Micro Credit Defaulter Project

Introduction

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on. Many microfinance institutions (MFI), experts and donorsare 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 MFShas 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 arecollaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be6(in Indonesian Rupiah), while, for the loan amount of 10(in Indonesian Rupiah), the payback amount should be 12(in Indonesian Rupiah). The sample data is provided to us from our client database. It is hereby given to you for this exercise. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

Feature Description

- label :Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}
- msisdn :mobile number of user
- aon :age on cellular network in days
- daily_decr30 :Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
- daily_decr90 :Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)

- rental30 :Average main account balance over last 30 days
- rental90 :Average main account balance over last 90 days
- last_rech_date_ma :Number of days till last recharge of main account
- last_rech_date_da : Number of days till last recharge of data account
- last_rech_amt_ma : Amount of last recharge of main account (in Indonesian Rupiah)
- cnt_ma_rech30 : Number of times main account got recharged in last 30 days
- fr_ma_rech30 : Frequency of main account recharged in last 30 days
- sumamnt_ma_rech30 : Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
- medianamnt_ma_rech30 : Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
- medianmarechprebal30 : Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
- cnt_ma_rech90 : Number of times main account got recharged in last 90 days
- fr_ma_rech90 : Frequency of main account recharged in last 90 days
- sumamnt_ma_rech90: Total amount of recharge in main account over last 90 days (in Indian Rupee)
- medianamnt_ma_rech90 :Median of amount of recharges done in main account over last 90 days at user level (in Indian Rupee)
- medianmarechprebal90 : Median of main account balance just before recharge in last 90 days at user level (in Indian Rupee)
- cnt_da_rech30 : Number of times data account got recharged in last 30 days
- fr_da_rech30 : Frequency of data account recharged in last 30 days
- cnt_da_rech90 : Number of times data account got recharged in last 90 days
- fr_da_rech90 : Frequency of data account recharged in last 90 days
- cnt_loans30 : Number of loans taken by user in last 30 days
- amnt_loans30 : Total amount of loans taken by user in last 30 days
- maxamnt_loans30 : maximum amount of loan taken by the user in last 30 days
- medianamnt_loans30 : Median of amounts of loan taken by the user in last 30 days
- cnt_loans90: Number of loans taken by user in last 90 days
- amnt_loans90 :Total amount of loans taken by user in last 90 days
- maxamnt_loans90 : maximum amount of loan taken by the user in last 90 days
- medianamnt_loans90: Median of amounts of loan taken by the user in last 90 days
- payback30 :Average payback time in days over last 30 days
- payback90: Average payback time in days over last 90 days
- pcircle: telecom circle
- pdate :date

Data Reading and Analysis

In [1]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df=pd.read_csv("micro_credit_defaulter.csv")

pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)

df.head()
```

Out[2]:

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

In [3]:
```
print('The shape of dataset:',df.shape)
```

```
The shape of dataset: (209593, 37)
```

In [4]:
```
df.drop('Unnamed: 0',axis=1,inplace=True)
df.head()
```

Out[4]:

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

In [5]:
```
#informtion about dataset
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
```

---> From here we can observe that there is three object type attributes are there .They are msisdn,pcircle,pdate

In [6]:
```
# let's check null values
df.isnull().sum()
```

Out[6]:
```
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
```

----> It's looking perfect there is no null or missing values in this data set

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 [7]:
```
unique = df.nunique()
unique = unique[unique.values == 1]
```

In [8]:
```
df.drop(labels = list(unique.index), axis =1, inplace=True)
print("So now we are left with",df.shape ,"rows & columns.")
```

```
So now we are left with (209593, 35) rows & columns.
```

In [9]:
```
df.head()
```

Out[9]:

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

In [10]:
```
df.describe().transpose()
```

Out[10]:

	count	mean	std	min	25%	50%
label	209593.0	0.875177	0.330519	0.000000	1.000	1.000000
aon	209593.0	8112.343445	75696.082531	-48.000000	246.000	527.000000
daily_decr30	209593.0	5381.402289	9220.623400	-93.012667	42.440	1469.175667
daily_decr90	209593.0	6082.515068	10918.812767	-93.012667	42.692	1500.000000
rental30	209593.0	2692.581910	4308.586781	-23737.140000	280.420	1083.570000
rental90	209593.0	3483.406534	5770.461279	-24720.580000	300.260	1334.000000
last_rech_date_ma	209593.0	3755.847800	53905.892230	-29.000000	1.000	3.000000
last_rech_date_da	209593.0	3712.202921	53374.833430	-29.000000	0.000	0.000000
last_rech_amt_ma	209593.0	2064.452797	2370.786034	0.000000	770.000	1539.000000
cnt_ma_rech30	209593.0	3.978057	4.256090	0.000000	1.000	3.000000
fr_ma_rech30	209593.0	3737.355121	53643.625172	0.000000	0.000	2.000000
sumamnt_ma_rech30	209593.0	7704.501157	10139.621714	0.000000	1540.000	4628.000000
medianamnt_ma_rech30	209593.0	1812.817952	2070.864620	0.000000	770.000	1539.000000
medianmarechprebal30	209593.0	3851.927942	54006.374433	-200.000000	11.000	33.900000
cnt_ma_rech90	209593.0	6.315430	7.193470	0.000000	2.000	4.000000
fr_ma_rech90	209593.0	7.716780	12.590251	0.000000	0.000	2.000000
sumamnt_ma_rech90	209593.0	12396.218352	16857.793882	0.000000	2317.000	7226.000000
medianamnt_ma_rech90	209593.0	1864.595821	2081.680664	0.000000	773.000	1539.000000
medianmarechprebal90	209593.0	92.025541	369.215658	-200.000000	14.600	36.000000
cnt_da_rech30	209593.0	262.578110	4183.897978	0.000000	0.000	0.000000
fr_da_rech30	209593.0	3749.494447	53885.414979	0.000000	0.000	0.000000
cnt_da_rech90	209593.0	0.041495	0.397556	0.000000	0.000	0.000000
fr_da_rech90	209593.0	0.045712	0.951386	0.000000	0.000	0.000000
cnt_loans30	209593.0	2.758981	2.554502	0.000000	1.000	2.000000
amnt_loans30	209593.0	17.952021	17.379741	0.000000	6.000	12.000000
maxamnt_loans30	209593.0	274.658747	4245.264648	0.000000	6.000	6.000000
medianamnt_loans30	209593.0	0.054029	0.218039	0.000000	0.000	0.000000
cnt_loans90	209593.0	18.520919	224.797423	0.000000	1.000	2.000000
amnt_loans90	209593.0	23.645398	26.469861	0.000000	6.000	12.000000
maxamnt_loans90	209593.0	6.703134	2.103864	0.000000	6.000	6.000000
medianamnt_loans90	209593.0	0.046077	0.200692	0.000000	0.000	0.000000
payback30	209593.0	3.398826	8.813729	0.000000	0.000	0.000000
payback90	209593.0	4.321485	10.308108	0.000000	0.000	1.666667

In [11]:
```
#Here we check the summary of object and datetime columns
df.describe(include=['object','datetime']).transpose()
```

Out[11]:

	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 [12]:
```
#Here the data copy [df1] variable
df1=df.copy()
```

In [13]:
```
#Deleting the duplicates entry in msidn column
df = df.drop_duplicates(subset = 'msisdn',keep='first')
df.shape
```

Out[13]:
```
(186243, 35)
```

Data Exploration

In [14]:
```
#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 [15]:
```
#Printing the float datatype columns and number of unique values in the particular col

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

```
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 [16]:
```
#Checking the number of number of defaulter and non defaulter customers.
df['label'].value_counts()
```

Out[16]:
```
1    160383
0     25860
Name: label, dtype: int64
```

In [17]:
```
#Checking the defaulter customers percentage wise.
df['label'].value_counts(normalize=True) *100
```

