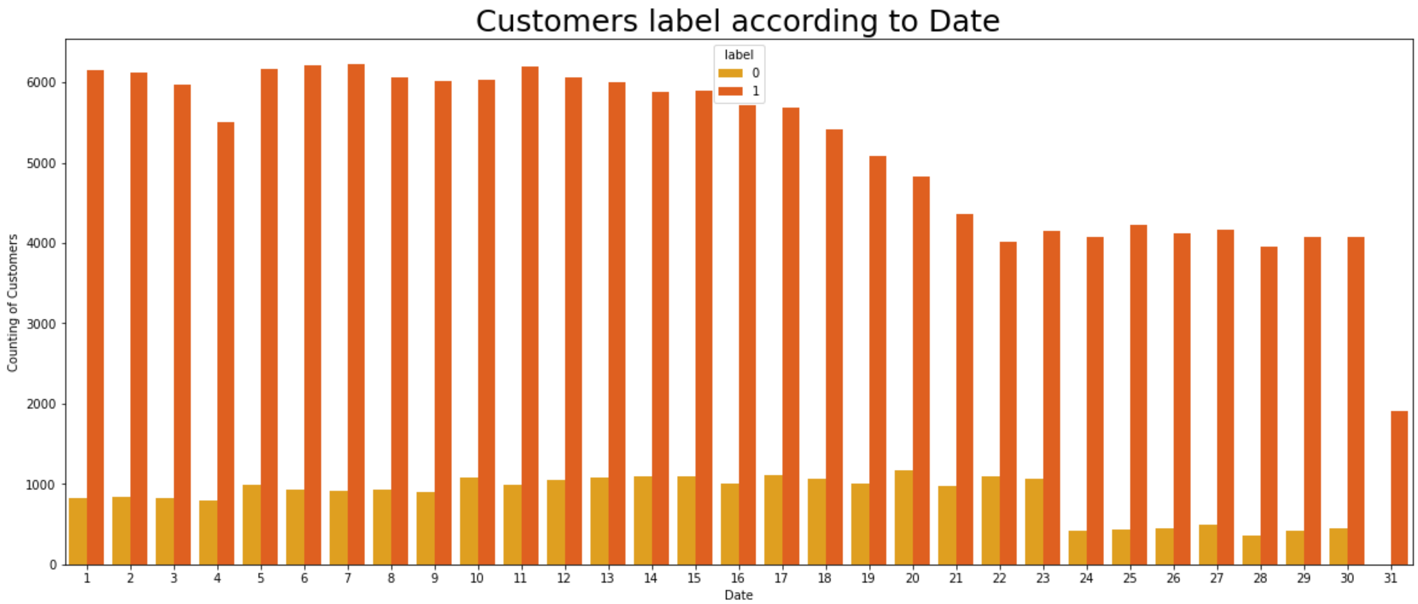
```
#Plotting the Histogram
df.hist(figsize=(20,20),color='r')
plt.show()
```

Observation:

We plot the histogram to display the shape and spread of continuous sample data.In a histogram, each bar groups numbers into ranges. Taller bars show that more data falls in that range

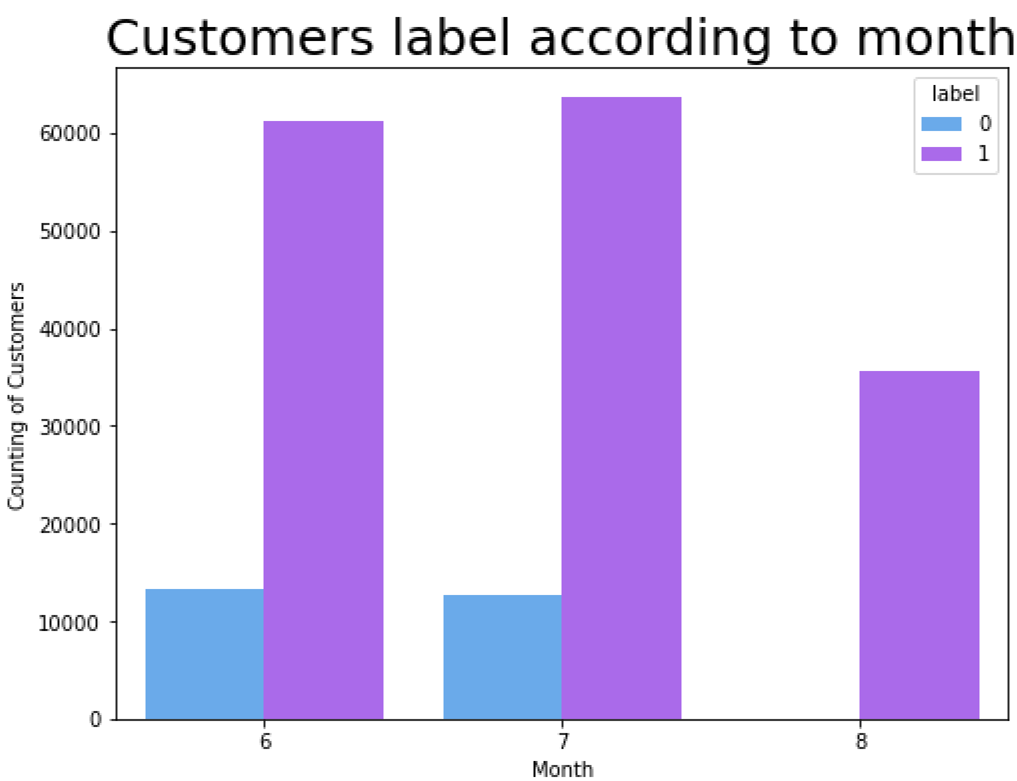
In [41]:

```
#Customer label according to Date
plt.figure(figsize=(20,8))
sns.countplot(x="pDay", hue='label', data=df, palette='autumn_r')
plt.title("Customers label according to Date", fontsize=25)
plt.xlabel('Date')
plt.ylabel('Counting of Customers')
plt.show()
```



In [42]:

```
#Customer label according to Month
plt.figure(figsize=(8,6))
sns.countplot(x="pMonth", hue='label', data=df, palette='cool')
plt.title("Customers label according to month", fontsize=25)
plt.xlabel('Month')
plt.ylabel('Counting of Customers')
plt.show()
```



Observation:

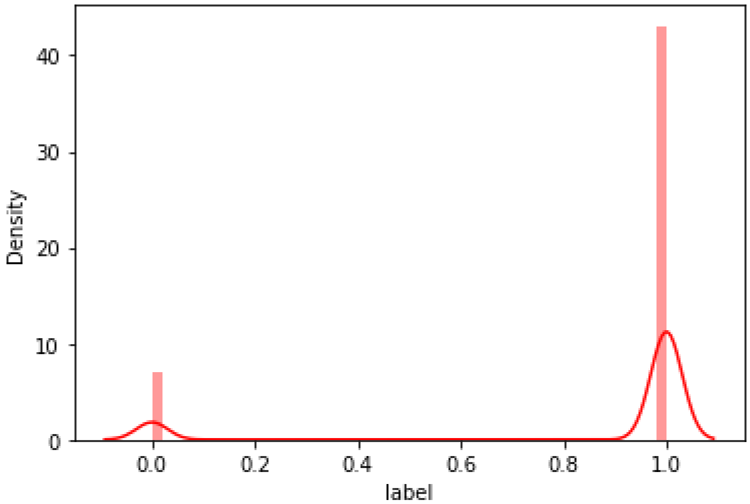
The first figure which is date vs label shows that the customers who did not pay their loans are from date 10 to 23. There are severals customers at June and July month who did not pay their loan.

In [43]:

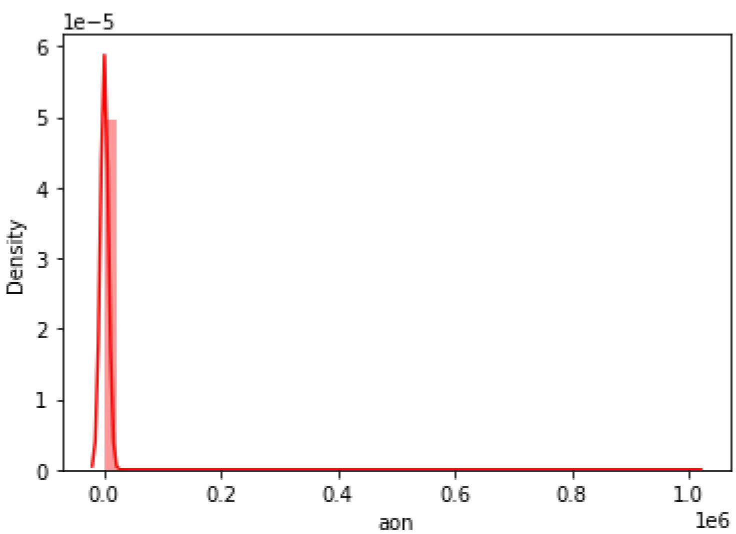
```
#checking skewness
```

```python
for col in df.describe().columns:
    sns.distplot(df[col],color='r')
    plt.show()
```

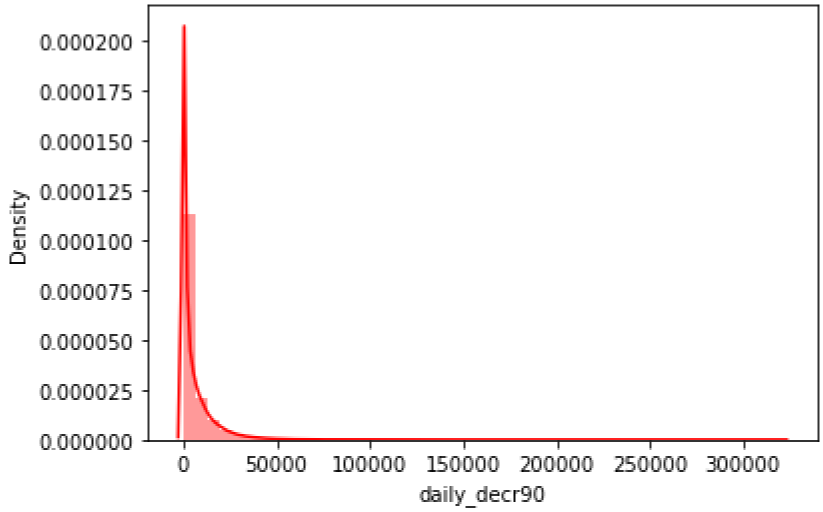
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



