NAME OF THE PROJECT

MICRO CREDIT DEFAULTER

Submitted by:

KHUSHBOO GUPTA

ACKNOWLEDGMENT

First and foremost, I would like to thank Flip Robo Technologies to provide me a chance to work on this project. It was a great experience to work on this project under your guidance.

I would like to present my gratitude to the following websites:

- Zendesk
- Kaggle
- Datatrained Notes
- Sklearn.org
- Crazyegg
- Towards data science

These websites were of great help and due to this, I was able to complete my project effectively and efficiently.

INTRODUCTION

• Business Problem Framing

Build a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan. In this case, Label ‘1’ indicates that the loan has been payed i.e. Non- defaulter, while, Label ‘0’ indicates that the loan has not been payed i.e. defaulter.

• Conceptual Background of the Domain Problem

Basic EDA concepts and classification algorithms must be known to work on this project. One should know what is a credit defaulter and what factors can help to determine whether a person is going to be a defaulter or not?

• Review of Literature

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.

Analytical Problem Framing

- Data Sources and their formats

The dataset is provided by the internship organization in an csv format which contains the data in code sheet. It contains 37 columns and 209593 rows. There are so many factors which can be used for the prediction whether a person is defaulter or not. It contains the loan history of a person such as how many loans a person takes in last 30 days or 90 days, what is maximum loan amount and many more on which we can predict a person is able to pay his credit in 5 days or not.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1		label	msisdn	aon	daily_decr	daily_decr	rental30	rental90	last_rech_	last_rech_	last_rech_	cnt_ma_re	fr_ma_rec	sumamnt_	medianam	medianma	cnt_ma_re	fr_ma_rec	sumamnt_	medianam	medianma	cnt_da_re	fr_da_rech
2	1	0	21408I707	272	3055.05	3065.15	220.13	260.13	2	0	1539	2	21	3078	1539	7.5	2	21	3078	1539	7.5	0	0
3	2	1	76462I703	712	12122	12124.75	3691.26	3691.26	20	0	5787	1	0	5787	5787	61.04	1	0	5787	5787	61.04	0	0
4	3	1	17943I703	535	1398	1398	900.13	900.13	3	0	1539	1	0	1539	1539	66.32	1	0	1539	1539	66.32	0	0
5	4	1	55773I707	241	21.228	21.228	159.42	159.42	41	0	947	0	0	0	0	0	1	0	947	947	2.5	0	0
6	5	1	03813I827	947	150.6193	150.6193	1098.9	1098.9	4	0	2309	7	2	20029	2309	29	8	2	23496	2888	35	0	0
7	6	1	35819I707	568	2257.363	2261.46	368.13	380.13	2	0	1539	4	10	6156	1539	15.4	8	0	11744	1539	55.9	0	0
8	7	1	96759I844	545	2876.642	2883.97	335.75	402.9	13	0	5787	1	0	5787	5787	277.8	1	0	5787	5787	277.8	0	0
9	8	1	09832I908	768	12905	17804.15	900.35	2549.11	4	55	3178	3	3	10404	3178	36	9	3	26095	3178	36	0	0
10	9	1	59772I844	1191	90.695	90.695	2287.5	2287.5	1	0	1539	4	1	6164	1539	39.9	4	1	6164	1539	39.9	0	0
11	10	1	56331I707	536	29.35733	29.35733	612.96	612.96	11	0	773	1	0	773	773	86.8	1	0	773	773	86.8	0	0
12	11	1	32893I827	1511	12.896	12.896	790.44	790.44	8	0	1539	2	5	2312	1156	16.83	2	5	2312	1156	16.83	0	0
13	12	0	82417I908	82	65.16667	65.16667	326.2	326.2	17	0	7526	2	0	9065	4532.5	489	2	0	9065	4532.5	489	0	0
14	13	1	11435I892	154	227.041	227.041	240.41	240.41	2	0	1547	4	2	19086	4773.5	63	7	30	28979	1720	92	0	0
15	14	1	66580I976	887	55.90933	55.90933	208.8	208.8	2	0	1539	5	5	7703	1539	20.9	7	5	8649	1539	22.9	0	0
16	15	1	63139I703	707	8919	10317.35	399.25	2453.78	3	0	770	3	6	3079	770	66	8	10	19185	1539	52.5	0	0
17	16	0	24075I892	1037	12	12	1216.8	1216.8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	17	0	82053I853	1583	1000	1000	1000.8	1087.88	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	18	1	37204I844	929	10.688	10.688	40	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	19	1	44217I904	832	14.4	14.4	1660.96	1660.96	1	0	2309	3	26	4618	1539	88.8	3	26	4618	1539	88.8	0	0
21	20	1	19611I908	450	48.935	48.935	726.3	726.3	1	0	1539	2	8	9539	4769.5	12	2	8	9539	4769.5	12	0	0
22	21	1	67813I905	100	769.614	777.46	1050.57	1167.3	6	0	770	5	20	8867	770	168	8	31	14380	771.5	52	0	0
23	22	0	75522I707	378	514.6933	515.2	56.26	58.2	2	0	773	1	0	773	773	542	2	64	1546	773	283.5	0	0
24	23	1	61590I952	463	1540	1541	969.12	969.12	4	0	770	1	0	770	770	43	2	66	1543	771.5	26.5	0	0
25	24	1	95027I908	857	58.02333	58.02333	479.44	479.44	2	0	1539	4	12	6164	1539	115.5	4	12	6164	1539	115.5	0	0
26	25	0	59645I827	966	291.5633	291.5633	-2020.09	-2020.09	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	26	1	59102I703	656	63.25	63.25	2855.7	2855.7	1	0	770	15	1	12674	770	254.6	19	1	14566	770	134.6	0	0
28	27	1	44692I905	1179	3703.272	3712.84	340.96	376.42	2	0	770	6	5	5395	771.5	38.5	9	5	7114	773	53.5	0	0
29	28	0	49345I908	871	505.6	508	9276.68	10569.17	16	0	770	1	0	770	770	77.9	1	0	770	770	77.9	0	0
30	29	1	64703I853	1610	6688.559	6705.26	3553.83	3653.05	1	0	4048	9	1	32149	2309	41.72	10	1	36489	3178.5	85.72	0	0

- Libraries Used

I am using different libraries to explore the datatset.

1. Pandas – It is used to load and store the dataset. We can discuss the dataset with the pandas different attributes like .info, .columns, .shape
2. Seaborn – It is used to plot the different types of plots like catplot, lineplot, countplot and more to have a better visualization of the dataset.
3. Matplotlib.pyplot – It helps to give a proper description to the plotted graph by seaborn and make our graph more informative.
4. Numpy – It is the library to perform the numerical analysis to the dataset

Load the Dataset

Importing the libraries

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Loading the dataset

In [2]:
```
pd.set_option('display.max_rows',None)   #displaying the maximum rows
pd.set_option('display.max_columns',None)   #displaying the maximum columns
df=pd.read_csv(r'F:\Internship - Data Science\Micro-Credit-Project--1-\Micro Credit Project\Data file.csv')
df.head()   #first five rows of the dataset
```

Out[2]:

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_r
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	

We have successfully load our dataset for our further processes.