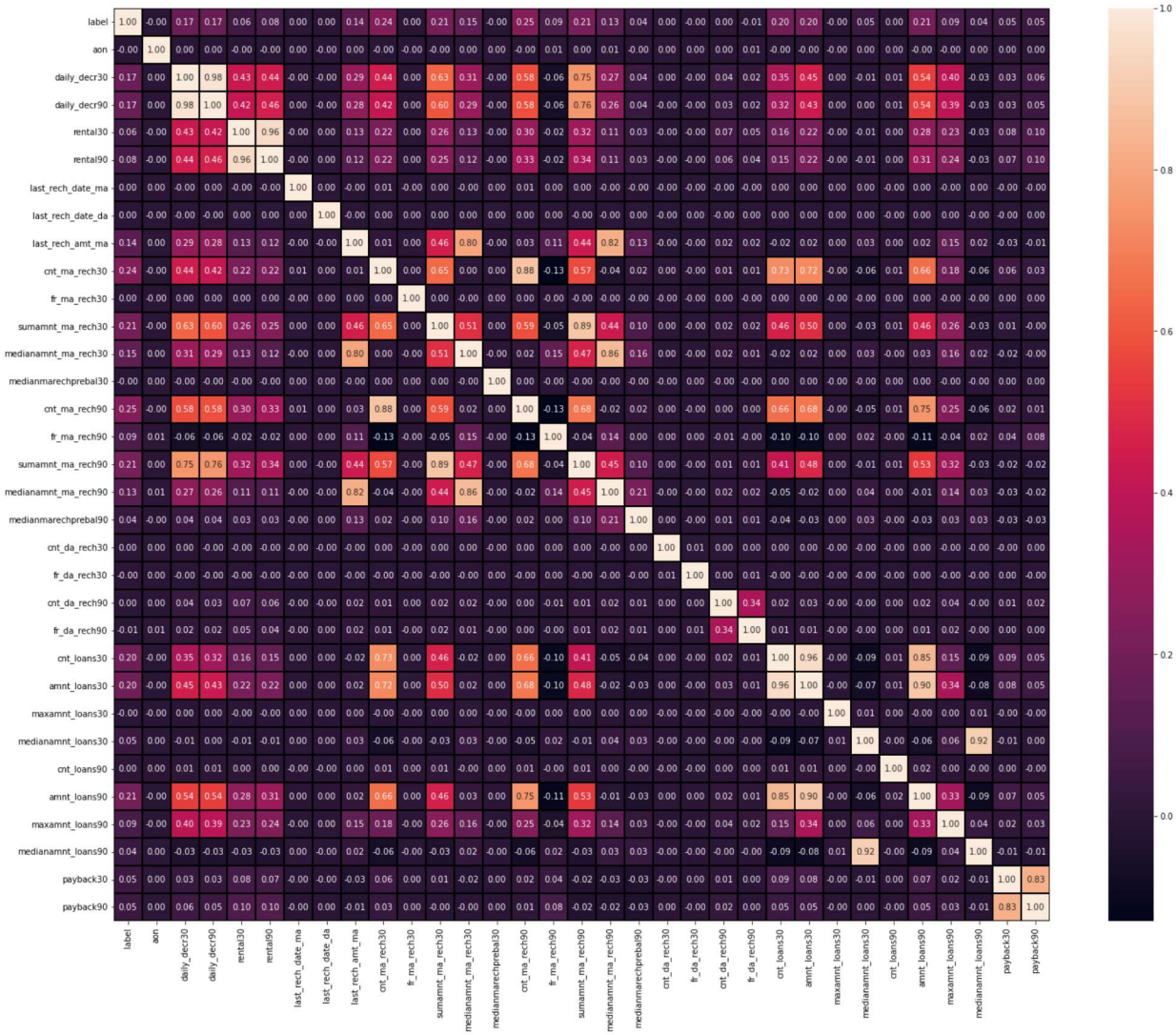
Out[17]:
```
1    86.114914
0    13.885086
Name: label, dtype: float64
```

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 [18]:
```
#Contructing a heat mat to visualize the correlation matrix
plt.figure(figsize=(25,20))
sns.heatmap(df.corr(),annot=True, linewidths=0.1,linecolor="black",fmt="0.2f")
```

Out[18]:
```
<AxesSubplot:>
```



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 [19]:
```
#Dropping the columns which is highly correlated with each other do avoid multicolined
df.drop(columns=['daily_decr30','rental30','amnt_loans30','medianamnt_loans30'],axis=1
```

In [20]:
```
#Now checking the shape
print(df.shape)
#Checking the unique value in pdate column.
df['pdate'].nunique()
```

Out[20]:
```
(186243, 31)
82
```

In [21]:
```
#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 [22]:
```
df.head()
```

Out[22]:

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

In [23]:
```
#Checking the number of months
df['pMonth'].unique()
```

Out[23]:
```
array([7, 8, 6], dtype=int64)
```

In [24]:
```
#After fetching the data from pdate column now we are going to drop it because it has
df.drop(columns=['pdate'],axis=1, inplace = True)
```

In [25]:
```
#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
```

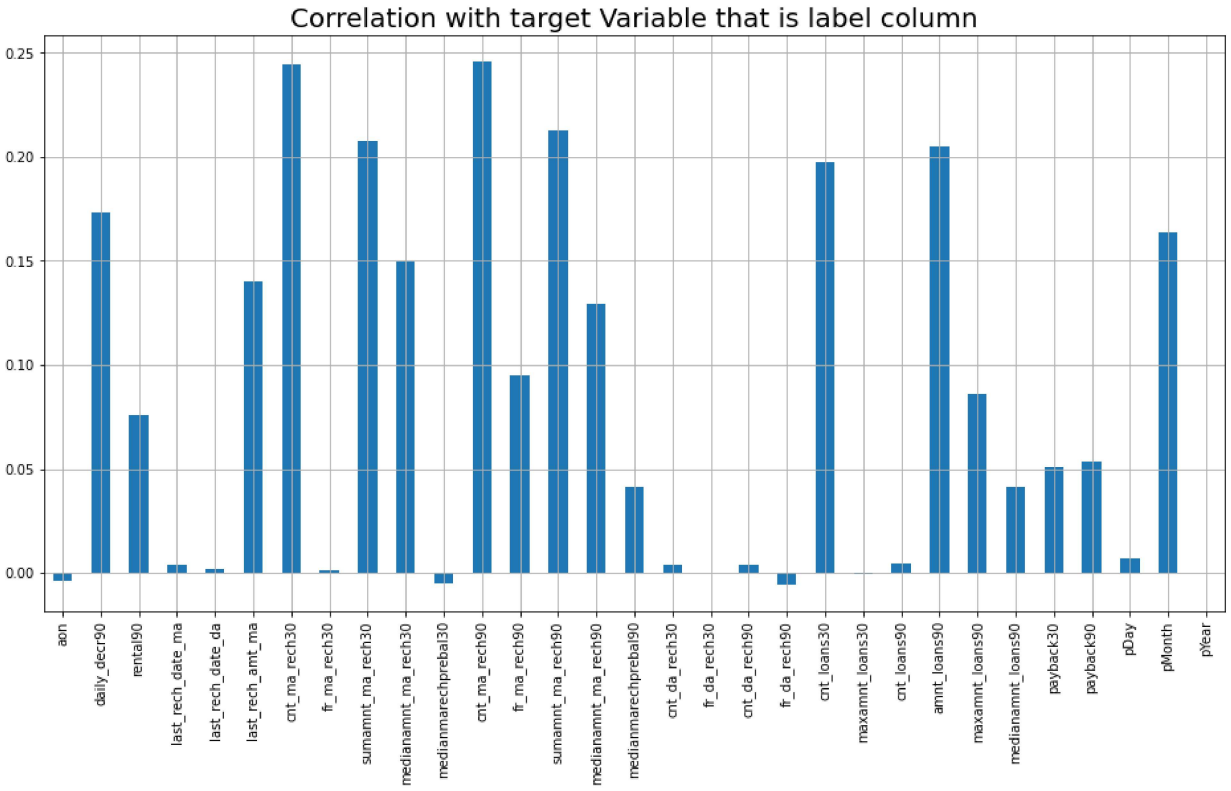
Data Visualization

In [26]:
```
#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=20)
```

Out[26]: Text(0.5, 1.0, 'Correlation with target Variable that is label column')

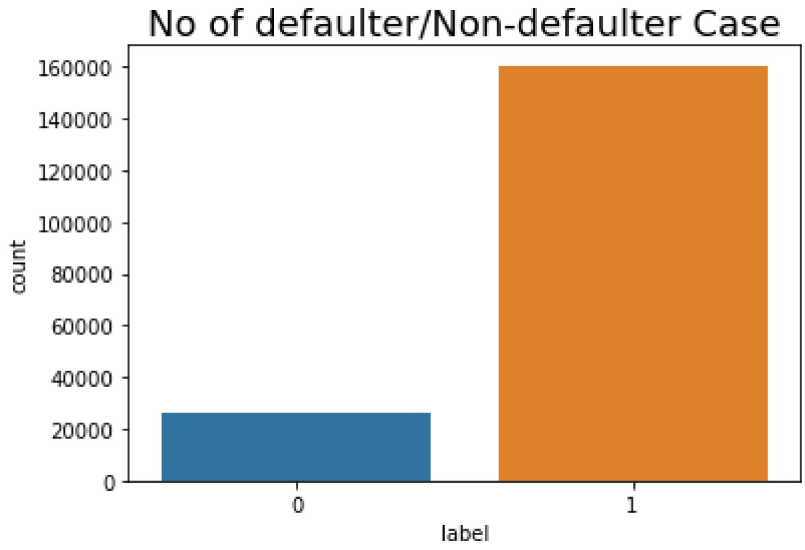


Observation:

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

In [27]:
```
#Checking the number of Fraud cases.
sns.countplot(x='label', data=df)
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.