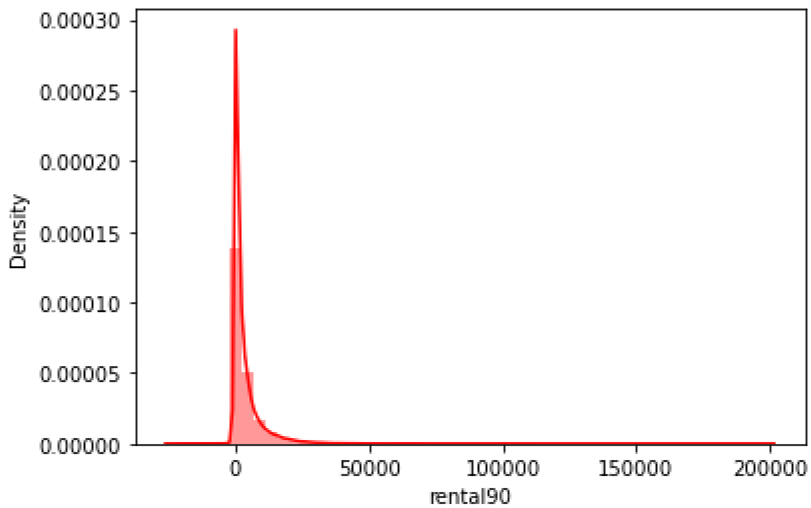

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



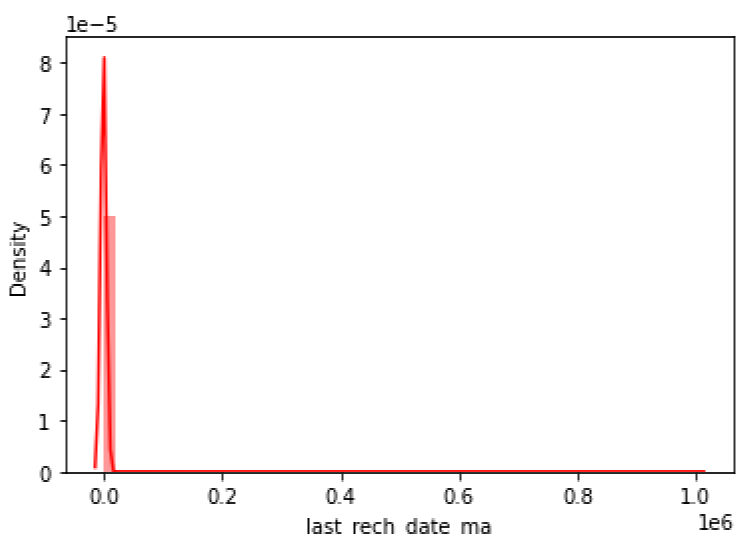

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

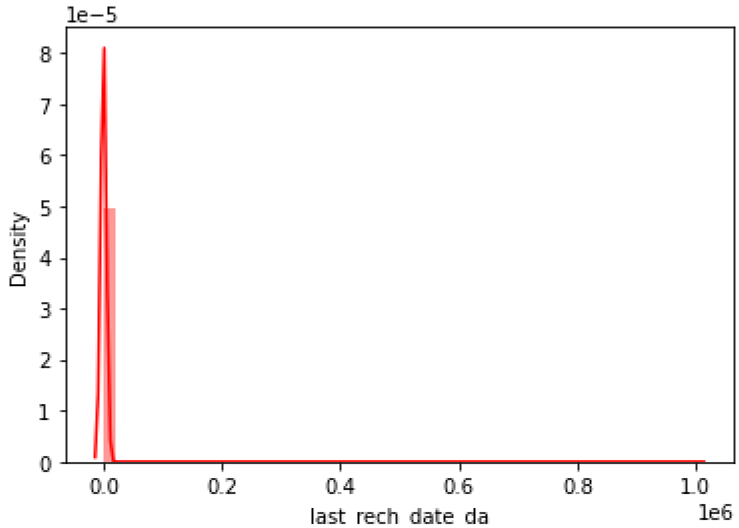



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

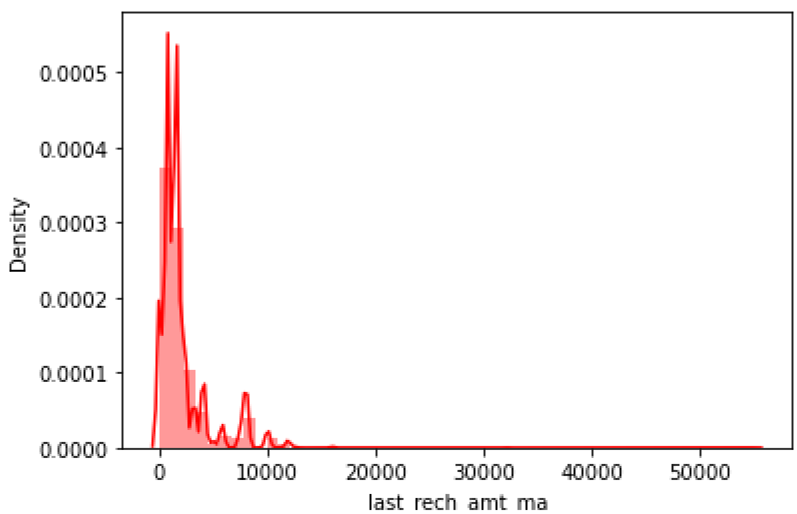



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
```

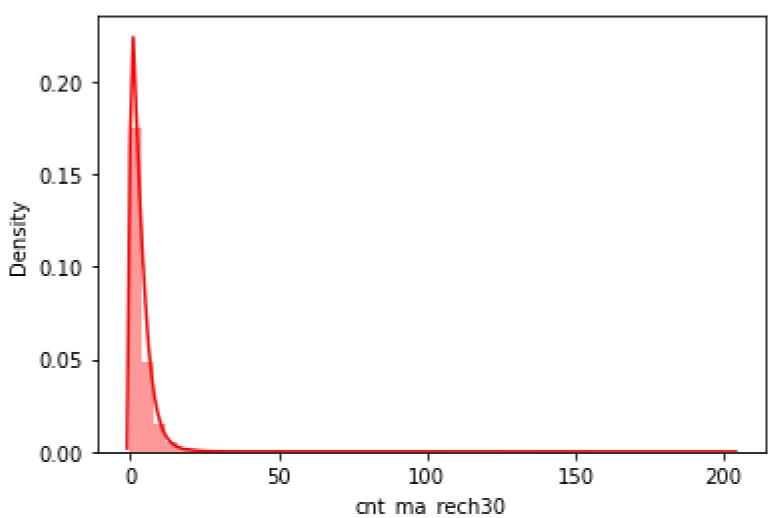
```
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

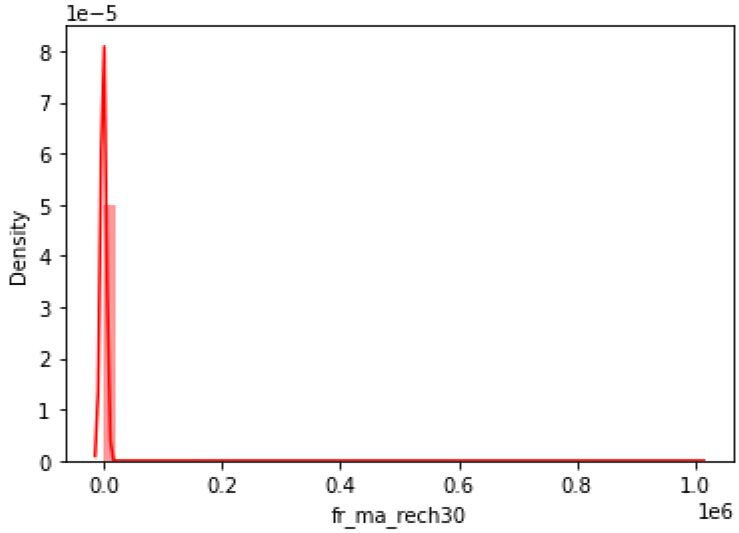



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

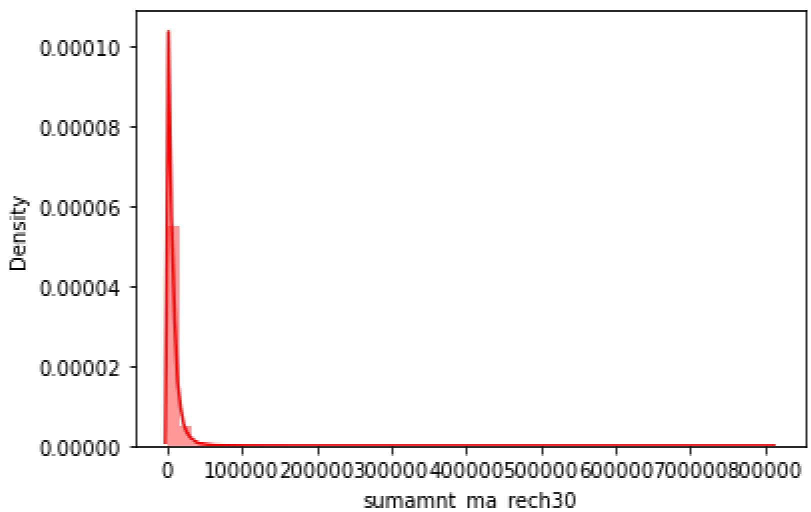



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
```

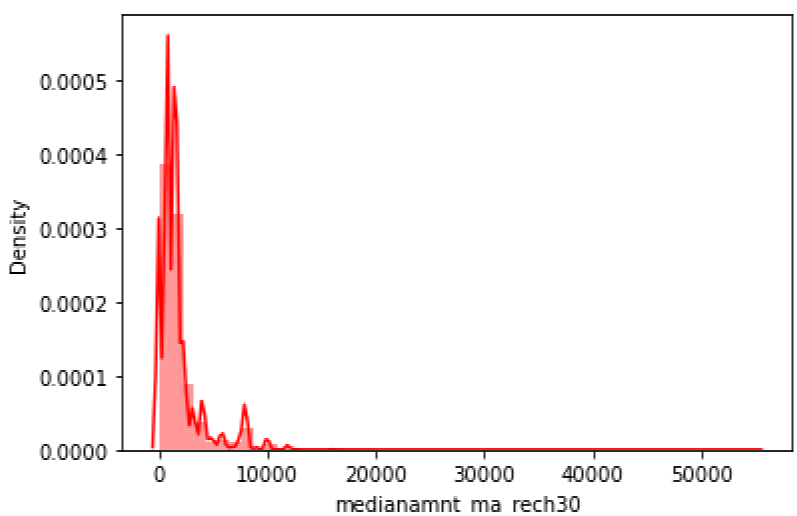
```
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