Checking the Attributes

- First & last five rows the dataset
- Shape of the dataset
- Columns present in the dataset
- Brief info about the dataset
- Datatype of each column
- Null values present in the dataset
- Number of unique values present in each column

In [4]:
```
df.columns   #columns present in tne dataset
```
Out[4]:
```
Index(['Unnamed: 0', 'label', 'msisdn', 'aon', 'daily_decr30', 'daily_decr90',
       'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_date_da',
       'last_rech_amt_ma', 'cnt_ma_rech30', 'fr_ma_rech30',
       'sumamnt_ma_rech30', 'medianamnt_ma_rech30', 'medianmarechprebal30',
       'cnt_ma_rech90', 'fr_ma_rech90', 'sumamnt_ma_rech90',
       'medianamnt_ma_rech90', 'medianmarechprebal90', 'cnt_da_rech30',
       'fr_da_rech30', 'cnt_da_rech90', 'fr_da_rech90', 'cnt_loans30',
       'amnt_loans30', 'maxamnt_loans30', 'medianamnt_loans30', 'cnt_loans90',
       'amnt_loans90', 'maxamnt_loans90', 'medianamnt_loans90', 'payback30',
       'payback90', 'pcircle', 'pdate'],
      dtype='object')
```

In [5]:
```
df.shape  #no. of rows & columns in the dataset
```
Out[5]:
```
(209593, 37)
```

Dataset contanins 209593 rows & 37 columns

A brief info about the dataset

In [6]:
```
df.info()  #briefly describes the datatype & null values in the dataset
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 37 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   Unnamed: 0            209593 non-null  int64  
 1   label                 209593 non-null  int64  
 2   msisdn                209593 non-null  object 
 3   aon                   209593 non-null  float64
 4   daily_decr30          209593 non-null  float64
 5   daily_decr90          209593 non-null  float64
 6   rental30              209593 non-null  float64
 7   rental90              209593 non-null  float64
 8   last_rech_date_ma     209593 non-null  float64
 9   last_rech_date_da     209593 non-null  float64
 10  last_rech_amt_ma      209593 non-null  int64  
 11  cnt_ma_rech30         209593 non-null  int64  
 12  fr_ma_rech30          209593 non-null  float64
 13  sumamnt_ma_rech30     209593 non-null  float64
 14  medianamnt_ma_rech30  209593 non-null  float64
 15  medianmarechprebal30  209593 non-null  float64
 16  cnt_ma_rech90         209593 non-null  int64  
```

Datatype of each column

In [10]:

```
df.dtypes
```

```
medianmarechprebal90    float64
cnt_da_rech30           float64
fr_da_rech30            float64
cnt_da_rech90             int64
fr_da_rech90              int64
cnt_loans30               int64
amnt_loans30              int64
maxamnt_loans30         float64
medianamnt_loans30      float64
cnt_loans90             float64
amnt_loans90              int64
maxamnt_loans90           int64
medianamnt_loans90      float64
payback30               float64
payback90               float64
Month                     int64
Day                       int64
Year                      int64
dtype: object
```

Unique values present in each column

In [11]:

```
df.nunique()
```

Out[11]:

```
label                        2
aon                       4507
daily_decr30            147025
daily_decr90            158669
rental30                132148
rental90                141033
last_rech_date_ma         1186
last_rech_date_da         1174
last_rech_amt_ma            70
cnt_ma_rech30               71
fr_ma_rech30              1083
sumamnt_ma_rech30        15141
medianamnt_ma_rech30       510
medianmarechprebal30     30428
cnt_ma_rech90              110
fr_ma_rech90                89
sumamnt_ma_rech90        31771
medianamnt_ma_rech90       608
medianmarechprebal90     29785
```

Finding the null values

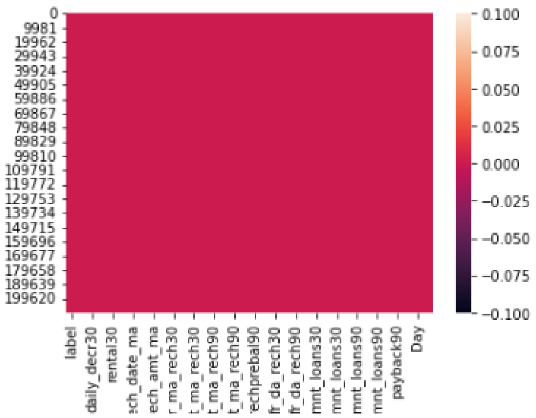
In [12]:
```
df.isnull().sum()
```

Out[12]:
```
label                   0
aon                     0
daily_decr30            0
daily_decr90            0
rental30                0
rental90                0
last_rech_date_ma       0
last_rech_date_da       0
last_rech_amt_ma        0
cnt_ma_rech30           0
fr_ma_rech30            0
sumamnt_ma_rech30       0
medianamnt_ma_rech30    0
medianmarechprebal30    0
cnt_ma_rech90           0
fr_ma_rech90            0
sumamnt_ma_rech90       0
medianamnt_ma_rech90    0
medianmarechprebal90    0
```

Our dataset doesn't contain the null values

In [13]:
```
sns.heatmap(df.isnull())  #plotting the null values
```

Out[13]: <AxesSubplot:>

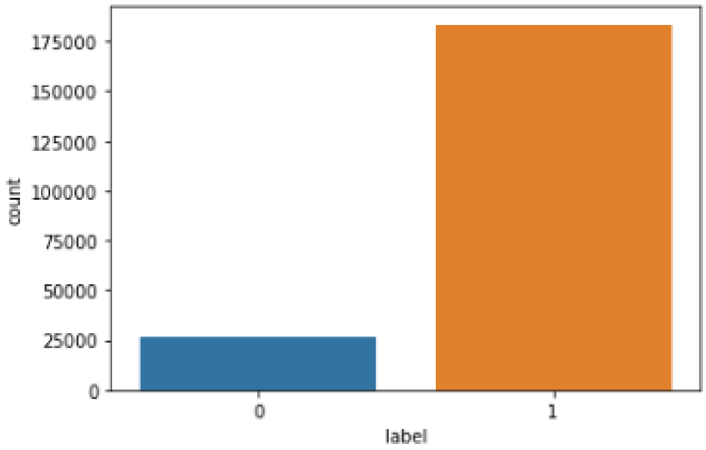


Now we have checked the attributes for the dataset and get a rough idea about the dataset like the no of rows & columns, datatype & null values in the dataset.

We don't have any null value in the dataset i.e. we don't have to deal with them. We will see whether the dataset is balance or not.

Target variable countplot

In [14]:
```
sns.countplot(df['label'])
```
Out[14]: <AxesSubplot:xlabel='label', ylabel='count'>



The dataset is imbalanced. Label '1' has approximately 87.5% records, while, label '0' has approximately 12.5% records.

Now, we see that the dataset is not balanced. The target has column has a large difference between both the labels. So, we have to make the dataset balanced for the proper ML model.
We will do that by using the SMOTE which will make some extra rows whose percentage is less in the dataset & make the counting of both the labels equal.

In [15]:
```
#separating the dependent & independent variable

x=df.iloc[:,1:]
y=df.iloc[:,0]
```

SMOTE

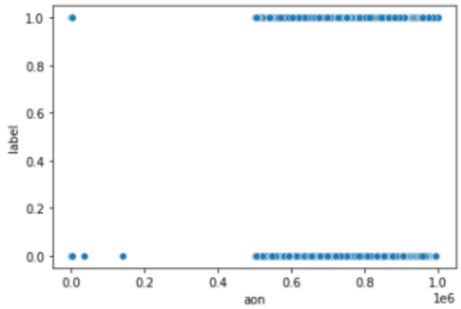
In [16]:
```
#balancing the data using SMOTE

from imblearn.over_sampling import SMOTE
smt= SMOTE(random_state=0,k_neighbors=1)
train_x,train_y=smt.fit_resample(x,y)
train_y.value_counts()
```
Out[16]:
```
0    183431
1    183431
Name: label, dtype: int64
```

Now the dataset is balance & we can proceed further.