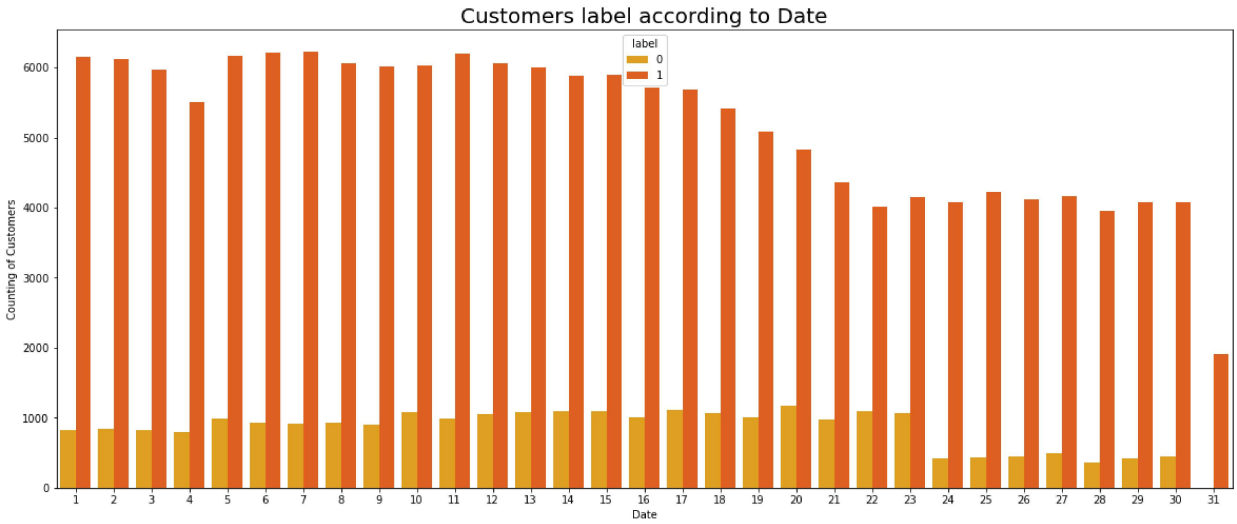
In [28]:
```
#Plotting the Histogram
df.hist(figsize=(20,20),color='r')
plt.show()
```

Observation:

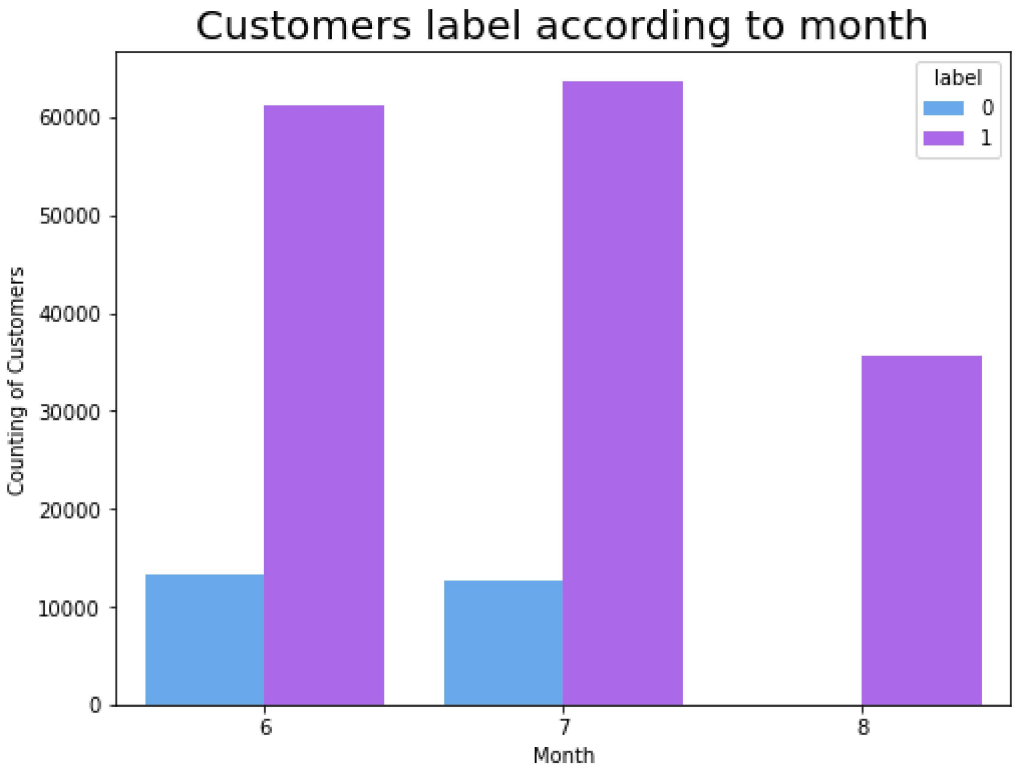
- We plot the histogram to display the shape and spread of continuous sample data.In a histogram, each bar groups numbers into ranges. Taller bars show that more data falls in that range

In [29]:
```
#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=20)
plt.xlabel('Date')
plt.ylabel('Counting of Customers')
plt.show()
```



In [30]:
```
#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=20)
plt.xlabel('Month')
plt.ylabel('Counting of Customers')
plt.show()
```

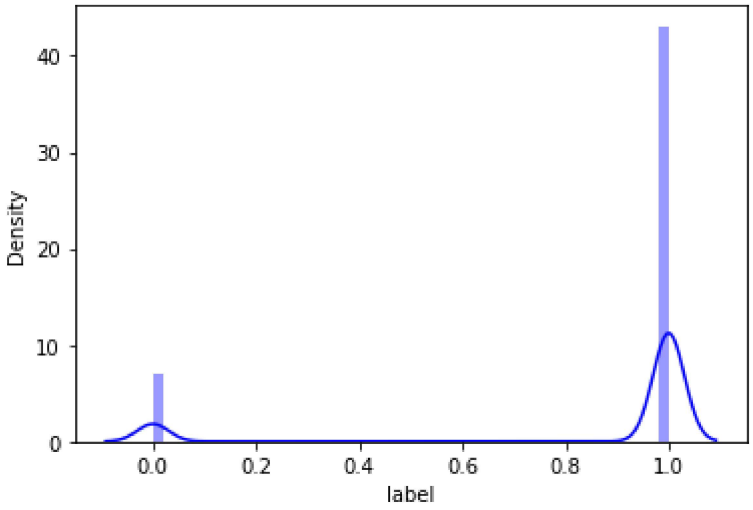


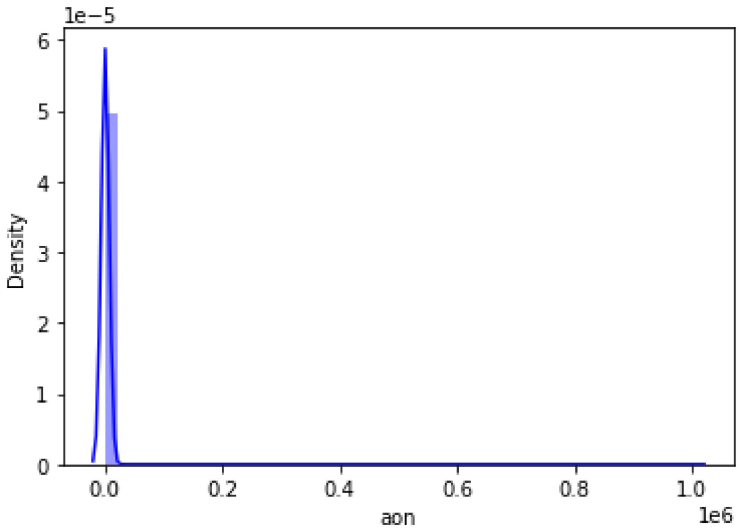
Observation:

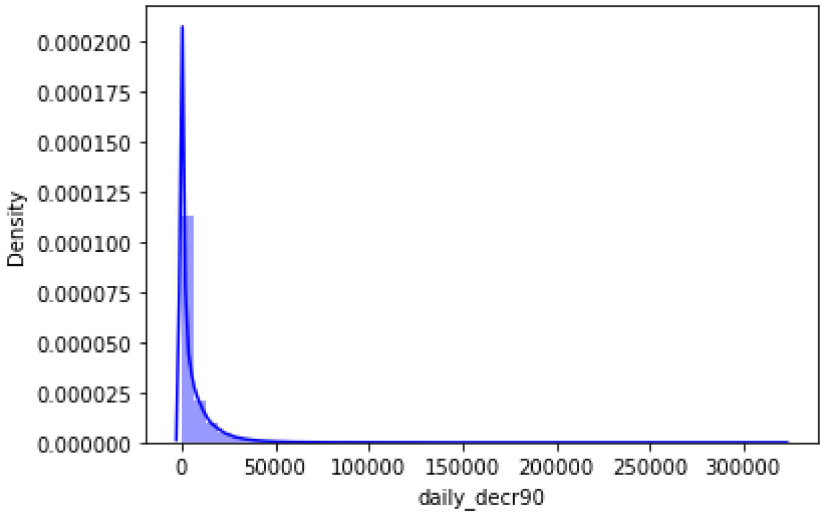
- The first figure which is date vs label shows that the customers who did not pay their loans are from date 10 to 23.
- There are severals customers at June and July month who did not pay their loan.