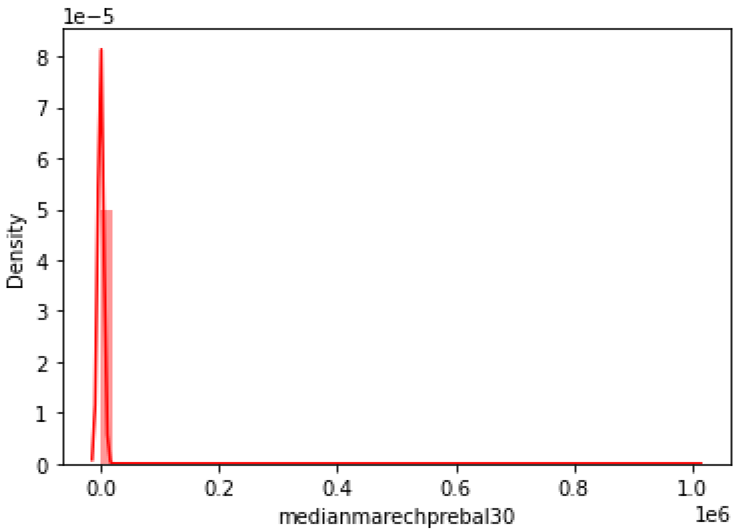



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

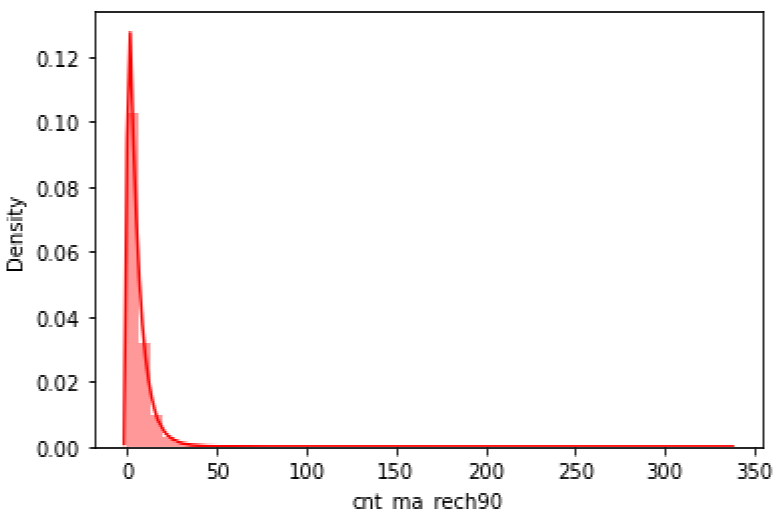



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
```

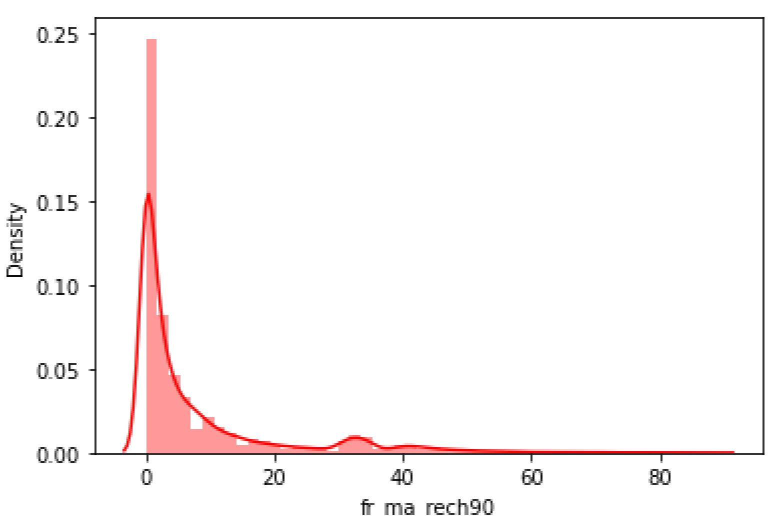
```
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



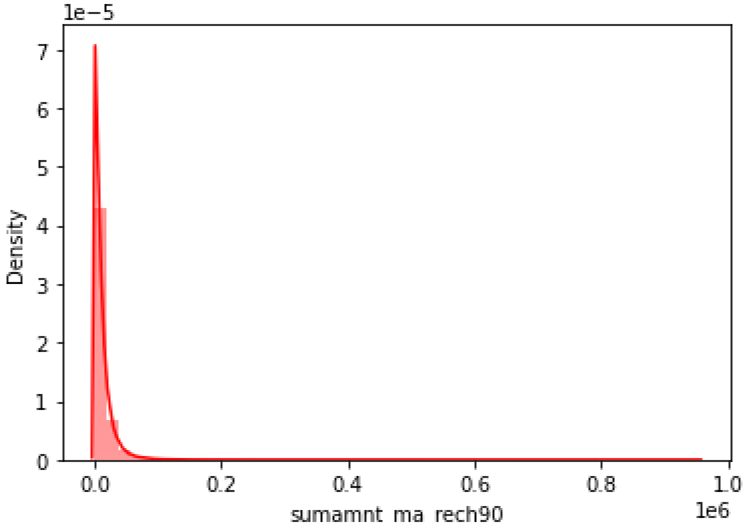

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



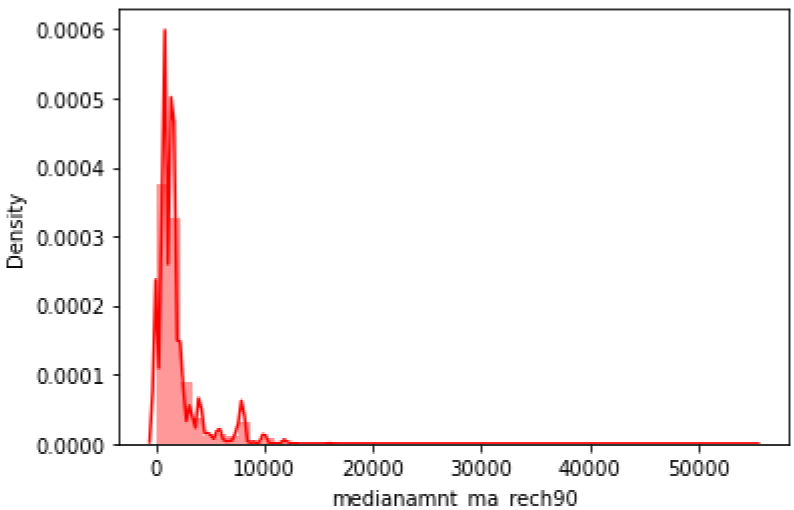

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



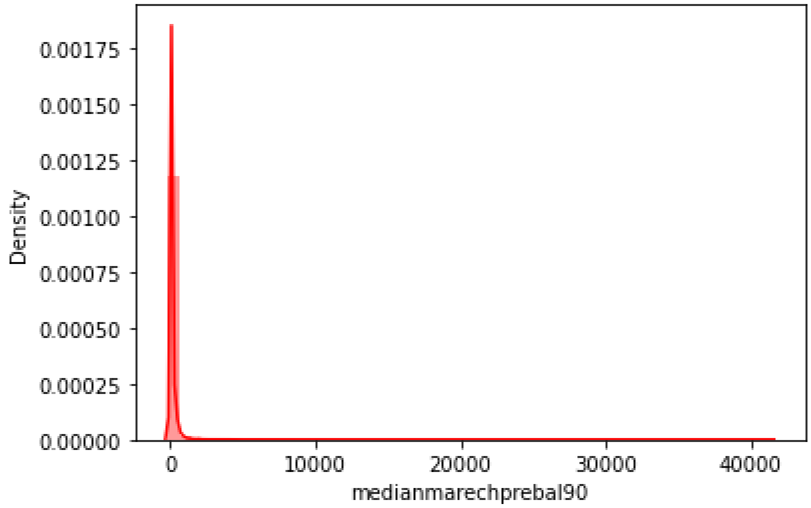
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