EXPLORATORY DATA ANALYSIS

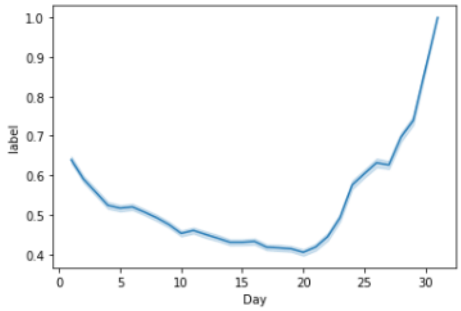
Visualizations

In [18]:
```
sns.scatterplot(df['aon'],df['label'])
```
Out[18]: <AxesSubplot:xlabel='aon', ylabel='label'>



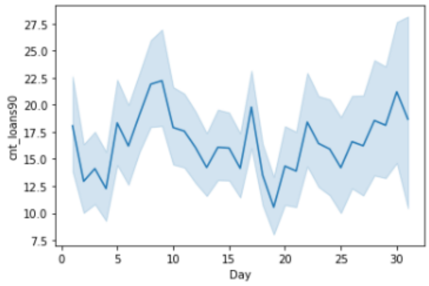
The data is on higher side

In [19]:
```
sns.lineplot(df['Day'],df['label'])
```
Out[19]: <AxesSubplot:xlabel='Day', ylabel='label'>



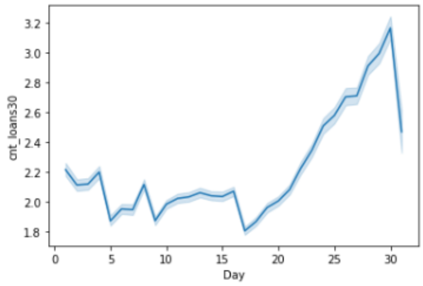
We have moreover the defaulters in the middle on the month.

In [20]:
```
sns.lineplot(df['Day'],df['cnt_loans90'])
```
Out[20]: <AxesSubplot:xlabel='Day', ylabel='cnt_loans90'>



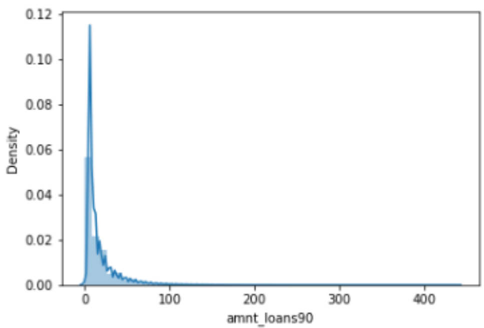
The count of loans over the 90 days w.r.t day of a month is varying. It is increasing and decreasing graph during the whole month and have many peaks & lows.

In [21]:
```
sns.lineplot(df['Day'],df['cnt_loans30'])
```
Out[21]: <AxesSubplot:xlabel='Day', ylabel='cnt_loans30'>

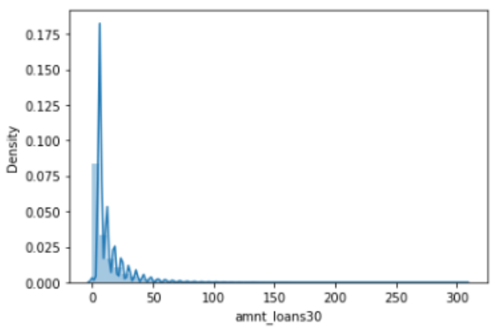


During last days of the month the count of credit increase sharply

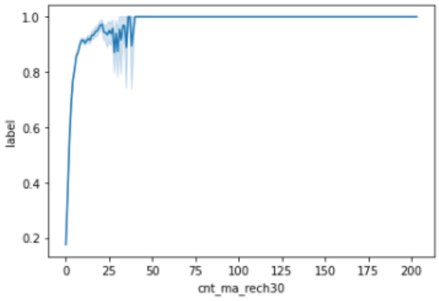
In [22]:
```
sns.distplot(df['amnt_loans90'])
```
Out[22]: <AxesSubplot:xlabel='amnt_loans90', ylabel='Density'>



In [23]:
```
sns.distplot(df['amnt_loans30'])
```
Out[23]: <AxesSubplot:xlabel='amnt_loans30', ylabel='Density'>

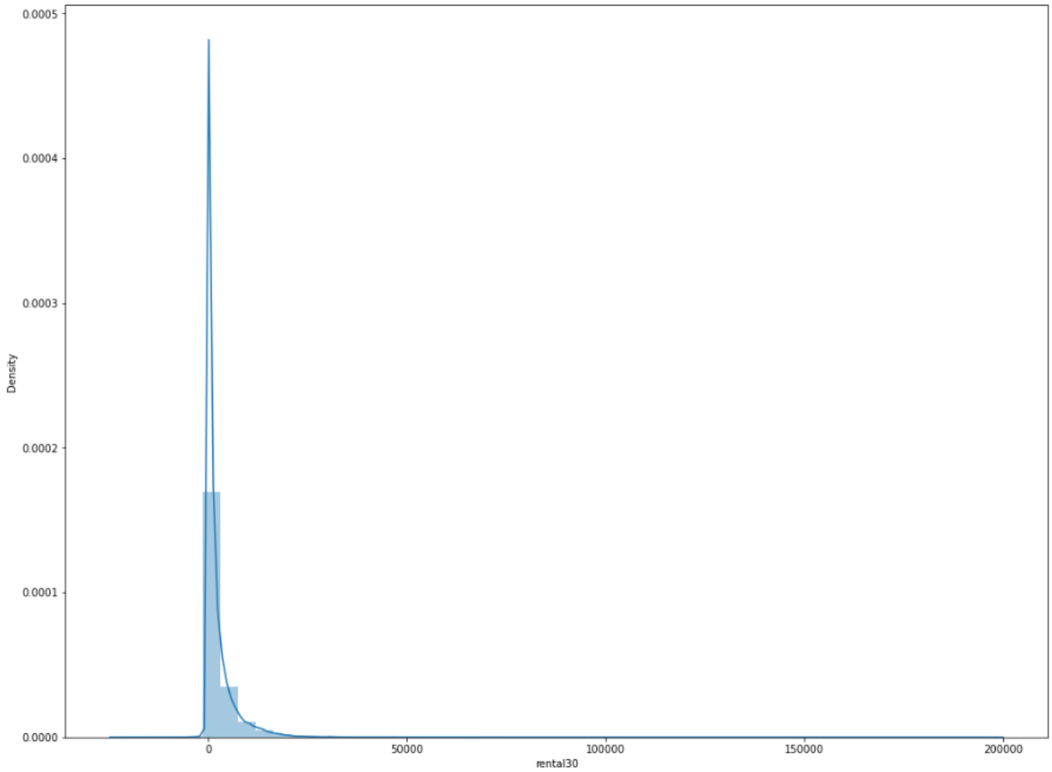


In [24]:
```
sns.lineplot(df['cnt_ma_rech30'],df['label'])
```
Out[24]: <AxesSubplot:xlabel='cnt_ma_rech30', ylabel='label'>



Peoples whose count of main account recharge over 30 days is on higher side are non-defaulters but with lower number are in the defaulters list