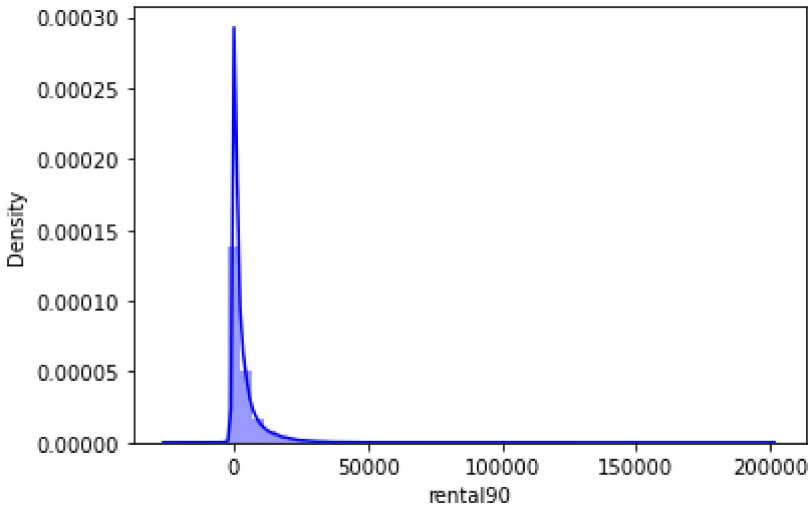
skewness

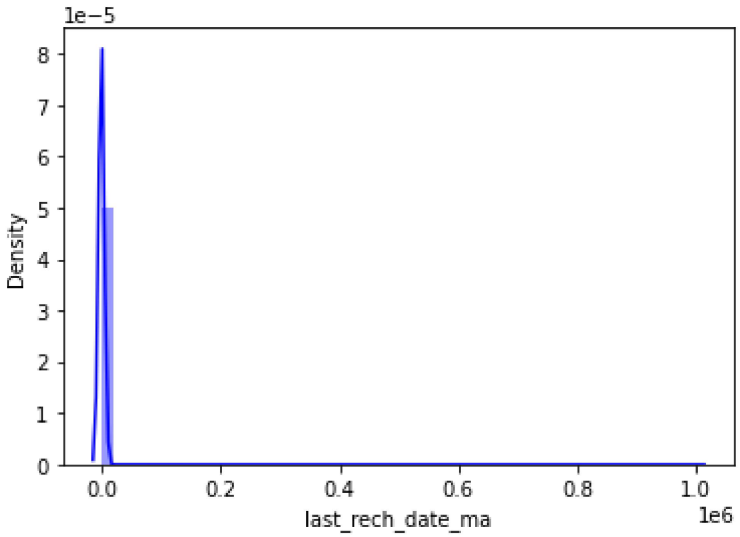
In [31]:
```
#checking skewness
for col in df.describe().columns:
    sns.distplot(df[col],color='b')
    plt.show()
```

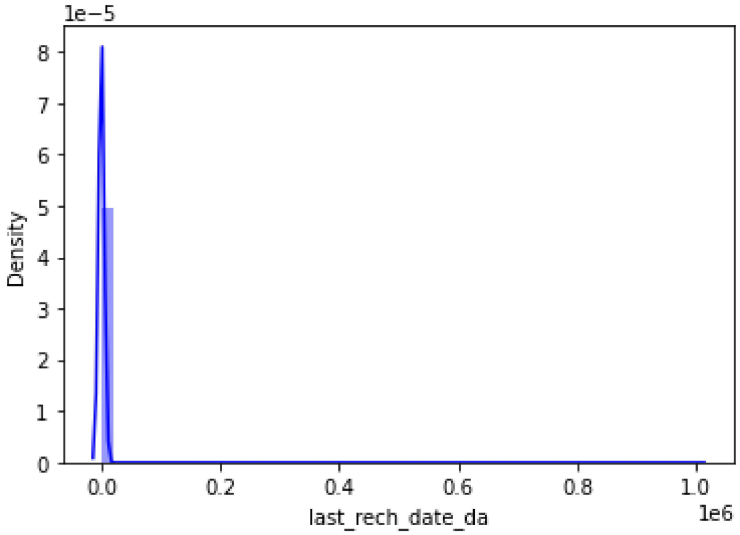


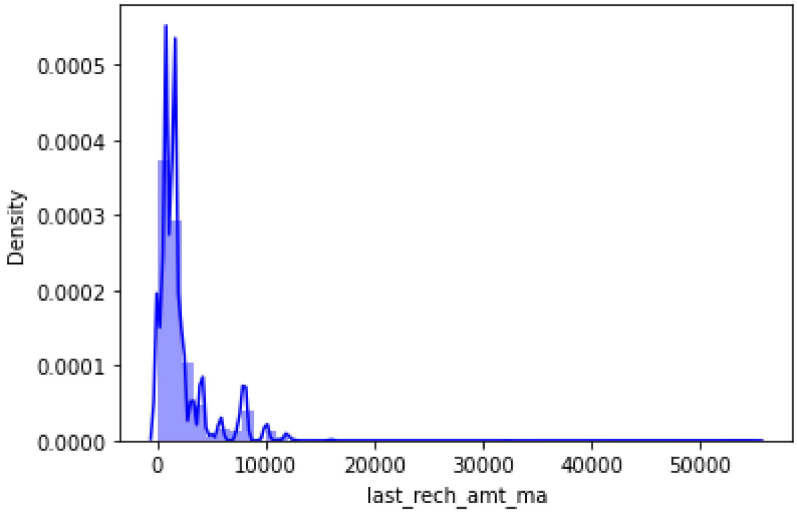


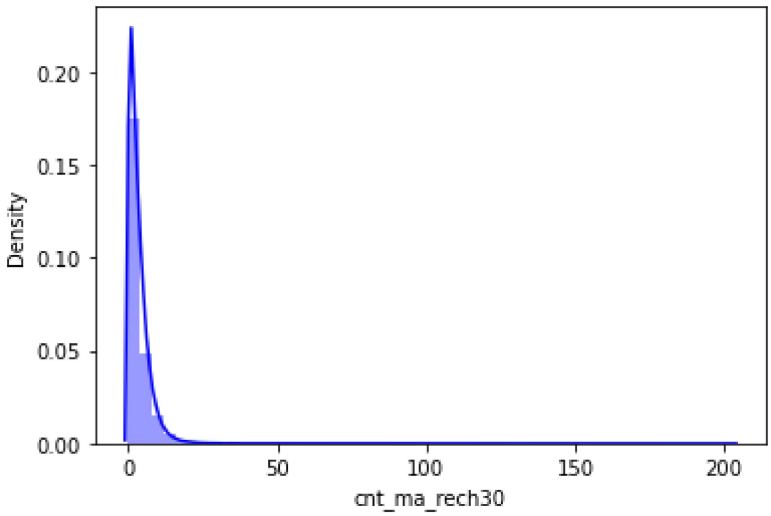

0.000200
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
Density
0
50000
100000
150000
200000
250000
300000
daily_decr90

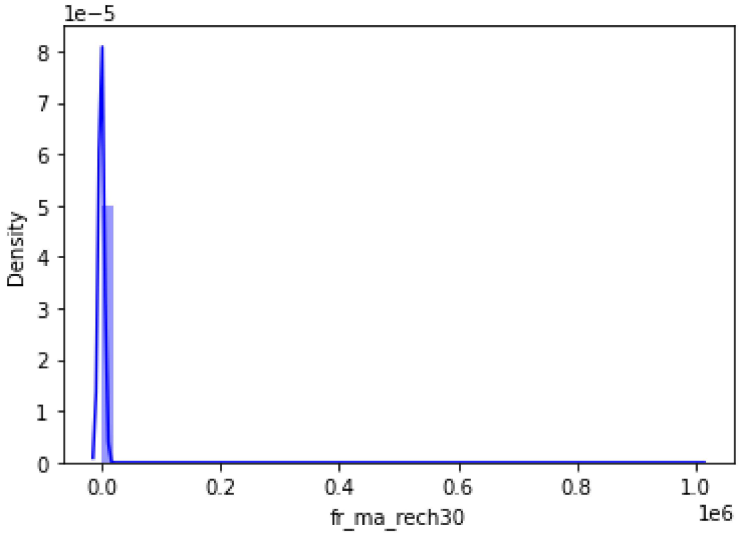

0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
Density
0
50000
100000
150000
200000
rental90