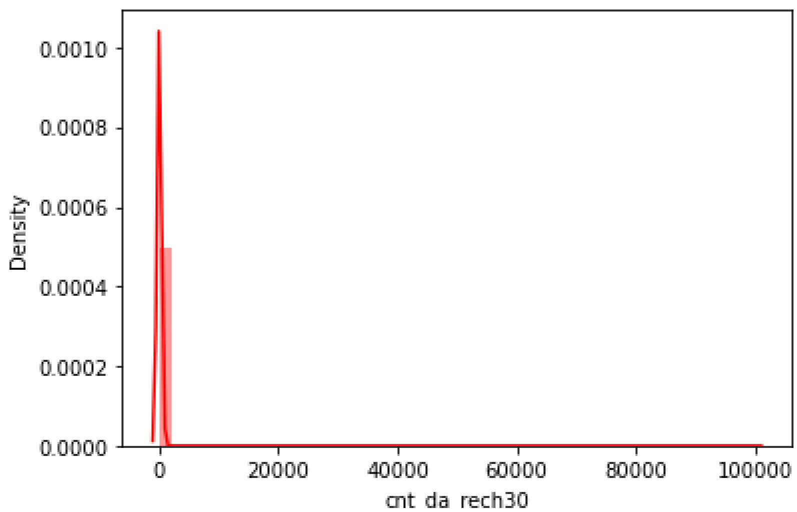

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



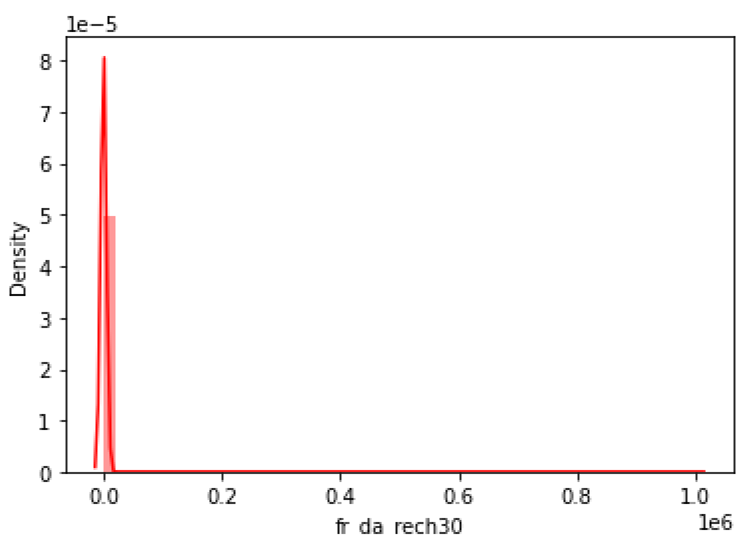

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

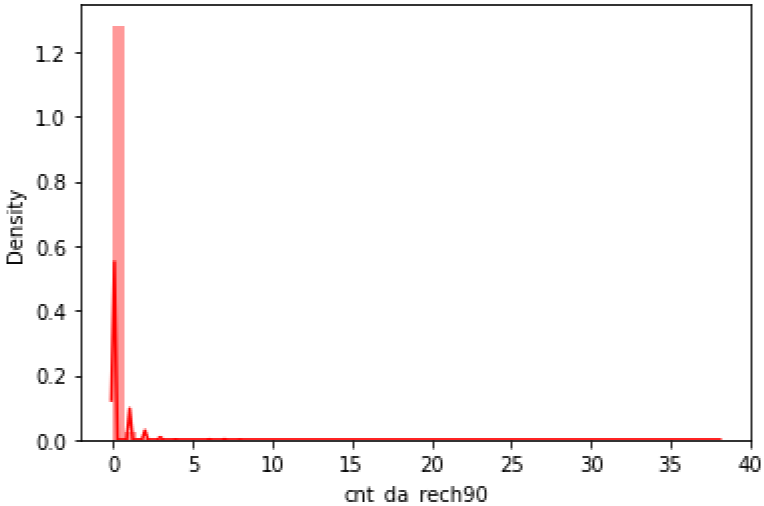



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

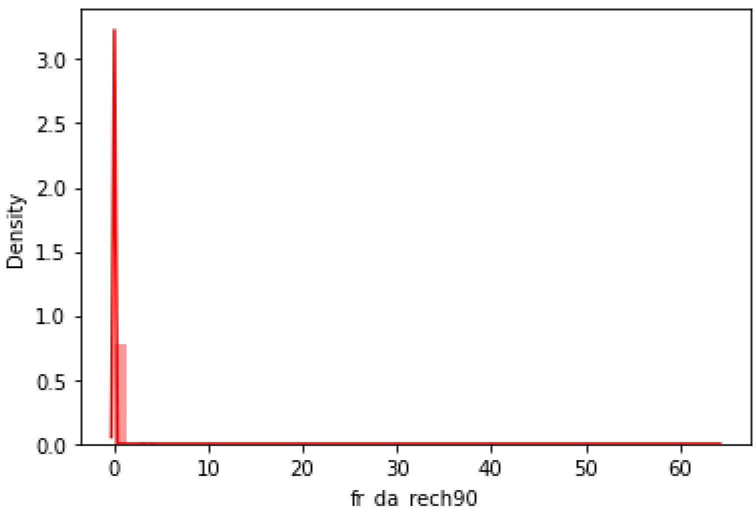



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
```

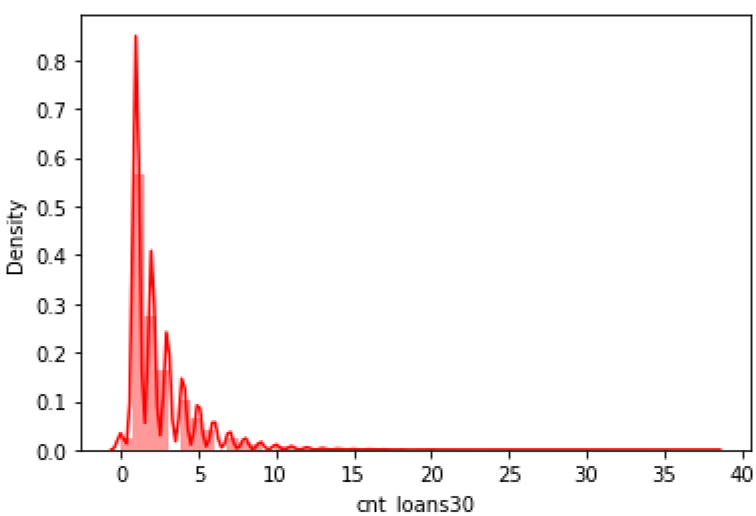
```
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

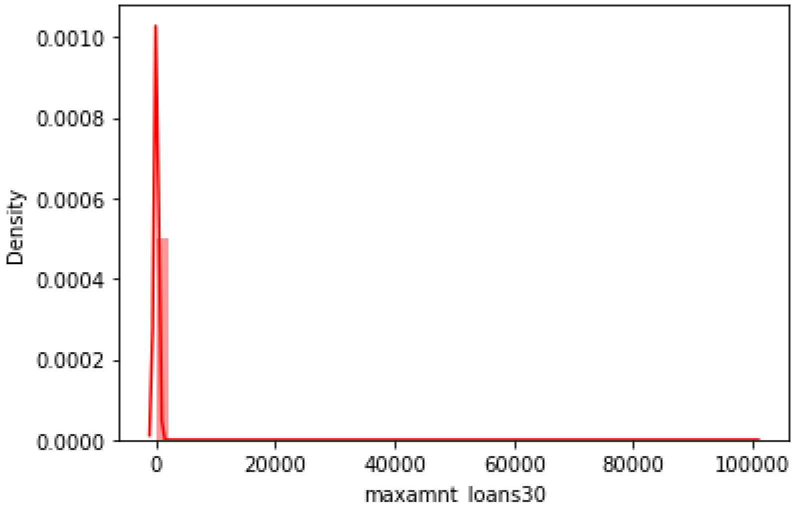



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

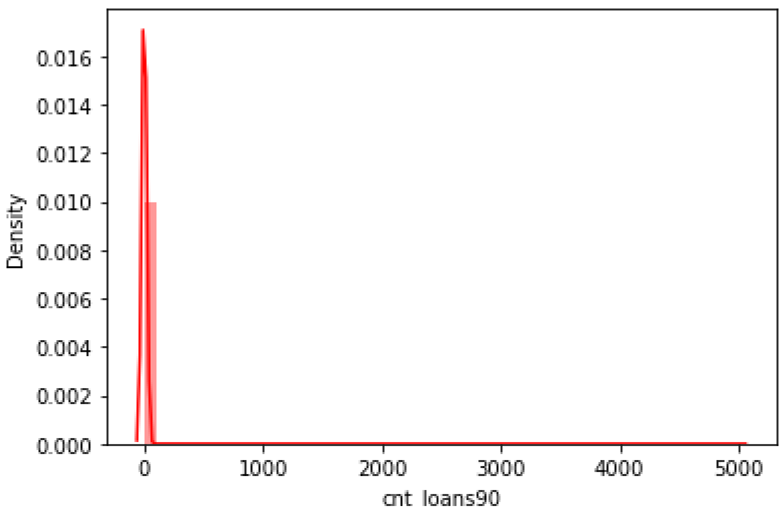



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
```

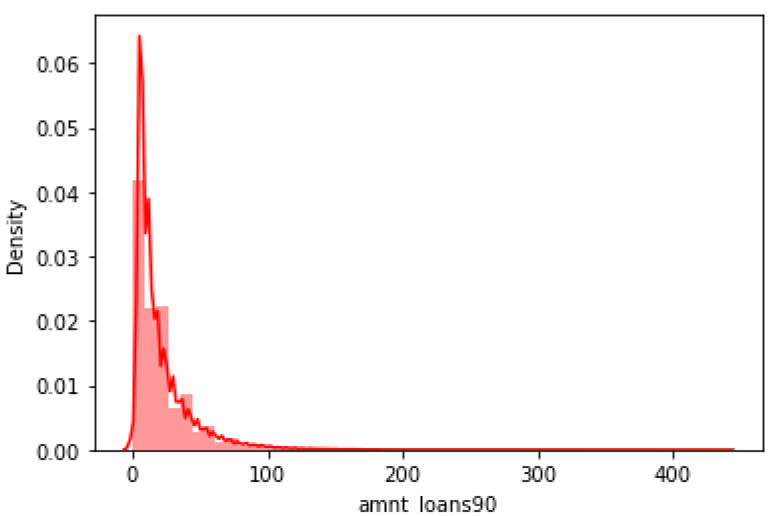
```
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

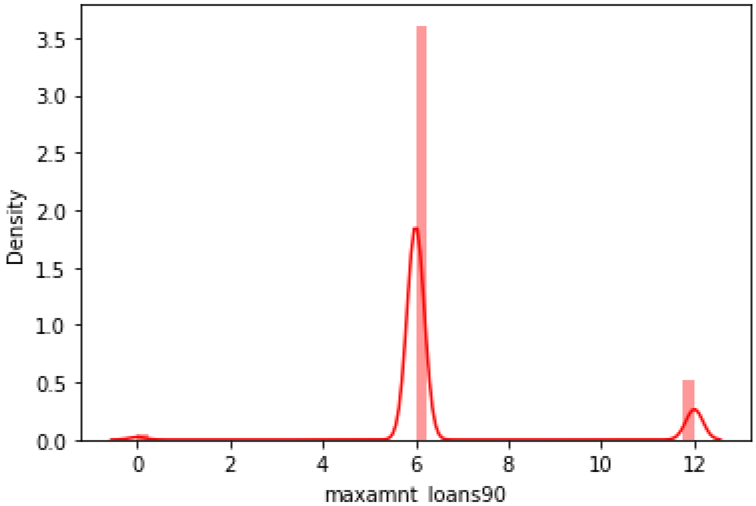



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

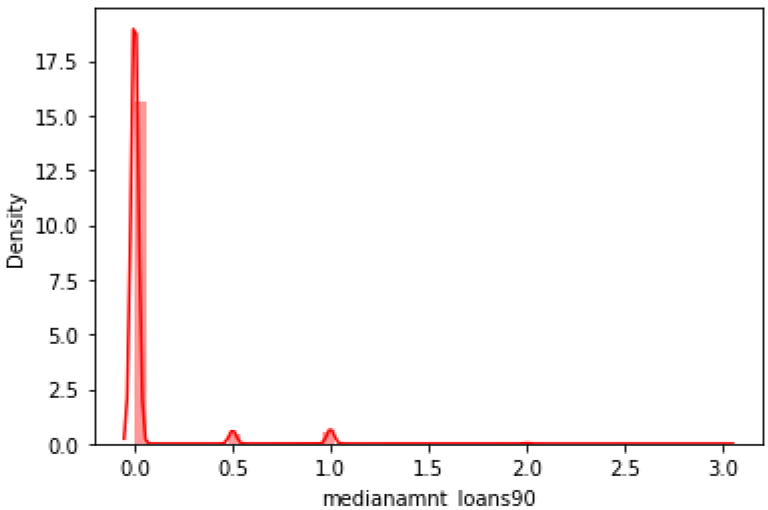



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
```

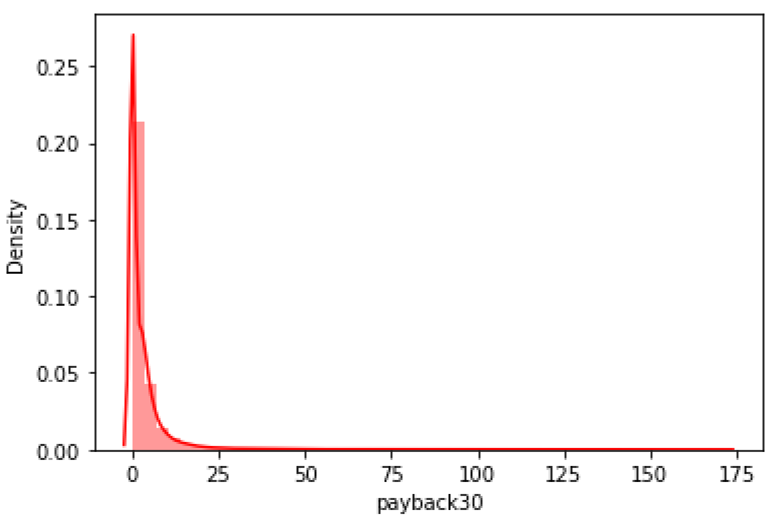
```
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