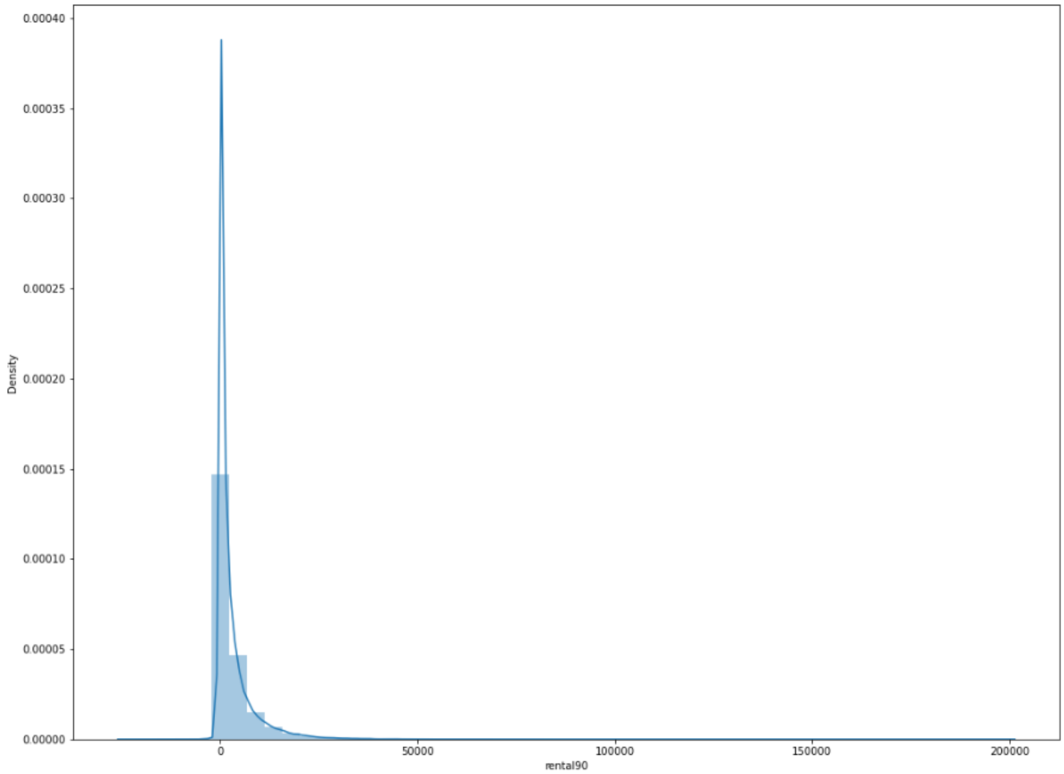
In [25]:
```
plt.figure(figsize=(17,13))
sns.distplot(df['rental30'])
```
Out[25]: <AxesSubplot:xlabel='rental30', ylabel='Density'>



The avg main account balance over the 30 days is lie between 0 to 50000 only

In [26]:
```
plt.figure(figsize=(17,13))
sns.distplot(df['rental90'])
```

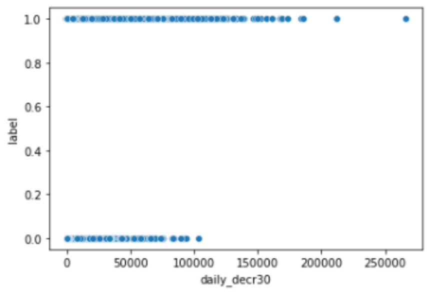
Out[26]: <AxesSubplot:xlabel='rental90', ylabel='Density'>



The avg main account balance over the 90 days is lie between 0 to 50000 only

In [27]:
```
sns.scatterplot(df['daily_decr30'],df['label'])
```

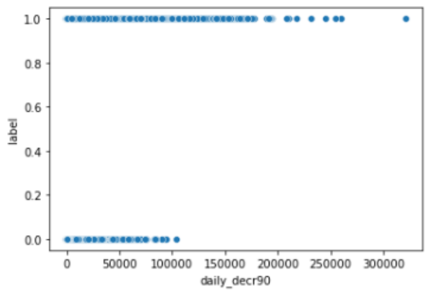
Out[27]: <AxesSubplot:xlabel='daily_decr30', ylabel='label'>



Defaulters daily spent from the mail account is max upto 100000 but it is higher for non-defaulters over the 30 days

In [28]:
```
sns.scatterplot(df['daily_decr90'],df['label'])
```

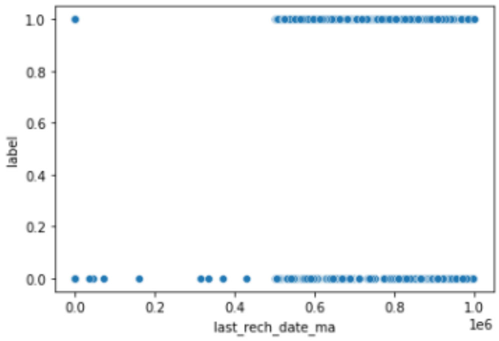
Out[28]: <AxesSubplot:xlabel='daily_decr90', ylabel='label'>



Defaulters daily spent from the mail account is max upto 100000 but it is higher for non-defaulters over the 90 days

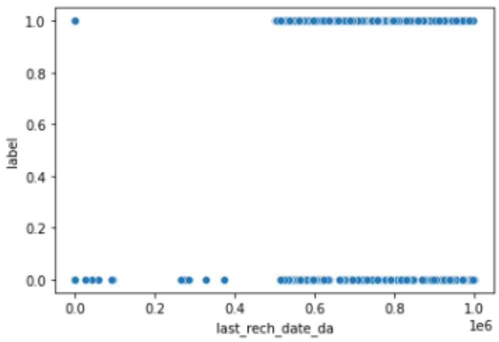
In [29]:
```
sns.scatterplot(df['last_rech_date_ma'],df['label'])
```

Out[29]: <AxesSubplot:xlabel='last_rech_date_ma', ylabel='label'>

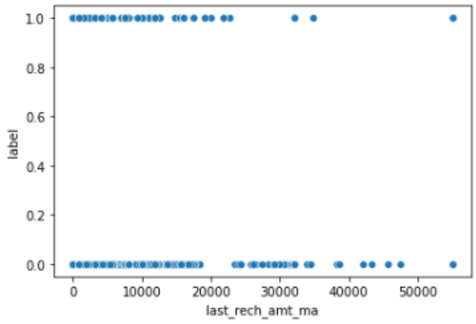


In [30]:
```
sns.scatterplot(df['last_rech_date_da'],df['label'])
```

Out[30]: <AxesSubplot:xlabel='last_rech_date_da', ylabel='label'>

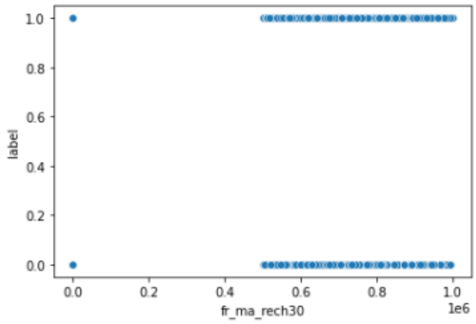


In [31]:
```
sns.scatterplot(df['last_rech_amt_ma'],df['label'])
```
Out[31]: <AxesSubplot:xlabel='last_rech_amt_ma', ylabel='label'>



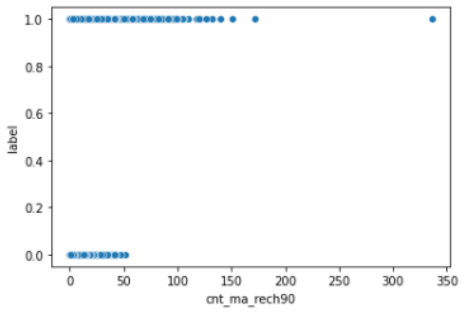
Defaulters last main account recharge amount is 0 to 40000 but for non defaulters it is 0 to 20000