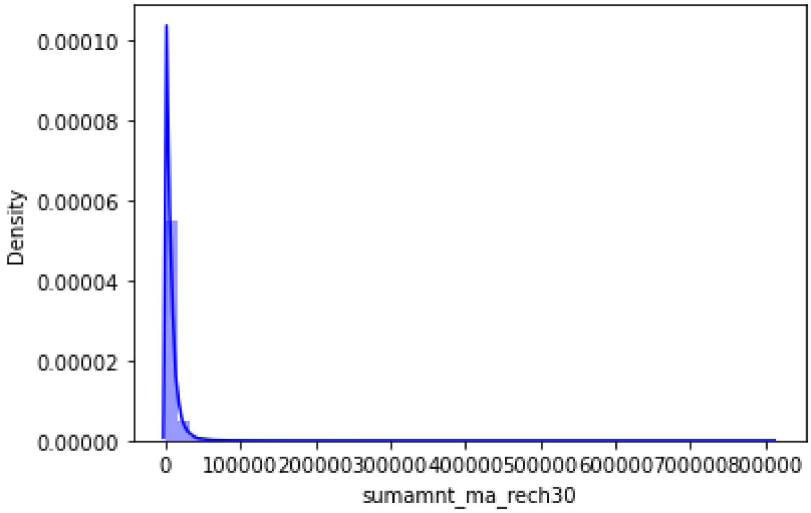

1e−5
8
7
6
5
4
3
2
1
0
Density
0.0
0.2
0.4
0.6
0.8
1.0
1e6
last_rech_date_ma

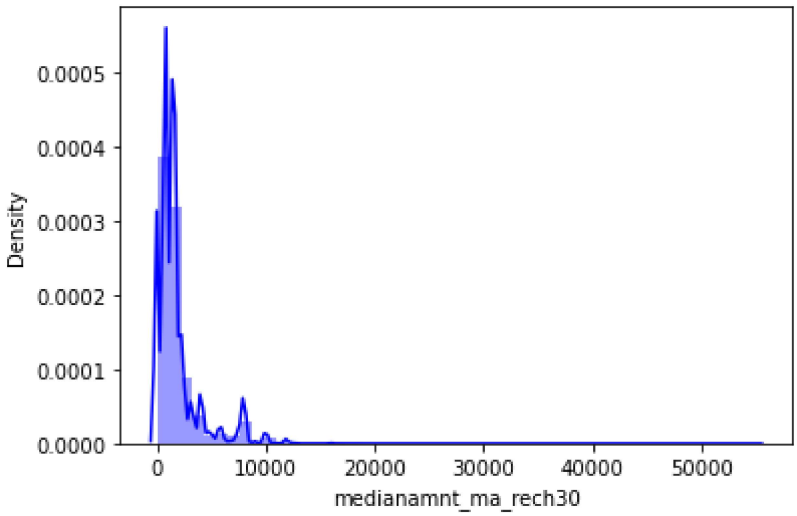

1e−5
8
7
6
5
4
3
2
1
0
Density
0.0
0.2
0.4
0.6
0.8
1.0
1e6
last_rech_date_da

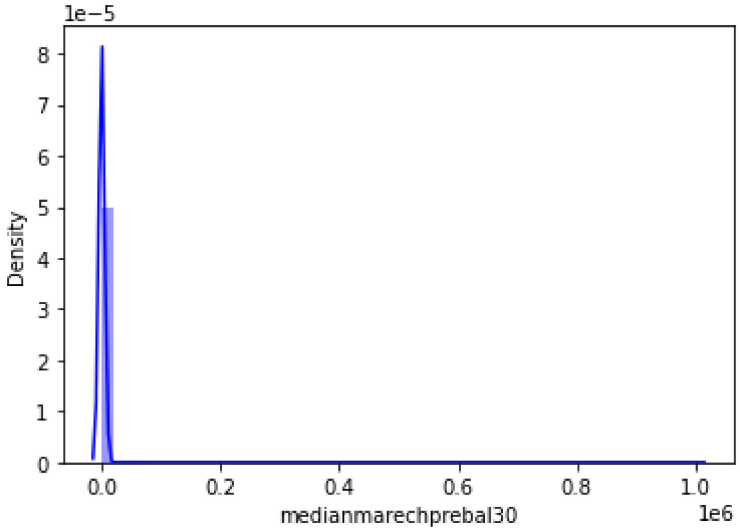

0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
Density
0
10000
20000
30000
40000
50000
last_rech_amt_ma

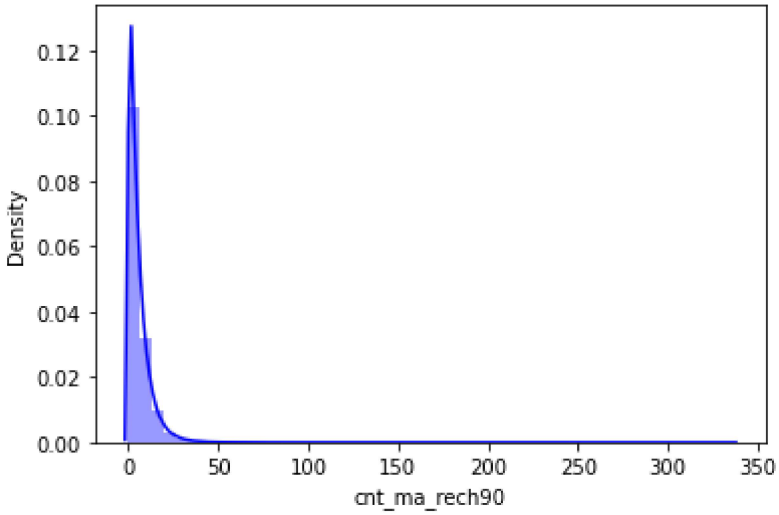

0.20
0.15
0.10
0.05
0.00
Density
0
50
100
150
200
cnt_ma_rech30

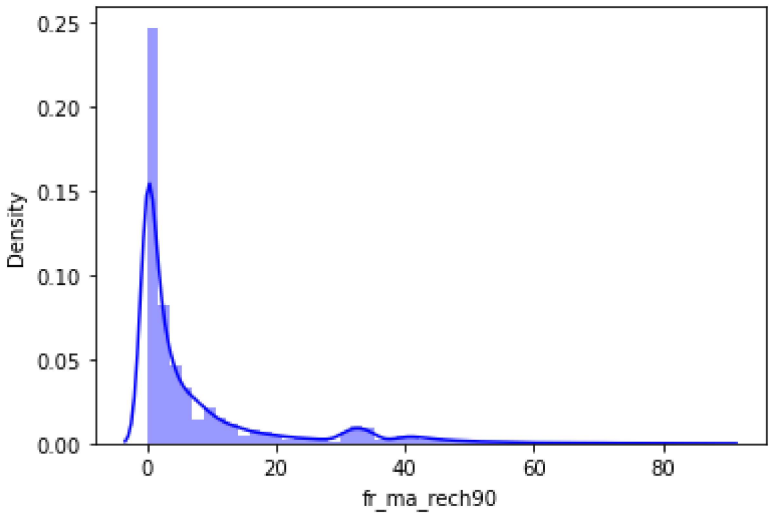

1e−5
8
7
6
5
4
3
2
1
0
Density
0.0
0.2
0.4
0.6
0.8
1.0
1e6
fr_ma_rech30

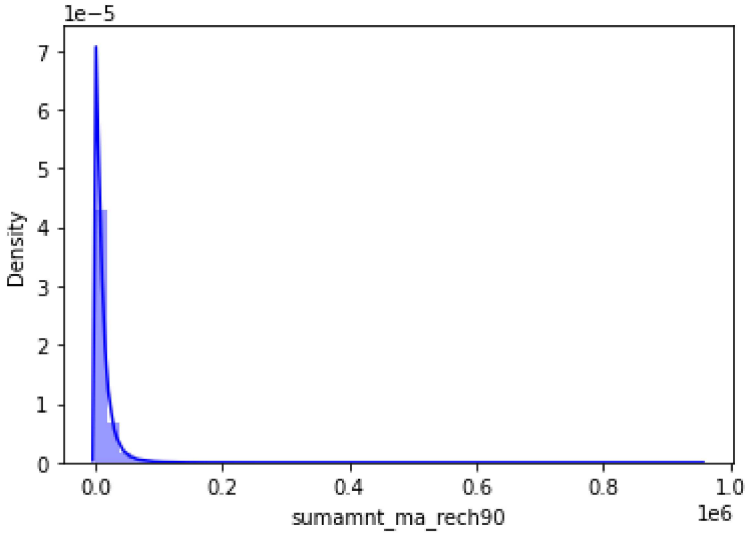

0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
Density
0
100000
200000
300000
400000
500000
600000
700000
800000
sumamnt_ma_rech30

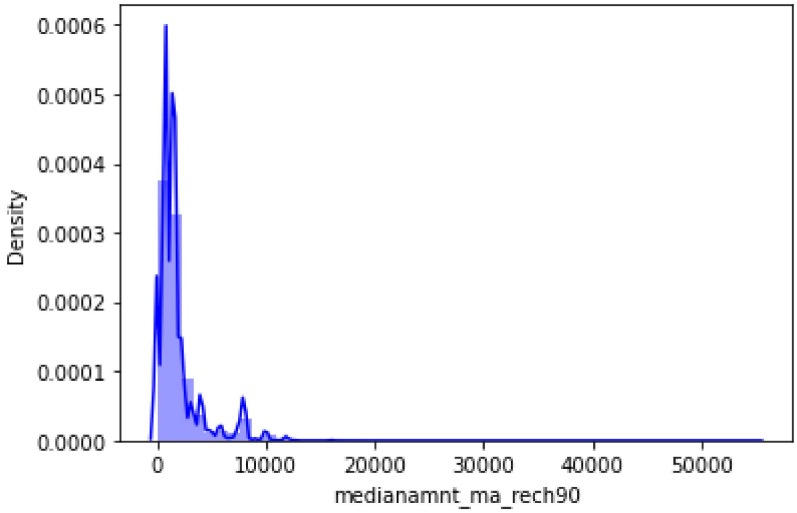

0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
Density
0
10000
20000
30000
40000
50000
medianamnt_ma_rech30