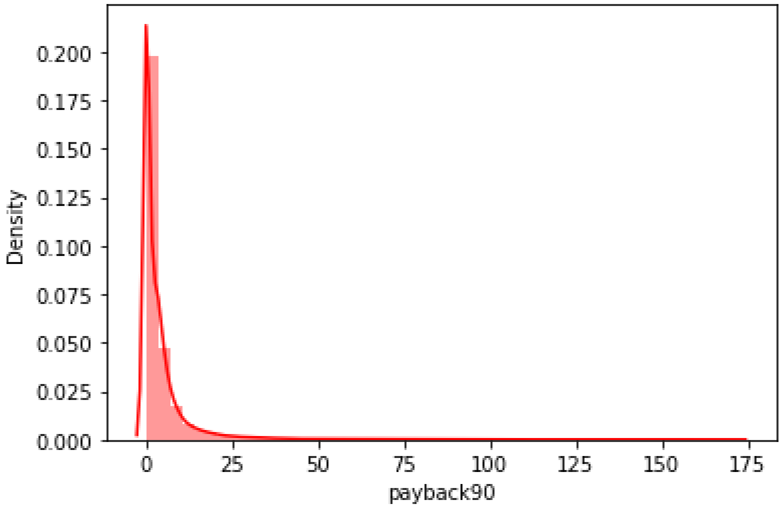



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

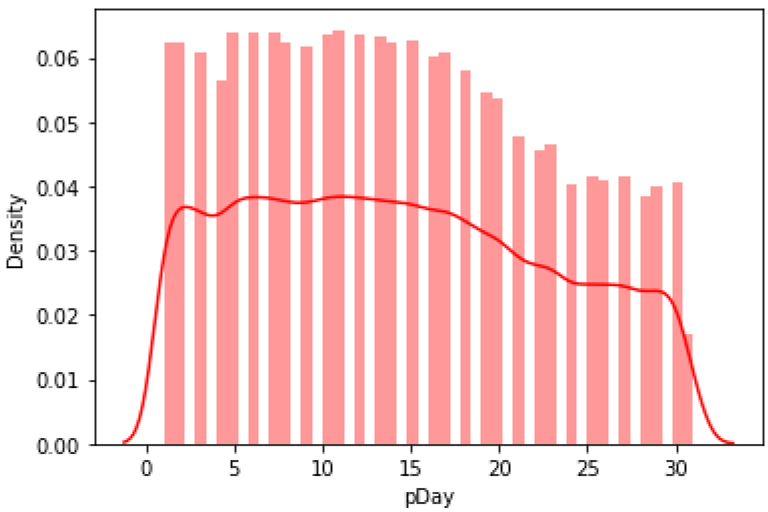



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
```

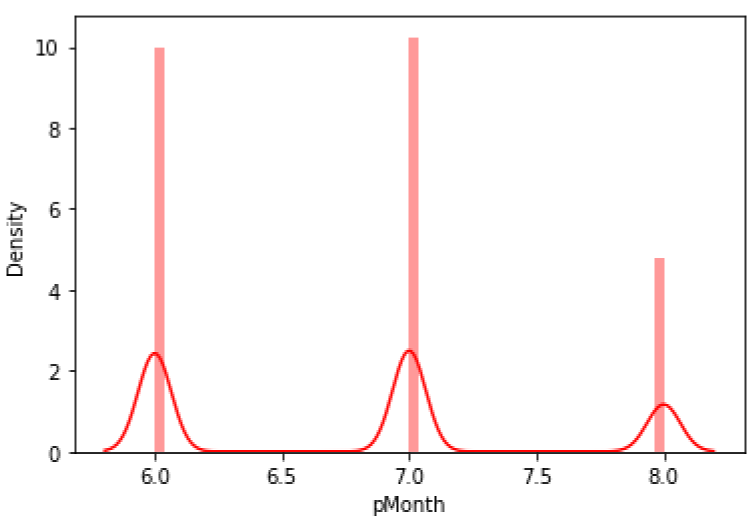
```
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

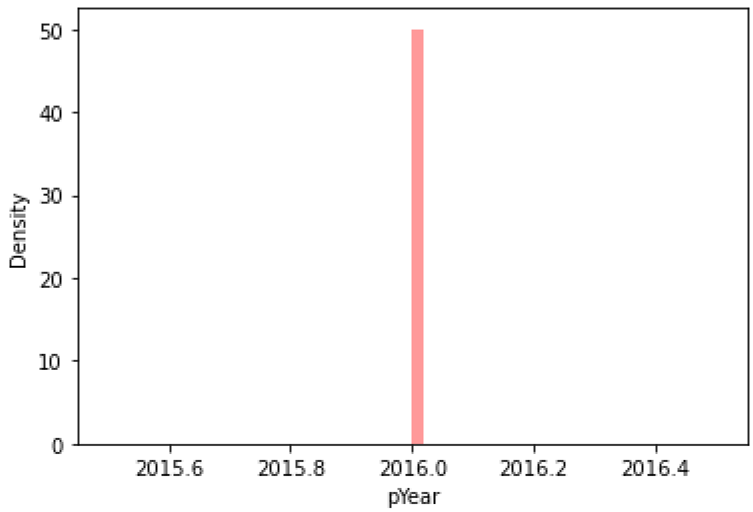



```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
```

```
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:316: UserWarning: Data
set has 0 variance; skipping density estimate. Pass `warn_singular=False` to disable this
warning.
  warnings.warn(msg, UserWarning)
```



In [44]:

```
df.skew()
```

```
C:\Users\Arun\AppData\Local\Temp/ipykernel_12624/1665899112.py:1: FutureWarning: Dropping
of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a
future version this will raise TypeError.  Select only valid columns before calling the re
duction.
  df.skew()
```

Out[44]:

```
label                   -2.088847
aon                     10.365026
daily_decr90             4.301490
rental90                 4.530925
last_rech_date_ma       14.852116
last_rech_date_da       14.781824
last_rech_amt_ma         3.830612
cnt_ma_rech30            3.471313
fr_ma_rech30            14.822224
sumamnt_ma_rech30        7.134012
medianamnt_ma_rech30     3.519213
medianmarechprebal30    14.677544
cnt_ma_rech90            3.558616
fr_ma_rech90             2.250443
sumamnt_ma_rech90        5.231693
medianamnt_ma_rech90     3.753115
medianmarechprebal90    43.576364
cnt_da_rech30           17.749485
fr_da_rech30            14.728609
cnt_da_rech90           28.396293
fr_da_rech90            28.959851
cnt_loans30              2.737584
maxamnt_loans30         17.718074
cnt_loans90             16.717192
amnt_loans90             3.165962
maxamnt_loans90          1.650198
medianamnt_loans90       4.774958
payback30                8.193009
payback90                6.763241
pDay                     0.200706
pMonth                   0.351293
pYear                    0.000000
dtype: float64
```

In [45]:
```
#Treating Skewness via square root method.
#df.skew()
#for col in df.skew().index:
    #if col in df.describe().columns:
        #if df[col].skew()>0.55:
            #df[col]=np.sqrt(df[col])
```

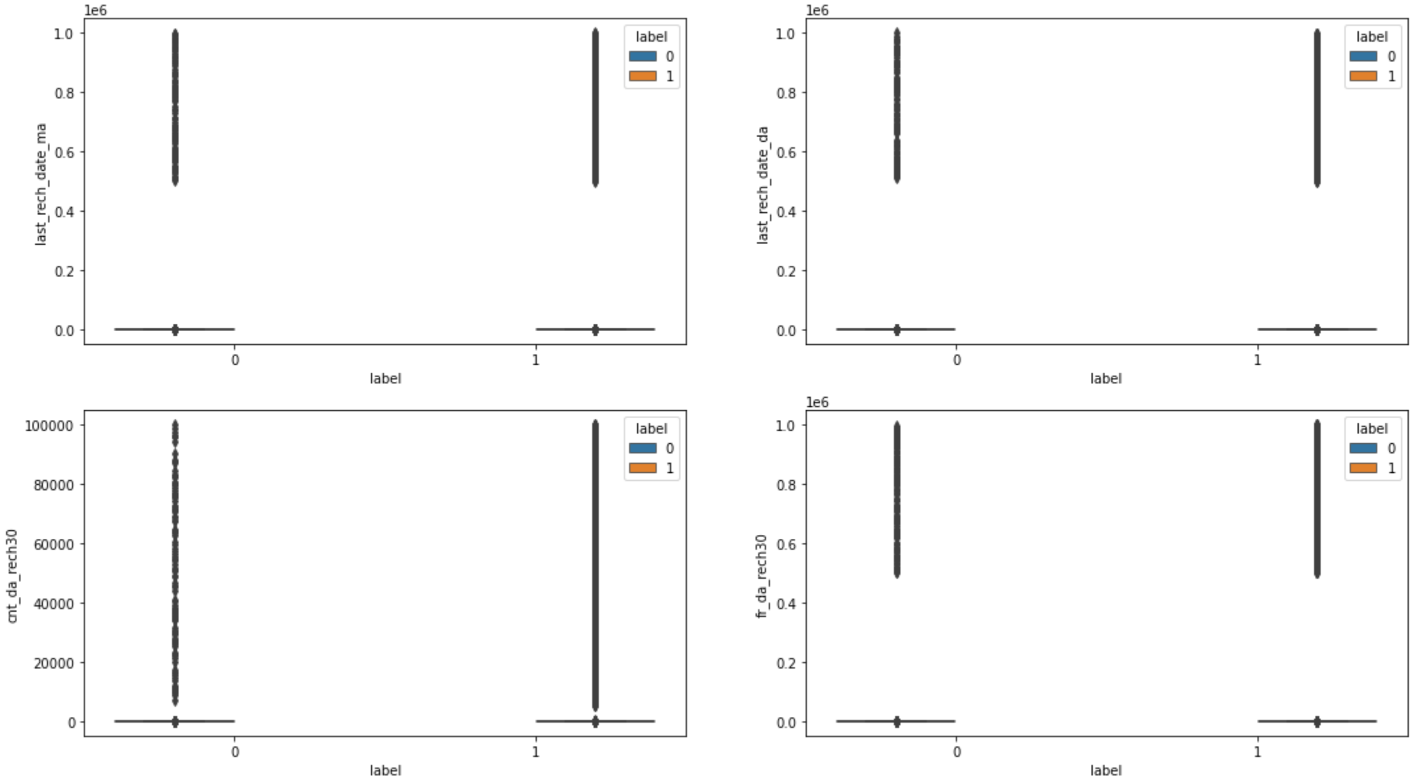
In [46]:
```
df.skew()
```

```
C:\Users\Arun\AppData\Local\Temp/ipykernel_12624/547062910.py:1: FutureWarning: Dropping o
f nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a
future version this will raise TypeError.  Select only valid columns before calling the re
duction.
  df.skew()
```

Out[46]:
```
label                   -2.088847
aon                     10.365026
daily_decr90             4.301490
rental90                 4.530925
last_rech_date_ma       14.852116
last_rech_date_da       14.781824
last_rech_amt_ma         3.830612
cnt_ma_rech30            3.471313
fr_ma_rech30            14.822224
sumamnt_ma_rech30        7.134012
medianamnt_ma_rech30     3.519213
medianmarechprebal30    14.677544
cnt_ma_rech90            3.558616
fr_ma_rech90             2.250443
sumamnt_ma_rech90        5.231693
medianamnt_ma_rech90     3.753115
medianmarechprebal90    43.576364
cnt_da_rech30           17.749485
fr_da_rech30            14.728609
cnt_da_rech90           28.396293
fr_da_rech90            28.959851
cnt_loans30              2.737584
maxamnt_loans30         17.718074
cnt_loans90             16.717192
amnt_loans90             3.165962
maxamnt_loans90          1.650198
medianamnt_loans90       4.774958
payback30                8.193009
payback90                6.763241
pDay                     0.200706
pMonth                   0.351293
pYear                    0.000000
dtype: float64
```

In [47]:
```
#plotting outliers

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2, figsize = (18, 10))
sns.boxplot(ax=ax1, x = 'label', y = 'last_rech_date_ma', hue = 'label', data = df)
sns.boxplot(ax=ax2, x = 'label', y = 'last_rech_date_da', hue = 'label', data = df)
sns.boxplot(ax=ax3, x = 'label', y = 'cnt_da_rech30', hue = 'label', data = df)
sns.boxplot(ax=ax4, x = 'label', y = 'fr_da_rech30', hue = 'label', data = df)
```

Out[47]:
```
<AxesSubplot:xlabel='label', ylabel='fr_da_rech30'>
```



Observation:

There are too many outliers present in our dataset.So we need to remove it. But before removing please check that only 8 to 10% of data removed.