In [32]:
```
sns.scatterplot(df['fr_ma_rech30'],df['label'])
```
Out[32]: <AxesSubplot:xlabel='fr_ma_rech30', ylabel='label'>



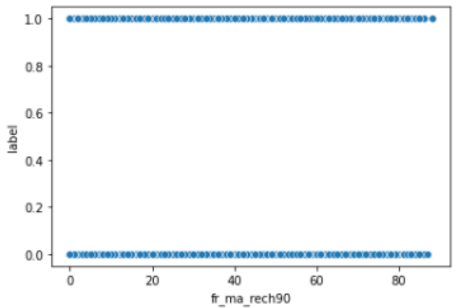
Frequency of main account recharge is on higher side

In [33]:
```
sns.scatterplot(df['cnt_ma_rech90'],df['label'])
```
Out[33]:
```
<AxesSubplot:xlabel='cnt_ma_rech90', ylabel='label'>
```



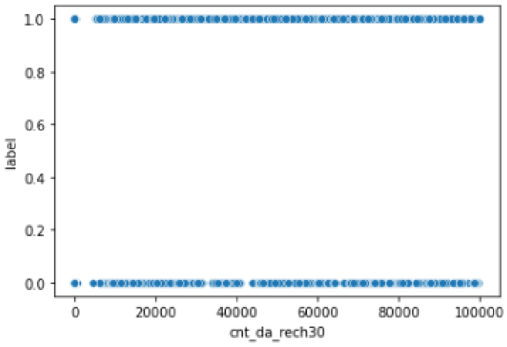
The count of the main account recharge over the 90 days is low for defaulters(0-50) & but for non-defaulters it is upto 150

In [34]:
```
sns.scatterplot(df['fr_ma_rech90'],df['label'])
```
Out[34]:
```
<AxesSubplot:xlabel='fr_ma_rech90', ylabel='label'>
```



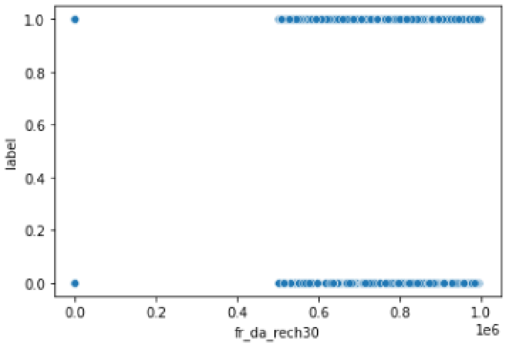
The data is distributed over the range. Frequency of main recharge account is rqually distributed for defaulyters as well as non-defaulters

In [35]:
```
sns.scatterplot(df['cnt_da_rech30'],df['label'])
```
Out[35]: <AxesSubplot:xlabel='cnt_da_rech30', ylabel='label'>



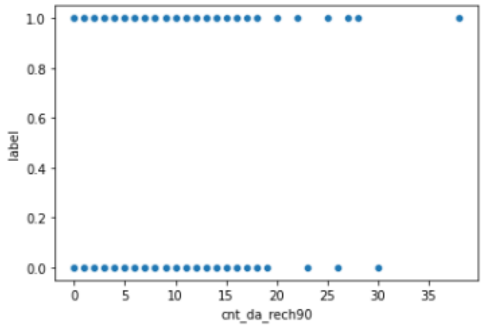
We have distributed data over the whole range of data. The count od data recharge account is very high over the 30 days

In [36]:
```
sns.scatterplot(df['fr_da_rech30'],df['label'])
```
Out[36]: <AxesSubplot:xlabel='fr_da_rech30', ylabel='label'>



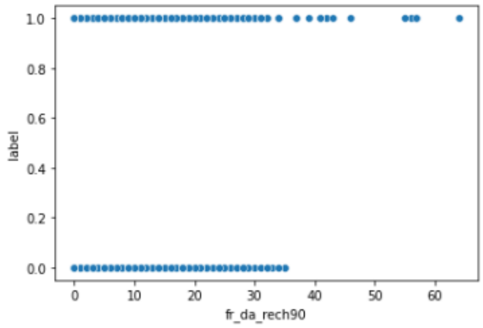
We have the data is on higher side i.e frequency is high over 30 days

In [37]:
```
sns.scatterplot(df['cnt_da_rech90'],df['label'])
```
Out[37]: <AxesSubplot:xlabel='cnt_da_rech90', ylabel='label'>



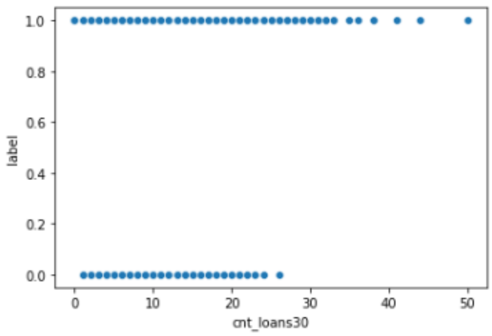
We have the distributed data over the range of 0 to 20 i.e. data recharge account count is distributed over defaulters or non-defaulters

In [38]:
```
sns.scatterplot(df['fr_da_rech90'],df['label'])
```
Out[38]: <AxesSubplot:xlabel='fr_da_rech90', ylabel='label'>



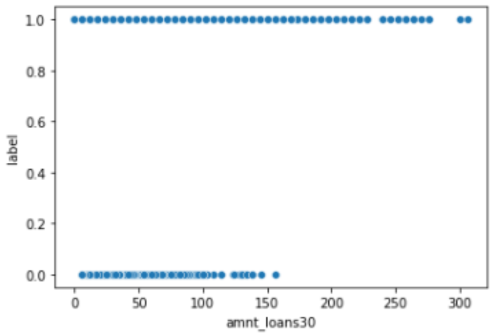
defaulters have very low frequency of data recharging over 90 days

In [39]:
```
sns.scatterplot(df['cnt_loans30'],df['label'])
```
Out[39]: <AxesSubplot:xlabel='cnt_loans30', ylabel='label'>



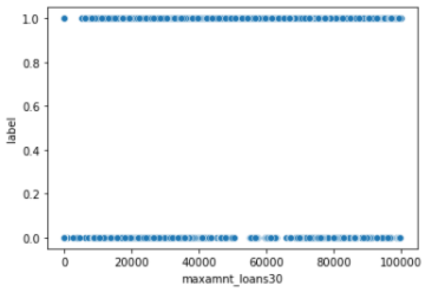
Person whose loan count is low is in deafulter list and ahve the outlers in it.