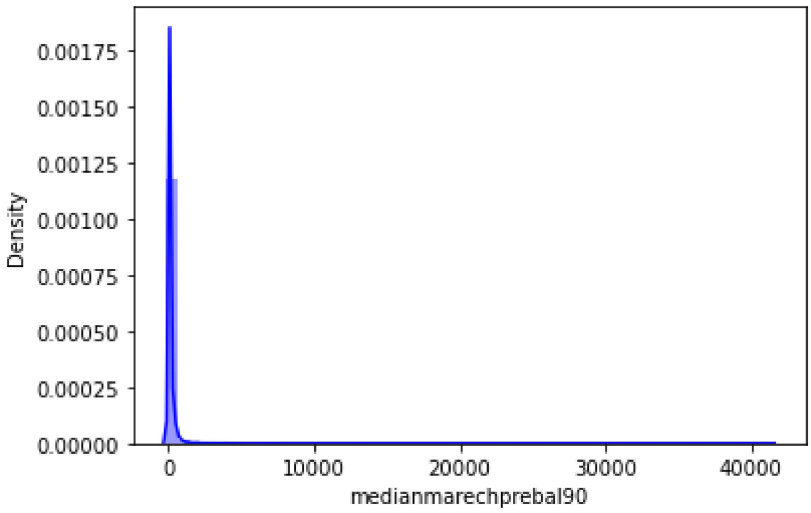

1e−5
8
7
6
5
4
3
2
1
0
Density
0.0
0.2
0.4
0.6
0.8
1.0
1e6
medianmarechprebal30

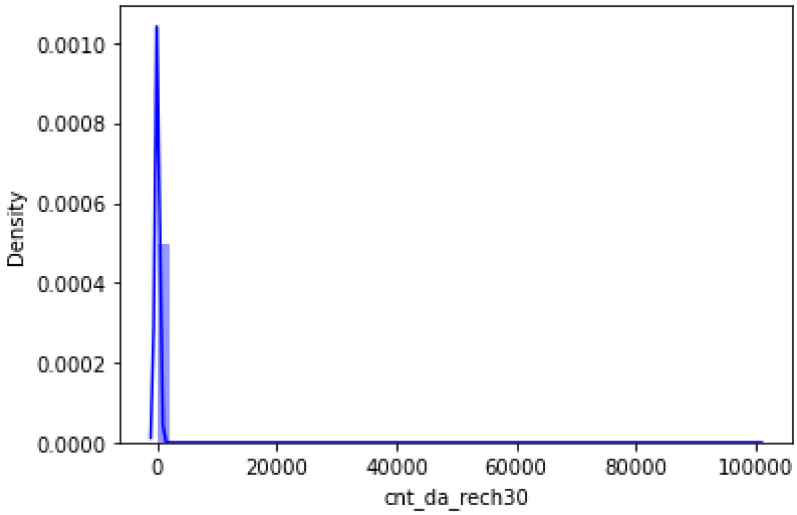

0.12
0.10
0.08
0.06
0.04
0.02
0.00
Density
0
50
100
150
200
250
300
350
cnt_ma_rech90

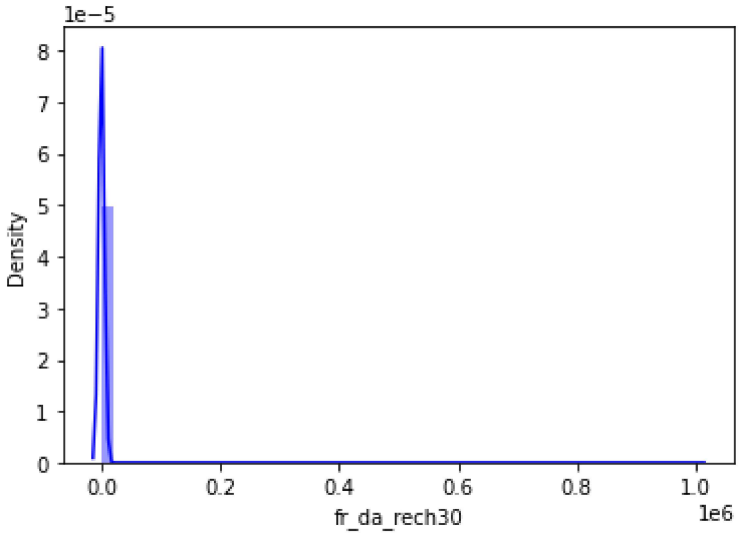

0.25
0.20
0.15
0.10
0.05
0.00
Density
0
20
40
60
80
fr_ma_rech90

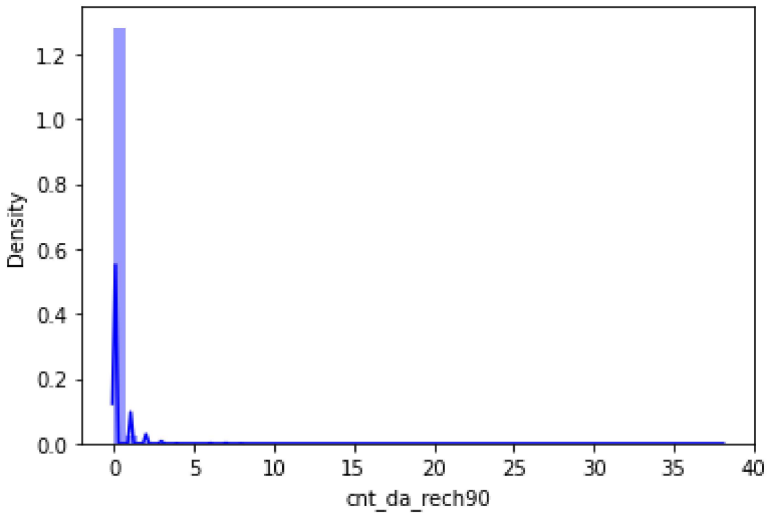

1e−5
Density
sumamnt_ma_rech90
1e6

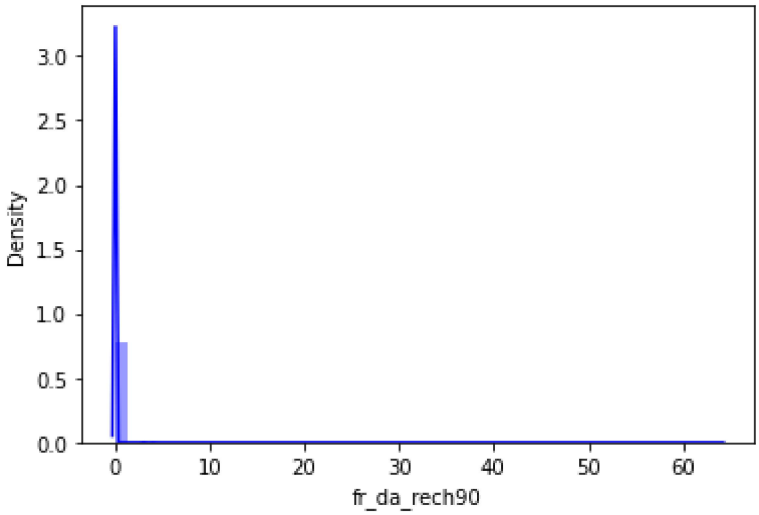

Density
medianamnt_ma_rech90

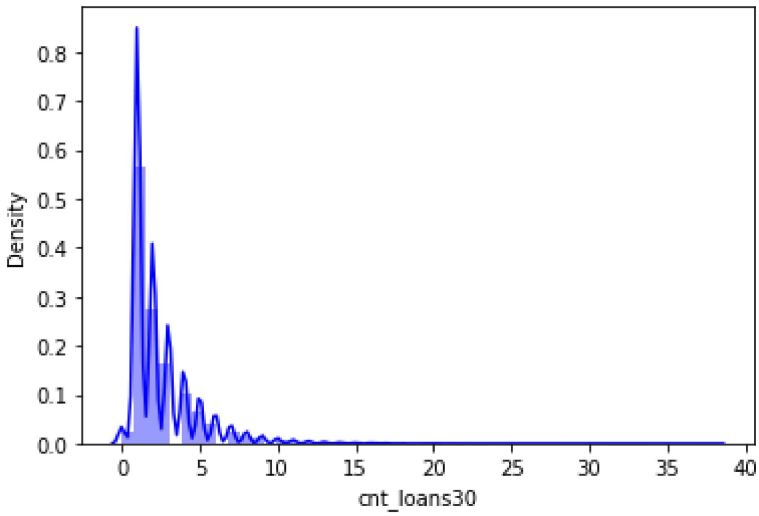

Density
medianmarechprebal90

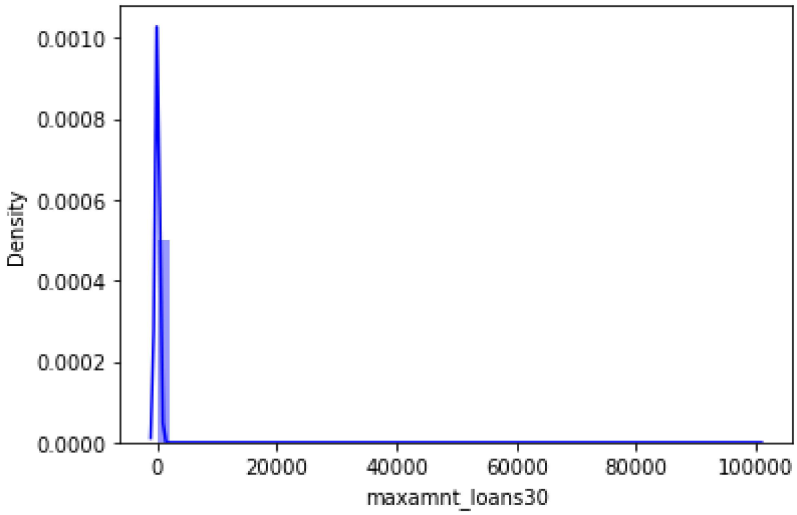

0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
20000
40000
60000
80000
100000
cnt_da_rech30