In [48]:

```
#Creating a copy of our dataset
df2=df1.copy()
#Dropping the object columns
df1.drop(columns=['msisdn','pdate'],axis=1,inplace=True)
```

In [49]:

```
df1.columns
```

Out[49]:

```
Index(['label', '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'],
      dtype='object')
```

In [50]:

```
from scipy.stats import zscore
z=np.abs(zscore(df1))
z
```

Out[50]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last
0	2.647896	0.103577	0.252299	0.276346	0.573844	0.558583	0.069637	0.069550	
1	0.377658	0.097764	0.731037	0.553380	0.231788	0.036020	0.069303	0.069550	
2	0.377658	0.100102	0.432011	0.429033	0.416020	0.447674	0.069619	0.069550	

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last
3	0.377658	0.103986	0.581326	0.555125	0.587935	0.576036	0.068914	0.069550	
4	0.377658	0.094660	0.567293	0.543274	0.369886	0.413227	0.069600	0.069550	
...	...	...	...	...	...	...	...	...	
209588	0.377658	0.101833	0.567157	0.543159	0.372140	0.414910	0.069656	0.069550	
209589	0.377658	0.092969	0.579622	0.553686	0.223791	0.304144	0.069600	0.069550	
209590	0.377658	0.093788	0.700790	0.533194	0.735567	0.937500	0.069619	0.069550	
209591	0.377658	0.084289	0.770755	0.594558	0.529352	0.433039	0.069637	0.068838	
209592	0.377658	0.086284	0.096744	0.141746	0.512620	0.494278	0.069433	0.069550	

209593 rows × 33 columns

In [51]:

```
threshold=3
print(np.where(z>3))
```

```
(array([    21,     22,     22, ..., 209586, 209587, 209587], dtype=int64), array([15, 15,
32, ..., 28, 26, 30], dtype=int64))
```

In [52]:

```
df1_new=df1[(z<3).all(axis=1)]
```

In [53]:

```
#Checking the shape
print(df1.shape,'\t\t',df1_new.shape)
```

```
(209593, 33)             (161465, 33)
```

In [55]:

```
#Converting the categorical data into numeric variables
# Transform Non numeric columns into Numeric columns

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

for column in df.columns:
    if df[column].dtype==np.number:
        continue
    df[column]=le.fit_transform(df[column])
```

```
C:\Users\Arun\AppData\Local\Temp/ipykernel_12624/2334802243.py:9: DeprecationWarning: Conv
erting `np.inexact` or `np.floating` to a dtype is deprecated. The current result is `floa
t64` which is not strictly correct.
  if df[column].dtype==np.number:
```

In [56]:

```
df.head()
```

Out[56]:

	label	msisdn	aon	daily_decr90	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_rech3
0	0	40191	272.0	3065.150000	260.13	2.0	0.0	14	
1	1	142291	712.0	12124.750000	3691.26	20.0	0.0	38	
2	1	33594	535.0	1398.000000	900.13	3.0	0.0	14	
3	1	104157	241.0	21.228000	159.42	41.0	0.0	10	

	label	msisdn	aon	daily_decr90	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_rech
4	1	6910	947.0	150.619333	1098.90	4.0	0.0	23	

5 rows × 33 columns

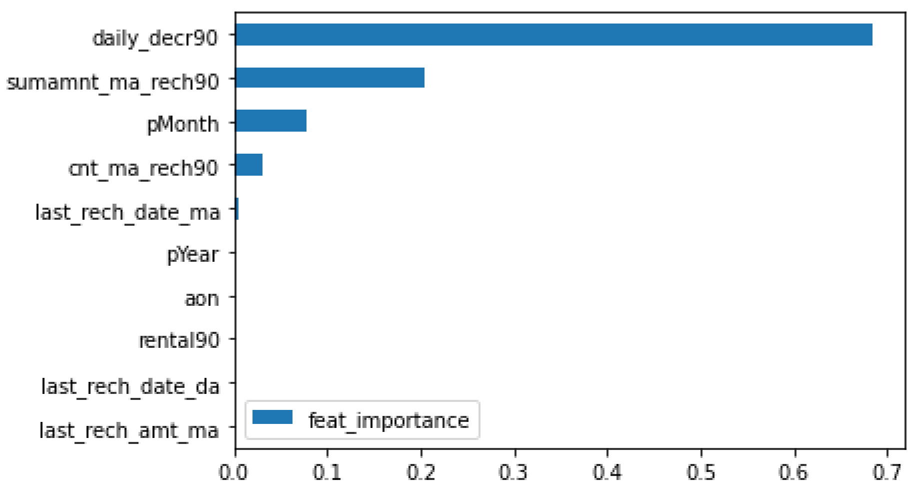
Feature importance

In [57]:
```
#Splitting the data into x and y

x = df.drop(['label'], axis=1)

y = df['label']
```

In [58]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(max_depth=3)
dt.fit(x, y)
```

Out[58]:
```
DecisionTreeClassifier(max_depth=3)
```

In [59]:
```
dt_features = pd.DataFrame(dt.feature_importances_, index=x.columns, columns=['feat_import
dt_features.sort_values('feat_importance').tail(10).plot.barh()
plt.show()
```



By looking at the daily_decr90 which is Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah), it seems that this feature helps to discriminate the data indeed. This feature can bring insights for company when analyzing a customers.

Model Training

In [60]:
```
#Scaling in input variables
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x=ss.fit_transform(x)
```

In [61]:
```
#Splitting the data into training and testing data
```

```
from sklearn.model_selection import train_test_split,cross_val_score
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=42,stratify=
```

In [62]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
```

In [63]:

```
KNN=KNeighborsClassifier(n_neighbors=10)
LR=LogisticRegression()
DT=DecisionTreeClassifier(random_state=20)
GNB=GaussianNB()
RF=RandomForestClassifier()
```

In [64]:

```
models = []
models.append(('KNeighborsClassifier', KNN))
models.append(('LogisticRegression', LR))
models.append(('DecisionTreeClassifier',DT))
models.append(('GaussianNB', GNB))
models.append(('RandomForestClassifier', RF))
```

In [65]:

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curv
```

In [67]:

```
Model=[]
score=[]
cvs=[]
rocscore=[]
for name,model in models:
    print('****************************',name,'*********************************')
    print('\n')
    Model.append(name)
    model.fit(x_train,y_train.values.ravel())
    print(model)
    pre=model.predict(x_test)
    print('\n')
    AS=accuracy_score(y_test,pre)
    print('Accuracy_score = ', AS)
    score.append(AS*100)
    print('\n')
    sc=cross_val_score(model,x,y,cv=10,scoring='accuracy').mean()
    print('Cross_val_Score = ', sc)
    cvs.append(sc*100)
    print('\n')
    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,pre)
    roc_auc= auc(false_positive_rate, true_positive_rate)
    print('roc_auc_score = ',roc_auc)
    rocscore.append(roc_auc*100)
    print('\n')
    print('classification_report\n',classification_report(y_test,pre))
    print('\n')
    cm=confusion_matrix(y_test,pre)
    print(cm)
    print('\n')
    plt.figure(figsize=(10,40))
    plt.subplot(911)
    plt.title(name)
    print(sns.heatmap(cm,annot=True))
    plt.subplot(912)
```

```
    plt.title(name)
    plt.plot(false_positive_rate, true_positive_rate, label = 'AUC= %0.2f'%roc_auc)
    plt.legend(loc='lower right')
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    print('\n\n')
```

```
**************************** KNeighborsClassifier ********************************


KNeighborsClassifier(n_neighbors=10)


Accuracy_score =  0.8699025477194019


Cross_val_Score =  0.8713937870453654


roc_auc_score =  0.6867161965572931


classification_report
               precision    recall  f1-score   support

           0       0.54      0.43      0.48      5172
           1       0.91      0.94      0.93     32077

    accuracy                           0.87     37249
   macro avg       0.73      0.69      0.70     37249
weighted avg       0.86      0.87      0.86     37249



[[ 2240  2932]
 [ 1914 30163]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)



**************************** LogisticRegression ********************************


LogisticRegression()


Accuracy_score =  0.8642379661198958


Cross_val_Score =  0.8642364984778247


roc_auc_score =  0.5250645042510697


classification_report
               precision    recall  f1-score   support

           0       0.63      0.06      0.10      5172
           1       0.87      0.99      0.93     32077

    accuracy                           0.86     37249
   macro avg       0.75      0.53      0.51     37249
```

```
weighted avg       0.83      0.86      0.81     37249


[[  287  4885]
 [  172 31905]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)


**************************** DecisionTreeClassifier ********************************


DecisionTreeClassifier(random_state=20)


Accuracy_score =  0.8717549464415152


Cross_val_Score =  0.8746583457298369


roc_auc_score =  0.740822571393308


classification_report
               precision    recall  f1-score   support

           0       0.54      0.56      0.55      5172
           1       0.93      0.92      0.93     32077

    accuracy                           0.87     37249
   macro avg       0.73      0.74      0.74     37249
weighted avg       0.87      0.87      0.87     37249


[[ 2894  2278]
 [ 2499 29578]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)


**************************** GaussianNB ********************************


GaussianNB()


Accuracy_score =  0.6136272114687643


Cross_val_Score =  0.6083770543601099


roc_auc_score =  0.717201145874796


classification_report
               precision    recall  f1-score   support

           0       0.25      0.86      0.38      5172
```

```
           1       0.96      0.57      0.72     32077

    accuracy                           0.61     37249
   macro avg       0.60      0.72      0.55     37249
weighted avg       0.86      0.61      0.67     37249
```