In [40]:
```
sns.scatterplot(df['amnt_loans30'],df['label'])
```
Out[40]: <AxesSubplot:xlabel='amnt_loans30', ylabel='label'>



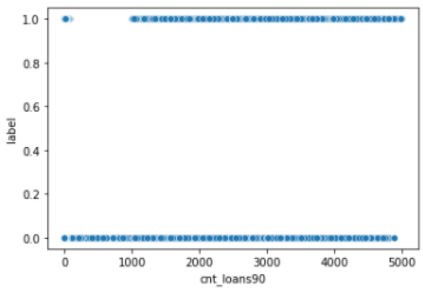
Here, a person with high loan amount is out of rist i.e non defaulter but with low amount is at risk i.e. defaulter

In [41]:
```
sns.scatterplot(df['maxamnt_loans30'],df['label'])
```
Out[41]: <AxesSubplot:xlabel='maxamnt_loans30', ylabel='label'>



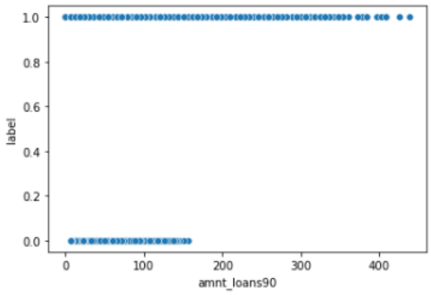
We have distributed data in this case. Either a person maximum amount value is low or high doen't affect too much its defaultibility nature over the last 30 days

In [43]:
```
sns.scatterplot(df['cnt_loans90'],df['label'])
```
Out[43]: <AxesSubplot:xlabel='cnt_loans90', ylabel='label'>



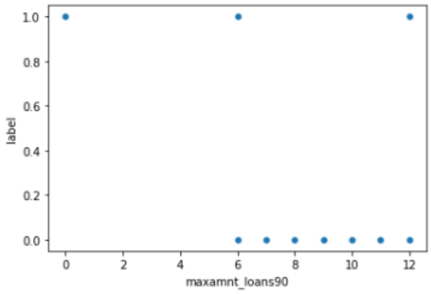
We have a distributed data in between 1000 to 5000

In [44]:
```
sns.scatterplot(df['amnt_loans90'],df['label'])
```
Out[44]: <AxesSubplot:xlabel='amnt_loans90', ylabel='label'>



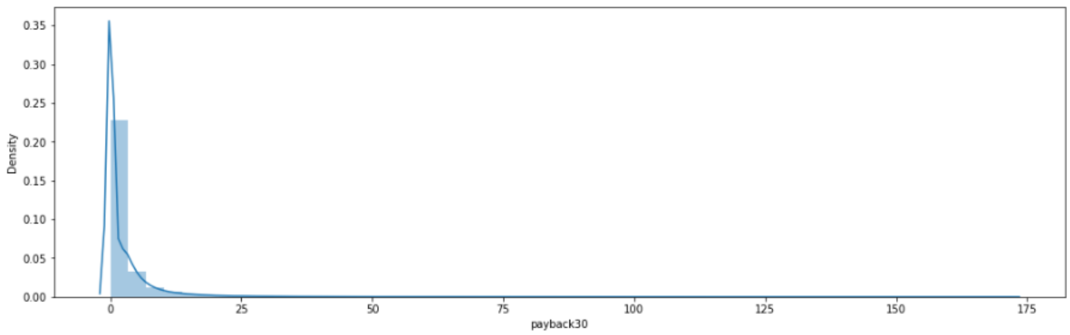
Peoples with high loan of amount are not in defaulter list but the ones who has an amount between 0 to 200 are in defaulter count

In [45]:
```
sns.scatterplot(df['maxamnt_loans90'],df['label'])
```
Out[45]: <AxesSubplot:xlabel='maxamnt_loans90', ylabel='label'>



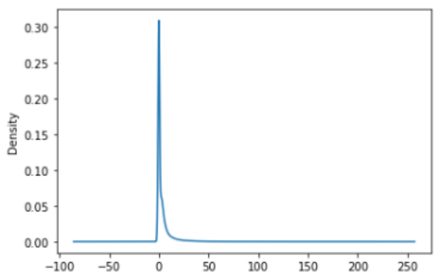
Peoples who takes a very high number of credit over last 90 days are in the deafulters list, a very few are there who are out of it.

In [52]:
```
plt.figure(figsize=(17,5))
sns.distplot(df['payback30'],bins=50)
```
Out[52]: <AxesSubplot:xlabel='payback30', ylabel='Density'>



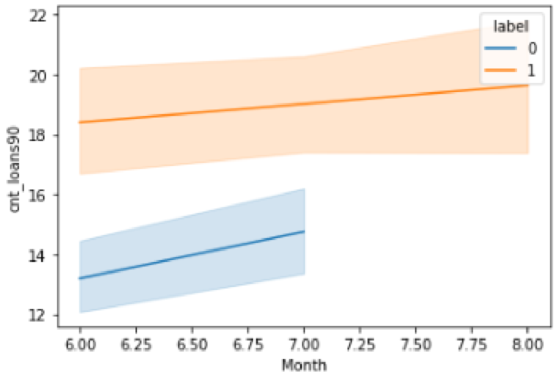
The paybacks over a 30 day cycle is on the earler side

In [51]:
```
df['payback90'].plot(kind='kde')
```
Out[51]: <AxesSubplot:ylabel='Density'>



Most of the payback has done in the early days

```
In [21]:  sns.lineplot(df['Month'],df['cnt_loans90'],hue=df['label'])
Out[21]: <AxesSubplot:xlabel='Month', ylabel='cnt_loans90'>
```



We have defaulters in 6th & 7th month whose total count of lons over 90 days around 12 to 16

Observations Made:

1. We have moreover the defaulters in the middle on the month.
2. The count of loans over the 90 days w.r.t day of a month is varying. It is increasing and decreasing graph during the whole month and have many peaks & lows.
3. During last days of the month the count of credit increase sharply
4. Peoples whose count of main account recharge over 30 days is on higher side are non-defaulters but with lower number are in the defaulters list
5. The average of main account balance over the 30 days is lie between 0 to 50000 only
6. The average of main account balance over the 90 days is lie between 0 to 50000 only
7. Defaulters daily spent from the main account is max up to 100000 but it is higher for non-defaulters over the 30 days
8. Defaulters daily spent from the main account is max up to 100000 but it is higher for non-defaulters over the 90 days
9. Defaulters last main account recharge amount is 0 to 40000 but for non-defaulters it is 0 to 20000
10. Frequency of main account recharge is on very higher either it is for defaulters or non-defaulters
11. The count of the main account recharge over the 90 days is low for defaulters (0-50) & but for non-defaulters it is up to 150
12. The data is distributed over the range. Frequency of main recharge account is equally distributed for defaulters as well as non-defaulters

13. We have distributed data over the whole range of data. The count of data recharge account is very high over the 30 days

14. We have the data is on higher side i.e. frequency is high for data account recharge over 30 days

15. We have the distributed data over the range of 0 to 20 i.e. data recharge account count is distributed over defaulters or non-defaulters

16. Defaulters have very low frequency of data recharging over 90 days

17. Person whose loan count is low is in defaulter list and have the outliers in it.

18. Here, a person with high loan amount is out of risk i.e. non defaulter but with low amount is at risk i.e. defaulter

19. We have distributed data in this case. Either a person maximum amount value is low or high doesn't affect too much its defaultibility nature over the last 30 days

20. We have a distributed data in between 1000 to 5000 for the count of loan over 90 days

21. Peoples with high loan of amount are not in defaulter list but the ones who has an amount between 0 to 200 are in defaulter count

22. Peoples who takes a very high number of credit over last 90 days are in the defaulters list, a very few are there who are out of it.

23. The paybacks over a 30-day cycle is on the earlier side

24. Most of the payback has done in the early days

25. We have defaulters in 6th & 7th month whose total count of loans over 90 days around 12 to 16

Statistical Summary & Correlation

We will describe the statistical summary of the dataset and find the correlation of each column.