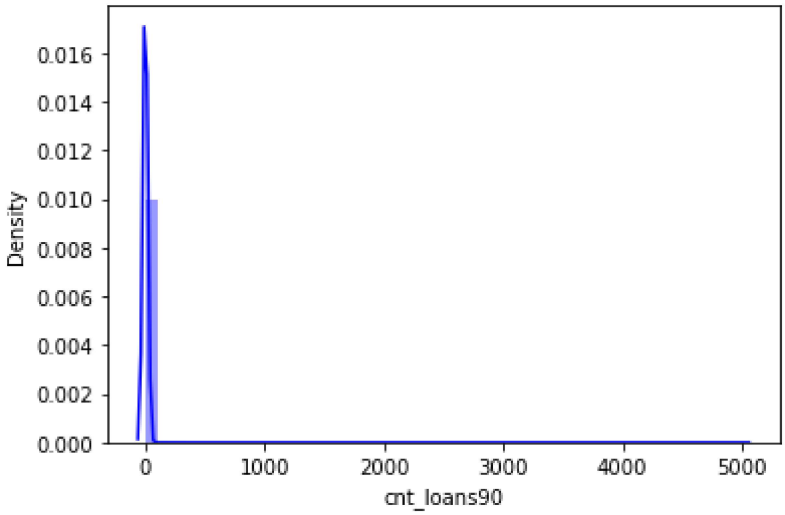

1e−5
8
7
6
5
4
3
2
1
0
Density
0.0
0.2
0.4
0.6
0.8
1.0
1e6
fr_da_rech30

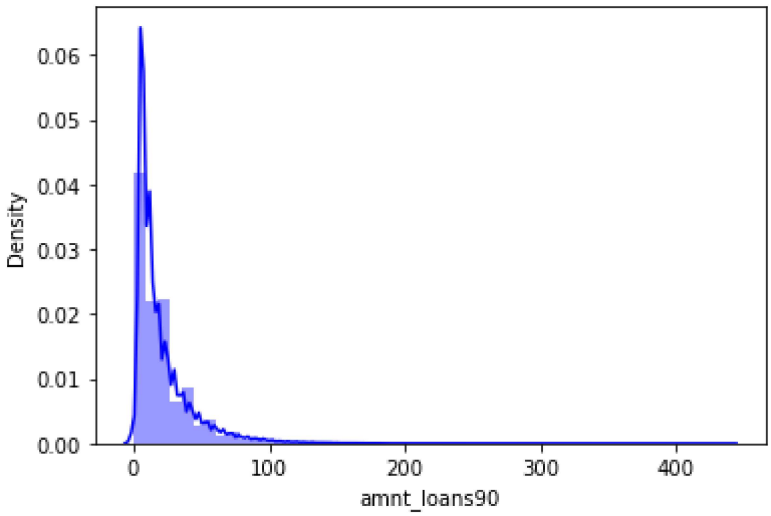

1.2
1.0
0.8
0.6
0.4
0.2
0.0
Density
0
5
10
15
20
25
30
35
40
cnt_da_rech90

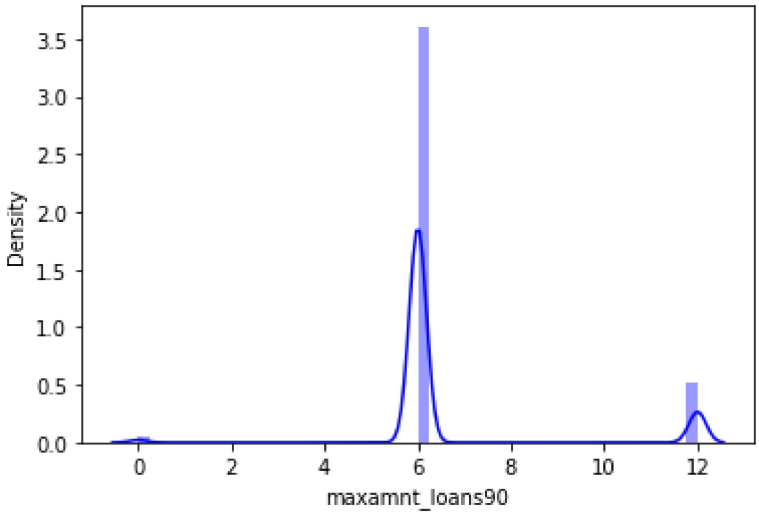

Density
fr_da_rech90

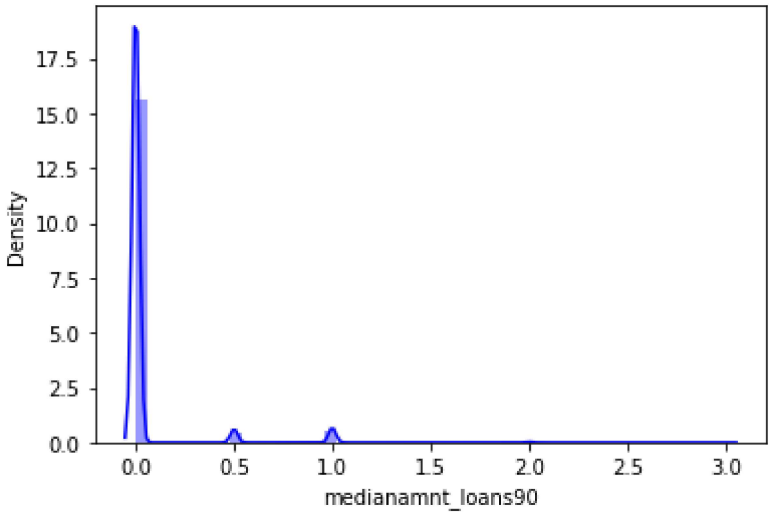

Density
cnt_loans30

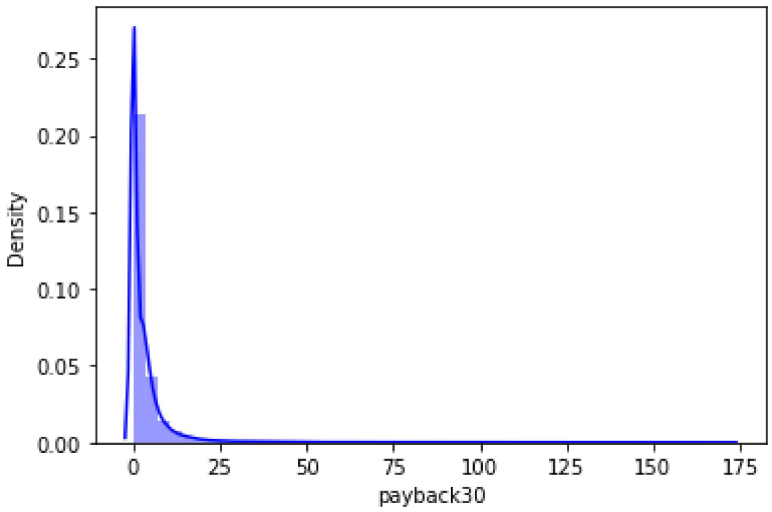

Density
maxamnt_loans30

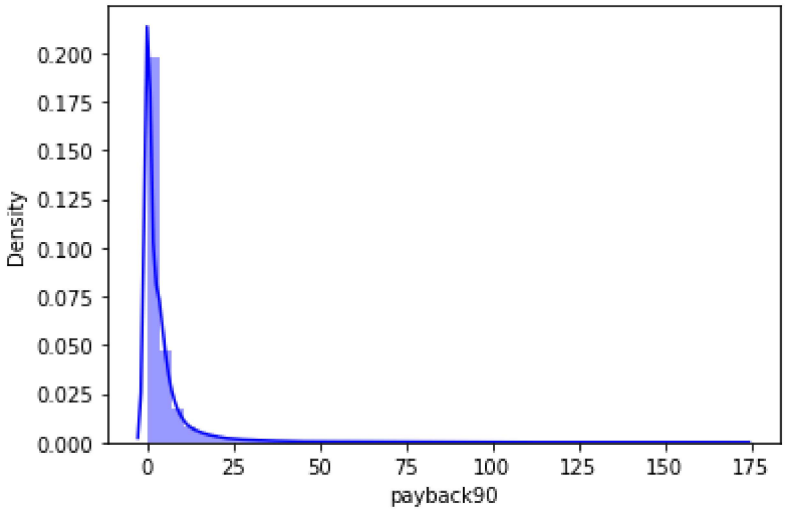

Density
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
1000
2000
3000
4000
5000
cnt_loans90