```
[[ 4451   721]
 [13671 18406]]
```

```
AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

```
*************************** RandomForestClassifier *******************************
```

```
RandomForestClassifier()
```

```
Accuracy_score =  0.9133936481516283
```

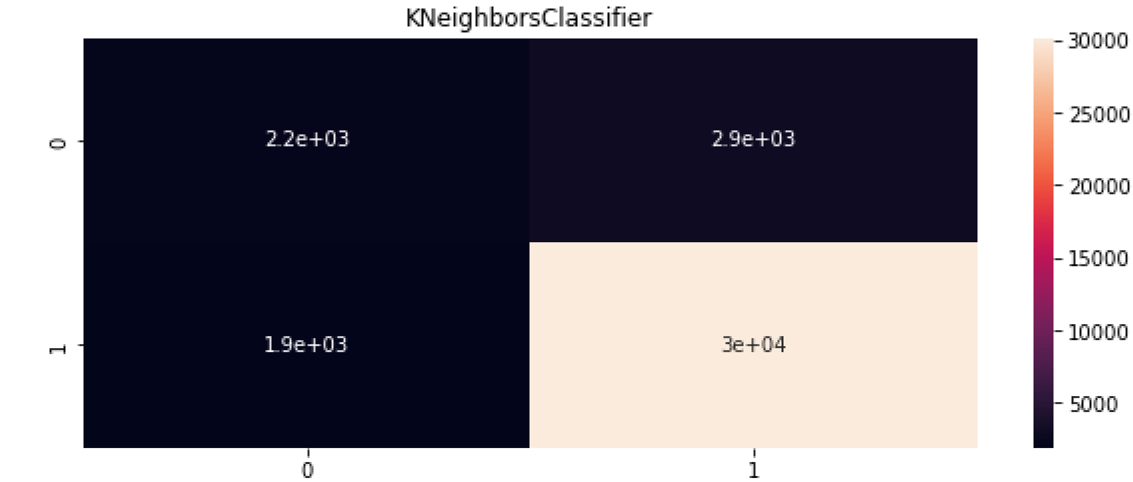
```
Cross_val_Score =  0.9134141830415832
```

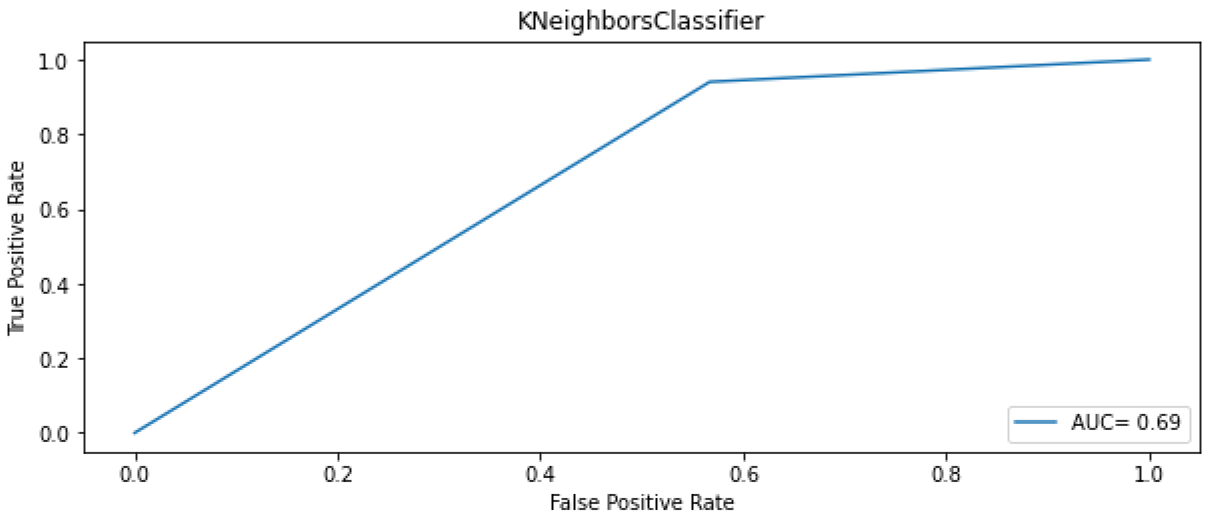
```
roc_auc_score =  0.7452946537600781
```

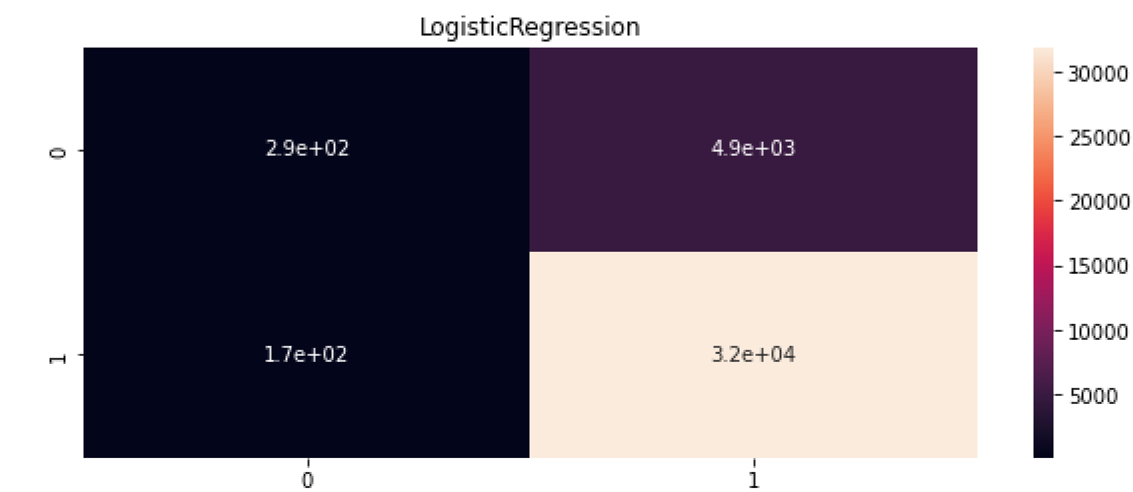
```
classification_report
              precision    recall  f1-score   support

           0       0.79      0.51      0.62      5172
           1       0.93      0.98      0.95     32077

    accuracy                           0.91     37249
   macro avg       0.86      0.75      0.79     37249
weighted avg       0.91      0.91      0.91     37249
```