Statistical Summary

In [22]:
```
df.describe()
```

Out[22]:

	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_rech3
count	366862.000000	366862.000000	366862.000000	366862.000000	366862.000000	366862.000000	366862.000000	366862.000000	366862.00000
mean	8353.341632	3606.937042	4010.135080	2402.875112	2977.747766	3400.343847	3739.374889	1703.388909	2.79874
std	77171.711019	7534.692924	8824.656092	4076.625836	5219.996216	51109.131274	53619.690818	2279.699716	3.70353
min	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	0.000000	0.00000
25%	217.270114	12.873338	13.106781	171.250000	184.673295	1.000000	0.000000	770.000000	0.00000
50%	471.613346	519.995322	526.000000	869.560000	1017.867085	3.000000	0.000000	947.000000	2.00000
75%	903.000000	3811.547200	3921.205477	2862.284364	3491.885000	8.000000	0.000000	1547.000000	4.00000
max	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	55000.000000	203.00000

Observations Made

- Now, we have 366862 rows after using the smote
- There is very larrge difference between the 75% and max value, means outliers are present in the dataset
- We have negative values also as our min value
- Some columns have difference between the mean, median, std so we can say that skewness is also present in the dataset

Correlation

In [23]:
```
corr=df.corr()
corr
```

Out[23]:

	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_rech3
aon	1.000000	-0.002630	-0.002785	-0.009281	-0.008683	-0.000829	-0.004128	-0.001496	-0.00789
daily_decr30	-0.002630	1.000000	0.979945	0.386044	0.418857	0.000417	-0.002329	0.293017	0.50432
daily_decr90	-0.002785	0.979945	1.000000	0.378815	0.427874	0.001035	-0.002377	0.280455	0.48210
rental30	-0.009281	0.386044	0.378815	1.000000	0.964804	-0.003005	0.003249	0.146020	0.21295
rental90	-0.008683	0.418857	0.427874	0.964804	1.000000	-0.002650	0.002900	0.146448	0.22748
last_rech_date_ma	-0.000829	0.000417	0.001035	-0.003005	-0.002650	1.000000	0.000023	-0.003165	0.00301
last_rech_date_da	-0.004128	-0.002329	-0.002377	0.003249	0.002900	0.000023	1.000000	-0.003317	-0.00402
last_rech_amt_ma	-0.001496	0.293017	0.280455	0.146020	0.146448	-0.003165	-0.003317	1.000000	0.09785
cnt_ma_rech30	-0.007890	0.504323	0.482104	0.212955	0.227480	0.003017	-0.004027	0.097855	1.00000

Observations Made

- Now, we have 366862 rows after using the smote
- There is very large difference between the 75% and max value, means outliers are present in the dataset
- We have negative values also as our min value
- Some columns have difference between the mean, median, std so we can say that skewness is also present in the dataset

Plotting & Removing the Outliers

We have some outliers present in the dataset, so let's handle them also. As the outliers in the dataset will affect our ML model. We need to remove all the outliers present in the dataset.

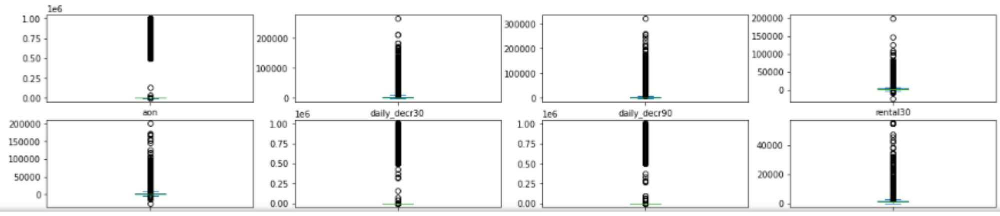
There is something called zscore which indicates how many standard deviations away an element is from the mean. We consider the points as outliers whose zscore is above 3 or less than -3. So we need to remove all such points from our dataset.

Using the threshold, we have removed all the points where the zscore is greater than 3. Now the total number of rows after removing the outliers are 302199.

Plotting the outliers

```
In [27]: df.plot(kind='box',subplots=True,layout=(9,4),figsize=(20,20))
```

```
medianamnt_loans90      AxesSubplot(0.327174,0.210472;0.168478x0.0712264)
payback30               AxesSubplot(0.529348,0.210472;0.168478x0.0712264)
payback90               AxesSubplot(0.731522,0.210472;0.168478x0.0712264)
Month                          AxesSubplot(0.125,0.125;0.168478x0.0712264)
Day                         AxesSubplot(0.327174,0.125;0.168478x0.0712264)
label                       AxesSubplot(0.529348,0.125;0.168478x0.0712264)
dtype: object
```



We have outliers in the dataset which has to be handled

```
In [30]: #removing the outliers from the data

         from scipy.stats import zscore
         z=np.abs(zscore(df))
         threshold=3
         print(np.where(z>3))
         df_new=df[(z<3).all(axis=1)]
         df=df_new
         df.shape
```

```
(array([     7,     24,     24, ..., 366852, 366852, 366852], dtype=int64), array([20,  3,  4, ...,  4, 25, 29], dtype=int6
4))
```

```
Out[30]: (302199, 35)
```

Now, our data cleaning & visualization part is done and we proceed with the model building.

MODEL BUILDING

We will import important libraries for the building the ML model and defining the different models for our easiness.

Finding the best random state for the train test split.

Model Building

```
In [33]: #importing the different machine learning models

         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
         from sklearn.model_selection import train_test_split
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.neighbors import KNeighborsClassifier
```

```
In [34]: #defining the models

         lg=LogisticRegression()
         rdc=RandomForestClassifier()
         dtc=DecisionTreeClassifier()
         knc=KNeighborsClassifier()
```

Finding the best random state

```
In [35]: model=[lg,rdc,svc,dtc,knc]
         maxAccu=0
         bestRS=0
         for i in range(40,60):
             x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=i,test_size=.30)
             lg.fit(x_train,y_train)
             pred=lg.predict(x_test)
             acc=accuracy_score(y_test,pred)
             if acc>maxAccu:
                 maxAccu=acc
                 bestRS=i
         print('Best Accuracy score is', maxAccu , 'on random state', bestRS)

         Best Accuracy score is 0.8794013804510321 on random state 51
```

```
In [36]: x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=51,test_size=.30)
```

Classification Algorithms

We have use six different regression algorithms to find the best model for our problem.

- **Logistic Regression**
- from sklearn.linear_model import LogisticRegression
- **Decision Tree Classifier**
- from sklearn.tree import DecisionTreeClassifier
- **Kneighbor Classifier**

- from sklearn.neighbors import KNeighborsClassifier
- **Random Forest Classifier**
- from sklearn.ensemble import RandomForestClassifier
- **Gaussian NB**
- from sklearn.naive_bayes import GaussianNB
- **SGD Classifier**
- from sklearn.linear_model import SGDClassifier

Logistic regression

In [37]:
```
lg.fit(x_train,y_train)
pred=lg.predict(x_test)
acc=accuracy_score(y_test,pred)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred))
print('Classification Report: ','\n',classification_report(y_test,pred))
```

```
Accuracy Score:  0.8794013804510321
Confusion Matrix:
 [[  269  7456]
 [   127 55026]]
Classification Report:
               precision    recall  f1-score   support

           0       0.68      0.03      0.07      7725
           1       0.88      1.00      0.94     55153

    accuracy                           0.88     62878
   macro avg       0.78      0.52      0.50     62878
weighted avg       0.86      0.88      0.83     62878
```

Decision Tree Classifier

In [38]:
```
dtc.fit(x_train,y_train)
pred=dtc.predict(x_test)
acc=accuracy_score(y_test,pred)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred))
print('Classification Report: ','\n',classification_report(y_test,pred))
```

```
Accuracy Score:  0.886876172906263
Confusion Matrix:
 [[ 4351  3374]
 [ 3739 51414]]
Classification Report:
               precision    recall  f1-score   support

           0       0.54      0.56      0.55      7725
           1       0.94      0.93      0.94     55153

    accuracy                           0.89     62878
   macro avg       0.74      0.75      0.74     62878
weighted avg       0.89      0.89      0.89     62878
```

In [39]:
```
rdc.fit(x_train,y_train)
pred=rdc.predict(x_test)
acc=accuracy_score(y_test,pred)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred))
print('Classification Report: ','\n',classification_report(y_test,pred))
```

```
Accuracy Score:  0.9220872165145202
Confusion Matrix: 
 [[ 3959  3766]
 [ 1133 54020]]
Classification Report: 
               precision    recall  f1-score   support

           0       0.78      0.51      0.62      7725
           1       0.93      0.98      0.96     55153

    accuracy                           0.92     62878
   macro avg       0.86      0.75      0.79     62878
weighted avg       0.92      0.92      0.91     62878
```

Kneighbor Classifier

In [40]:
```
knc.fit(x_train,y_train)
pred=knc.predict(x_test)
acc=accuracy_score(y_test,pred)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred))
print('Classification Report: ','\n',classification_report(y_test,pred))
```

```
Accuracy Score:  0.8904545309965329
Confusion Matrix: 
 [[ 3243  4482]
 [ 2406 52747]]
Classification Report: 
               precision    recall  f1-score   support

           0       0.57      0.42      0.48      7725
           1       0.92      0.96      0.94     55153

    accuracy                           0.89     62878
   macro avg       0.75      0.69      0.71     62878
weighted avg       0.88      0.89      0.88     62878
```

Gaussian NB

In [43]:
```
from sklearn.naive_bayes import GaussianNB
gnb=GaussianNB()
gnb.fit(x_train,y_train)
pred=gnb.predict(x_test)
acc=accuracy_score(y_test,pred)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred))
print('Classification Report: ','\n',classification_report(y_test,pred))
```

```
Accuracy Score:  0.5920671777092147
Confusion Matrix: 
 [[ 6720  1005]
 [24645 30508]]
Classification Report: 
               precision    recall  f1-score   support

           0       0.21      0.87      0.34      7725
           1       0.97      0.55      0.70     55153

    accuracy                           0.59     62878
   macro avg       0.59      0.71      0.52     62878
weighted avg       0.88      0.59      0.66     62878
```

SGD Classifier

In [44]:
```
from sklearn.linear_model import SGDClassifier
sgd=SGDClassifier()
sgd.fit(x_train,y_train)
pred=sgd.predict(x_test)
acc=accuracy_score(y_test,pred)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred))
print('Classification Report: ','\n',classification_report(y_test,pred))
```

```
Accuracy Score:  0.8763160405865327
Confusion Matrix: 
 [[    8  7717]
 [   60 55093]]
Classification Report: 
               precision    recall  f1-score   support

           0       0.12      0.00      0.00      7725
           1       0.88      1.00      0.93     55153

    accuracy                           0.88     62878
   macro avg       0.50      0.50      0.47     62878
weighted avg       0.78      0.88      0.82     62878
```

Hence, we are getting the best accuracy score through the Random Forest Classifier Model. We will go ahead with this to find the cross val score and hypermeter tuning.

Cross Val Score & Hypermeter Tuning

Cross-validation provides information about how well a classifier generalizes, specifically the range of expected errors of the classifier. Cross Val Score tells how the model is generalized at a particular cross validation.

At CV=3 we get the best results i.e. the Random Forest Classifier more generalized at cv=3, so we calculate the hyper parameters at this value.

We will find which parameters of random forest classifier are the best foe our model. We will do this using Grid Search CV method & also calculate the accuracy score at those best parameters.

Cross Val Score

In [45]:
```
from sklearn.model_selection import cross_val_score
for i in range(3,7):
    cr=cross_val_score(rdc,x,y,cv=i)
    cr_mean=cr.mean()
    print("at cv= ", i)
    print('cross val score = ',cr_mean*100)
```

```
at cv=  3
cross val score =  92.09801871334051
at cv=  4
cross val score =  92.12282863375371
at cv=  5
cross val score =  92.10660693821755
at cv=  6
cross val score =  92.14954605860824
```

Hypermeter Tuning

In [47]:
```
from sklearn.model_selection import GridSearchCV
# creating parameters
param={'criterion':['gini','entropy'],
       'max_features':['sqrt','log2','auto'],
       'max_depth':[2,3,4,5]}

GCV=GridSearchCV(rdc,param,cv=6,scoring='accuracy')
GCV.fit(x_train,y_train)
GCV.best_params_
```

Out[47]:
```
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt'}
```

In [48]:
```
GCV_pred=GCV.best_estimator_.predict(x_test)
accuracy_score(y_test,GCV_pred)
```

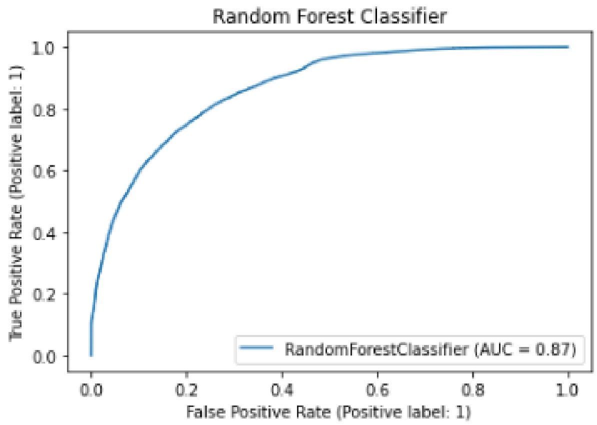
Out[48]:
```
0.9025255256210439
```

AUC ROC Curve

In our model AUC>0.5, so there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values.

AUC ROC Curve

In [49]:
```
from sklearn.metrics import plot_roc_curve
plot_roc_curve(GCV.best_estimator_,x_test,y_test)
plt.title('Random Forest Classifier')
plt.show()
```



Saving the Model

Saving the best model – Random Forest Classifier in this case for future predictions. Let’s see what are the actual test data and what our model predicts.

Saving the model

In [50]:
```
import pickle
filename='micro_credit.pkl'
pickle.dump(lg, open(filename,'wb'))
```

Conclusion

In [52]:
```
a=np.array(y_test)
pred=np.array(GCV_pred)
micro_credit=pd.DataFrame({'Actual':a,'Predicted':pred})
micro_credit
```

2	1	1
3	1	1
4	1	1
5	1	1
6	1	1
7	1	1
8	0	1
9	1	1
10	1	1
11	1	1
12	1	1
13	1	1
14	1	1
15	1	1

Here, we check the actual values versus predicted values to have a lookon our model and somehow we got an idea that our model predicts well.

Hence up to some good extensions our model predicted so well.

CONCLUSION

Conclusion of the Study

The results of this study suggest following outputs which might be useful for the company to improve the selection of customers & further investment for the credit:

- There are lot of things that is going to decide whether the customer is going to be a defaulter or not. As we see above in our visualizations, a lot of things like count of loans taken, maximum amount of loan, last data account recharge, last main account recharge and many more. One needs to analyse every aspect to have good hands on the prediction of the label.
- With the machine learning it become easier to predict the label but yes it is not 100% accurate, it provides an idea and accordingly we can analyse the situation and prepare the strategies to improve the selection of customers.

- ## Learning Outcomes of the Study in respect of Data Science
  - I got to know the different factors on which we can determine whether the customer is going to pay the credit in 5 days or not.
  - It was fun to deal with this project and learn different algorithms and how hypermeter tuning can refine our model.
  - It was difficult to handle so much columns simultaneously but yes every difficulty leads to the new things.