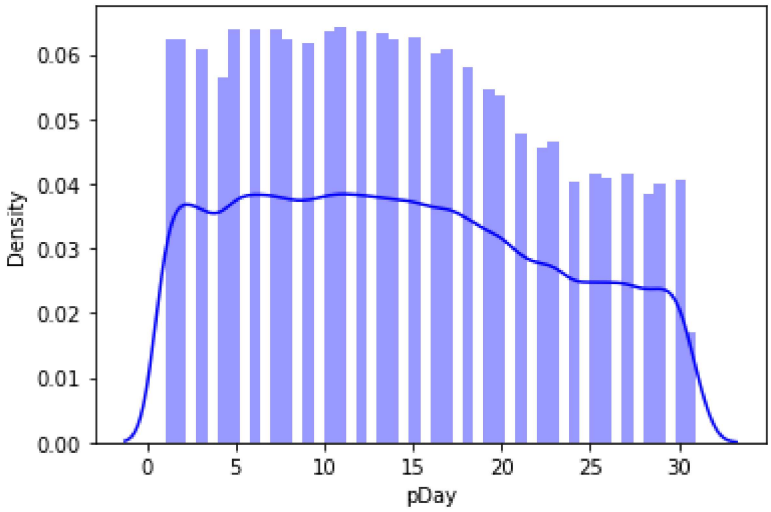

Density
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
100
200
300
400
amnt_loans90

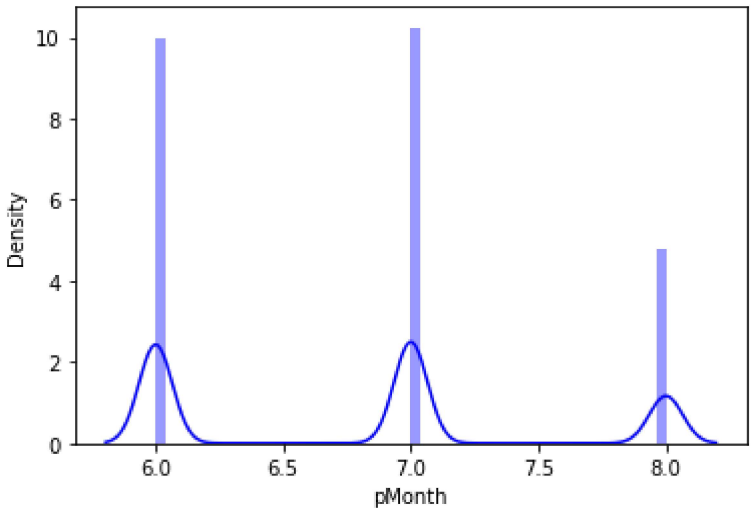

Density
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
10
12
maxamnt_loans90

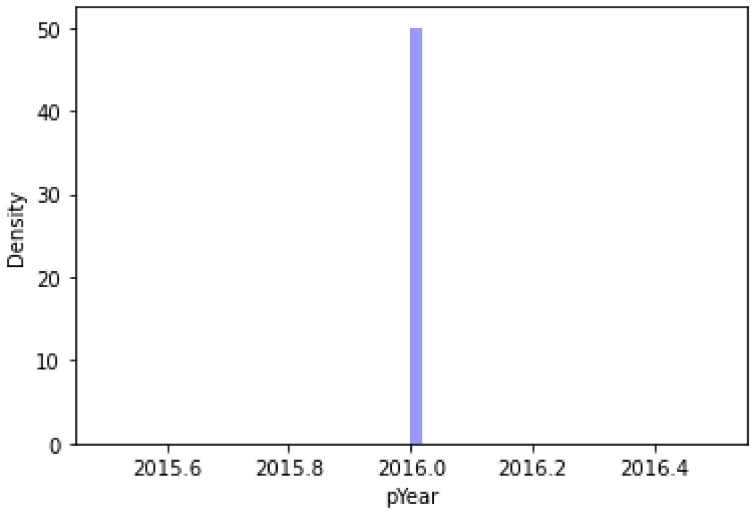

Density
medianamnt_loans90


Density
payback30


Density
payback90


Density
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
5
10
15
20
25
30
pDay


Density
10
8
6
4
2
0
6.0
6.5
7.0
7.5
8.0
pMonth


Density
50
40
30
20
10
0
2015.6
2015.8
2016.0
2016.2
2016.4
pYear

In [32]:
```
df.skew()
```

Out[32]:
```
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 [33]:
```
#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 [34]:
```
df.skew()
```

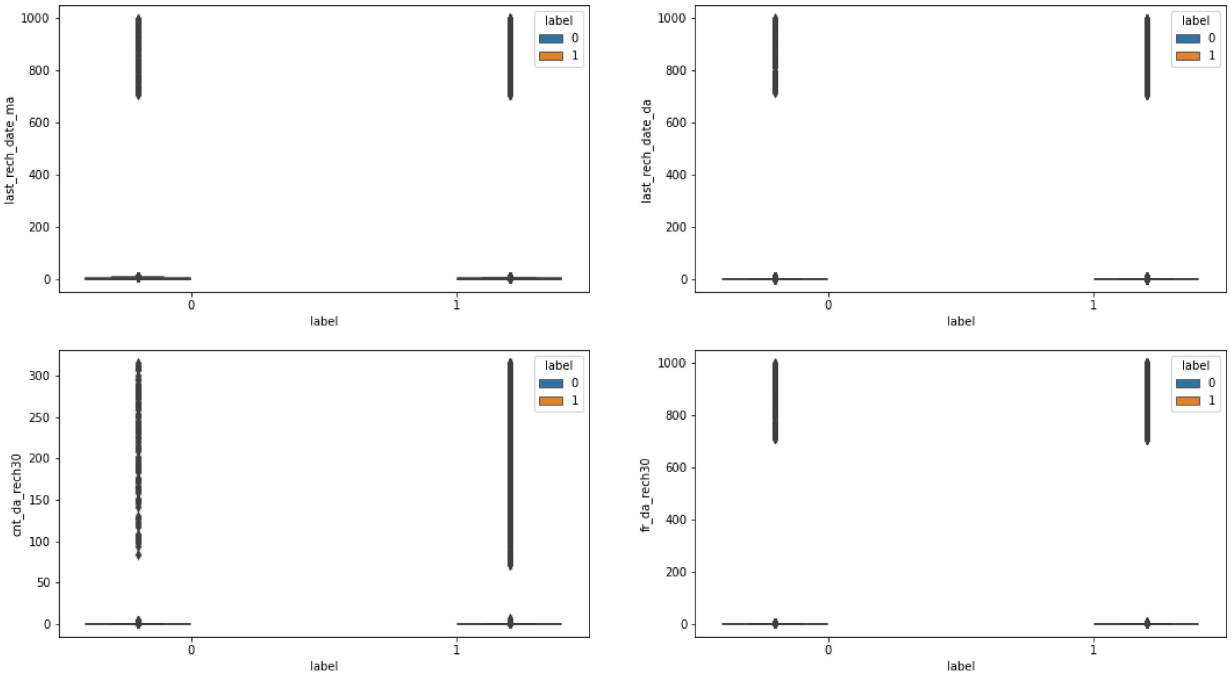
Out[34]:
```
label                   -2.088847
aon                      9.753837
daily_decr90             1.249642
rental90                 1.356501
last_rech_date_ma       14.249692
last_rech_date_da       14.212446
last_rech_amt_ma         0.780525
cnt_ma_rech30            0.331739
fr_ma_rech30            14.269759
sumamnt_ma_rech30        0.826903
medianamnt_ma_rech30     0.617369
medianmarechprebal30    13.824799
cnt_ma_rech90            0.582326
fr_ma_rech90             1.010242
sumamnt_ma_rech90        0.987018
medianamnt_ma_rech90     0.742013
medianmarechprebal90     4.231474
cnt_da_rech30           15.539060
fr_da_rech30            14.201794
cnt_da_rech90            8.159487
fr_da_rech90            19.838971
cnt_loans30              1.063634
maxamnt_loans30         15.559444
cnt_loans90             14.096883
amnt_loans90             1.353973
maxamnt_loans90         -1.100783
medianamnt_loans90       3.936508
payback30                2.011413
payback90                1.831942
pDay                     0.200706
pMonth                   0.351293
pYear                    0.000000
dtype: float64
```

Find out the outliers

In [35]:
```
#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[35]:
```
<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 [36]:
```
#Creating a copy of our dataset
df2=df1.copy()
#Dropping the object columns
df1.drop(columns=['msisdn','pdate'],axis=1,inplace=True)
```

In [37]:
```
df1.columns
```

Out[37]:
```
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')
```

Removing the outliers

In [ ]:
```
from scipy.stats import zscore
z=np.abs(zscore(df1))
z
```

In [ ]:
```
threshold=3
print(np.where(z>3))
```

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

In [ ]:
```
#Checking the shape
print(df1.shape,'\t\t',df1_new.shape)
```

Converting the categorical data into numeric variables

In [ ]:
```
# 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])
```

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

Splitting the data into x and y

In [ ]:
```
#Splitting the data into x and y
x = df.drop(['label'], axis=1)

y = df['label']
```

DecisionTreeClassifier

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

In [ ]:
```
dt_features = pd.DataFrame(dt.feature_importances_, index=x.columns, columns=['feat_im
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 [ ]:
```
#Scaling in input variables
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x=ss.fit_transform(x)
```

In [ ]:
```
#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,strat
```

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

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

In [ ]:
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_
```

In [ ]:
```
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')
```

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

In [ ]:
```

```