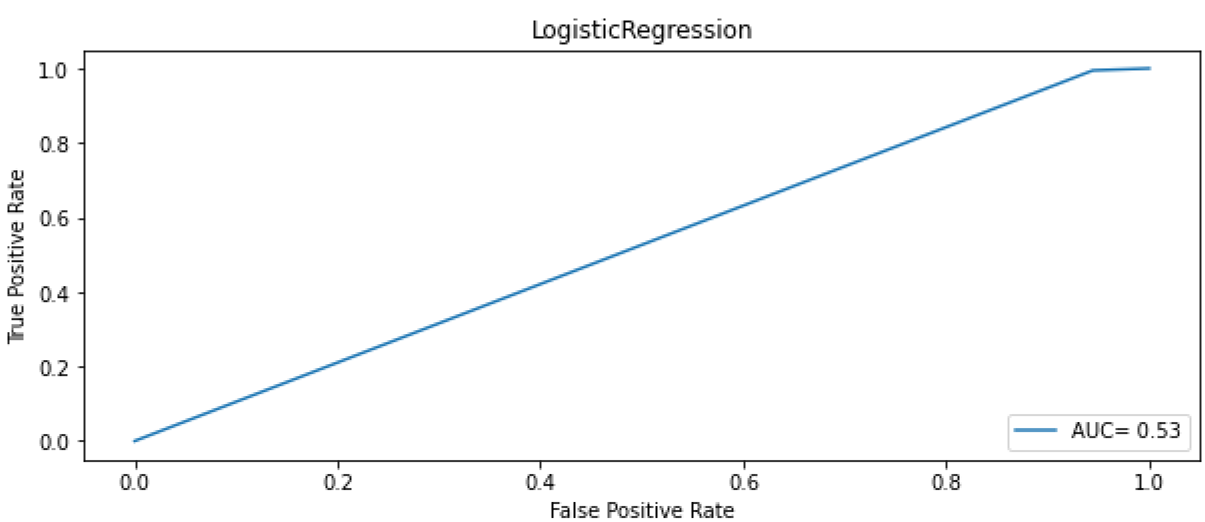
```
[[ 2651  2521]
 [  705 31372]]
```

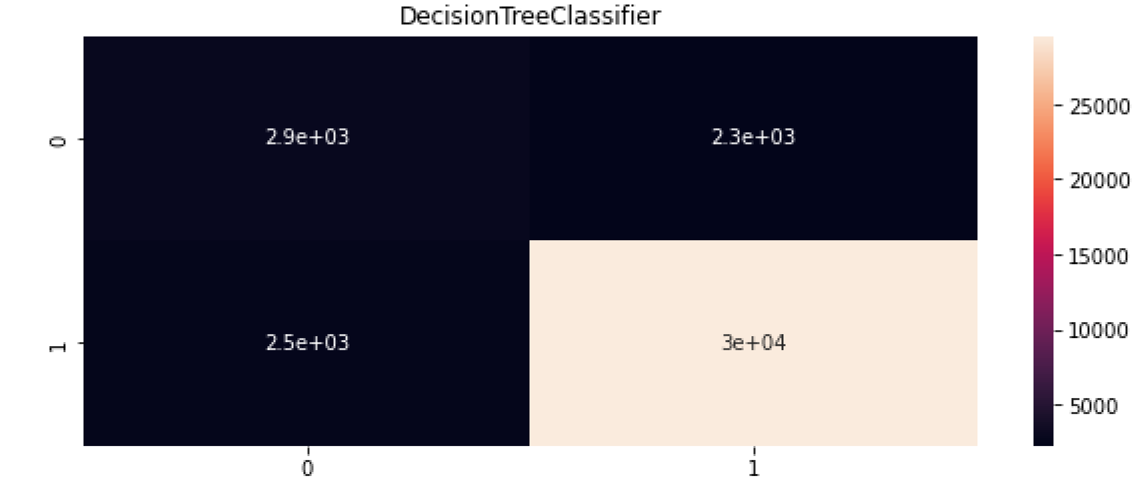
```
AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

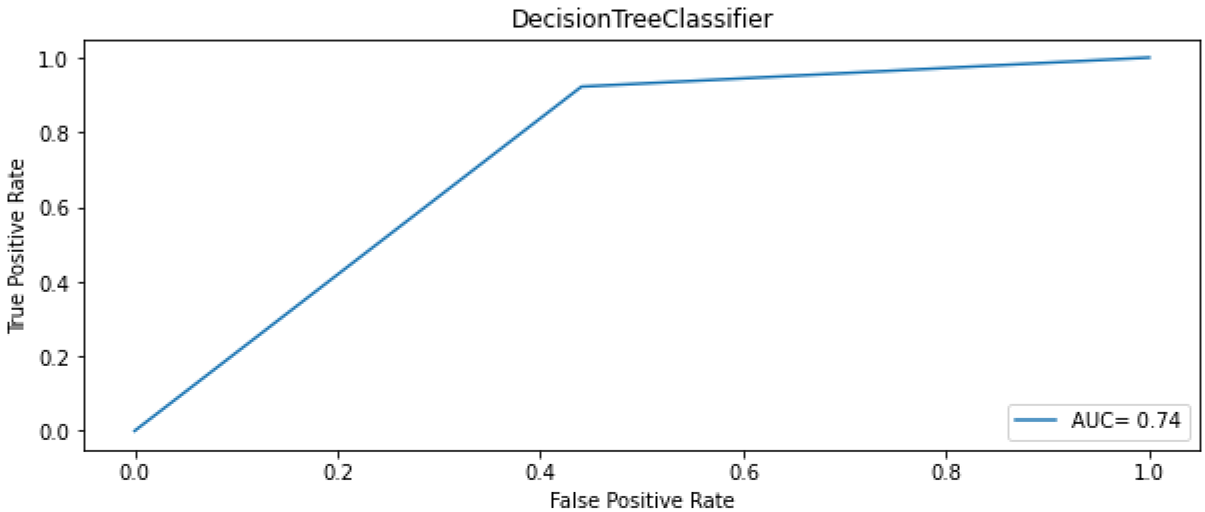

KNeighborsClassifier
2.2e+03
2.9e+03
1.9e+03
3e+04
0
1
0
1
30000
25000
20000
15000
10000
5000

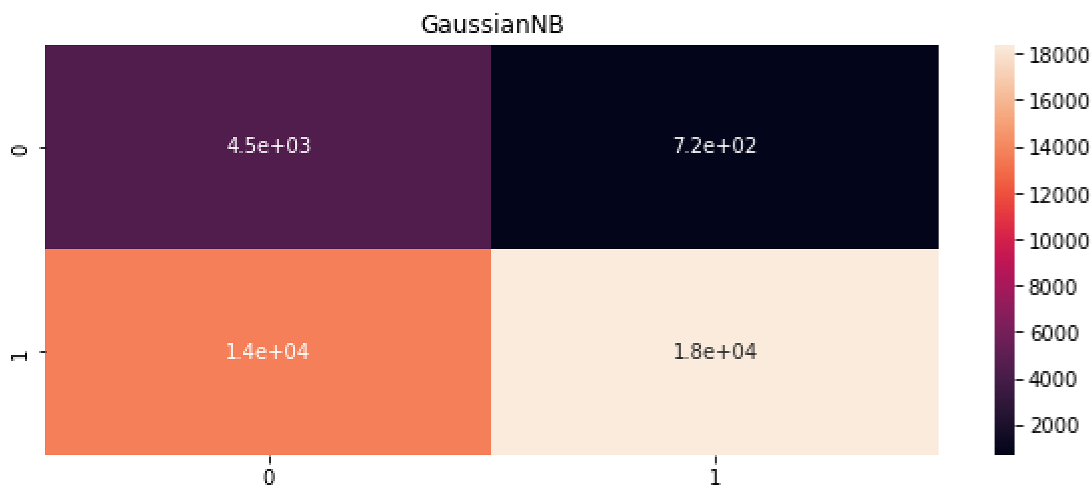

KNeighborsClassifier
True Positive Rate
False Positive Rate
AUC= 0.69
0.0
0.2
0.4
0.6
0.8
1.0

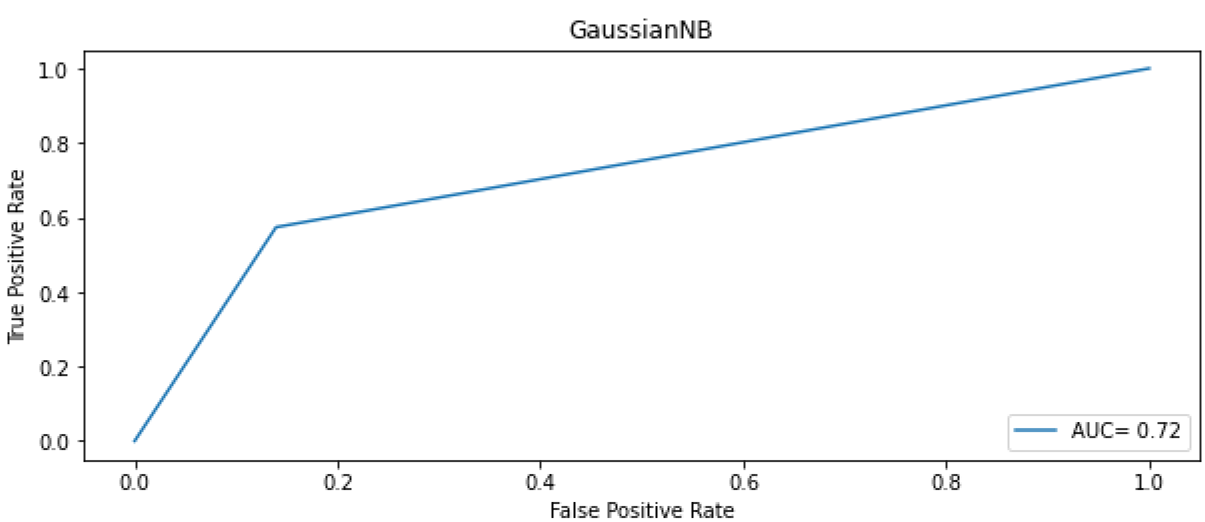

LogisticRegression
2.9e+02
4.9e+03
1.7e+02
3.2e+04
0
1
0
1
30000
25000
20000
15000
10000
5000

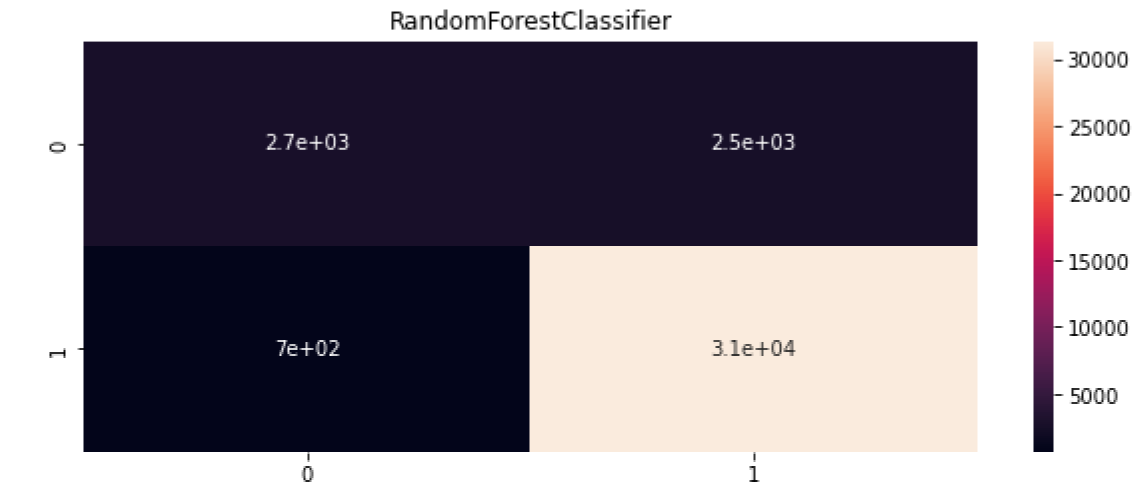

LogisticRegression
True Positive Rate
False Positive Rate
AUC= 0.53
0.0
0.2
0.4
0.6
0.8
1.0

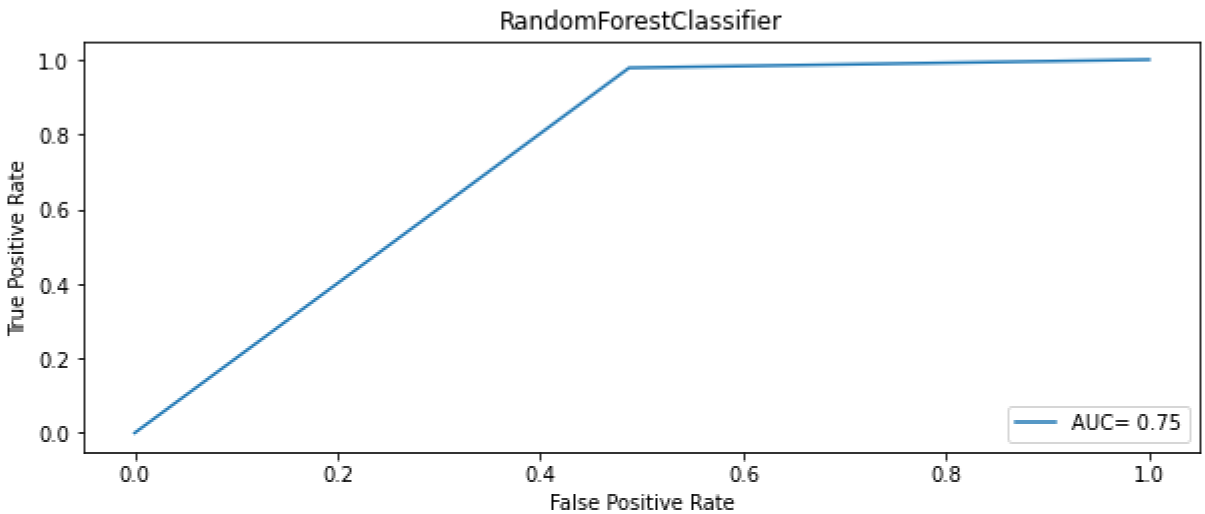

DecisionTreeClassifier
2.9e+03
2.3e+03
2.5e+03
3e+04
0
1
0
1
25000
20000
15000
10000
5000


DecisionTreeClassifier
True Positive Rate
False Positive Rate
AUC= 0.74
0.0
0.2
0.4
0.6
0.8
1.0


GaussianNB
4.5e+03
7.2e+02
1.4e+04
1.8e+04
0
1
0
1
18000
16000
14000
12000
10000
8000
6000
4000
2000


GaussianNB
True Positive Rate
False Positive Rate
AUC= 0.72
0.0
0.2
0.4
0.6
0.8
1.0





In [70]:

```
result=pd.DataFrame({'Model': Model, 'Accuracy_score': score, 'Cross_val_score':cvs, 'Roc_
result
```

Out[70]:

	Model	Accuracy_score	Cross_val_score	Roc_auc_curve
0	KNeighborsClassifier	86.990255	87.139379	68.671620
1	LogisticRegression	86.423797	86.423650	52.506450
2	DecisionTreeClassifier	87.175495	87.465835	74.082257
3	GaussianNB	61.362721	60.837705	71.720115
4	RandomForestClassifier	91.339365	91.341418	74.529465

So here 'RandomForestClassifier Model' is the best model out of all model tested above and by looking this we can conclude that our model is predicting around 92% of correct results for Label ‘0’ indicates that the loan has not been payed i.e. defaulter.

In [ ]: