In [1]:
```
!pip install imblearn
```

```
Requirement already satisfied: imblearn in c:\users\rakesh lodem\anaconda3\lib\site-packages (0.0)
Requirement already satisfied: imbalanced-learn in c:\users\rakesh lodem\anaconda3\lib\site-packages (from imblea
rn) (0.9.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\rakesh lodem\anaconda3\lib\site-packages (from im
balanced-learn->imblearn) (2.1.0)
Requirement already satisfied: scipy>=1.3.2 in c:\users\rakesh lodem\anaconda3\lib\site-packages (from imbalanced
-learn->imblearn) (1.6.2)
Requirement already satisfied: scikit-learn>=1.1.0 in c:\users\rakesh lodem\anaconda3\lib\site-packages (from imb
alanced-learn->imblearn) (1.1.2)
Requirement already satisfied: joblib>=1.0.0 in c:\users\rakesh lodem\anaconda3\lib\site-packages (from imbalance
d-learn->imblearn) (1.0.1)
Requirement already satisfied: numpy>=1.17.3 in c:\users\rakesh lodem\anaconda3\lib\site-packages (from imbalance
d-learn->imblearn) (1.20.1)
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
df=pd.read_csv(r'C:\Users\RAKESH~1\AppData\Local\Temp\Rar$DIa3004.14953\Data file.csv')
df.head()
```

Out[3]:

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

5 rows × 37 columns

In [4]:
```
df.shape
```

Out[4]:
```
(209593, 37)
```

In [5]:
```
df.dtypes
```

Out[5]:
```
Unnamed: 0               int64
label                    int64
msisdn                  object
aon                    float64
daily_decr30           float64
daily_decr90           float64
rental30               float64
rental90               float64
last_rech_date_ma      float64
last_rech_date_da      float64
last_rech_amt_ma         int64
cnt_ma_rech30            int64
fr_ma_rech30           float64
sumamnt_ma_rech30      float64
medianamnt_ma_rech30   float64
medianmarechprebal30   float64
cnt_ma_rech90            int64
fr_ma_rech90             int64
sumamnt_ma_rech90        int64
medianamnt_ma_rech90   float64
medianmarechprebal90   float64
cnt_da_rech30          float64
```

```
fr_da_rech30            float64
cnt_da_rech90             int64
fr_da_rech90              int64
cnt_loans30               int64
amnt_loans30              int64
maxamnt_loans30         float64
medianamnt_loans30      float64
cnt_loans90             float64
amnt_loans90              int64
maxamnt_loans90           int64
medianamnt_loans90      float64
payback30               float64
payback90               float64
pcircle                  object
pdate                    object
dtype: object
```

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
Unnamed: 0              0
label                   0
msisdn                  0
aon                     0
daily_decr30            0
daily_decr90            0
rental30                0
rental90                0
last_rech_date_ma       0
last_rech_date_da       0
last_rech_amt_ma        0
cnt_ma_rech30           0
fr_ma_rech30            0
sumamnt_ma_rech30       0
medianamnt_ma_rech30    0
medianmarechprebal30    0
cnt_ma_rech90           0
fr_ma_rech90            0
sumamnt_ma_rech90       0
medianamnt_ma_rech90    0
medianmarechprebal90    0
cnt_da_rech30           0
fr_da_rech30            0
cnt_da_rech90           0
fr_da_rech90            0
cnt_loans30             0
amnt_loans30            0
maxamnt_loans30         0
medianamnt_loans30      0
cnt_loans90             0
amnt_loans90            0
maxamnt_loans90         0
medianamnt_loans90      0
payback30               0
payback90               0
pcircle                 0
pdate                   0
dtype: int64
```

In [7]:

```
df=df.drop(['Unnamed: 0','msisdn'],axis=1)
df.head()
```

Out[7]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_rech30	...	ma
0	0	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	2	...	
1	1	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	1	...	
2	1	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	1	...	
3	1	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	0	...	
4	1	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	7	...	

5 rows × 35 columns

In [8]:

```
df.dtypes
```

Out[8]:

```
label                     int64
aon                     float64
daily_decr30            float64
daily_decr90            float64
rental30                float64
rental90                float64
last_rech_date_ma       float64
last_rech_date_da       float64
last_rech_amt_ma          int64
cnt_ma_rech30             int64
fr_ma_rech30            float64
sumamnt_ma_rech30       float64
medianamnt_ma_rech30    float64
medianmarechprebal30    float64
cnt_ma_rech90             int64
fr_ma_rech90              int64
sumamnt_ma_rech90         int64
medianamnt_ma_rech90    float64
medianmarechprebal90    float64
cnt_da_rech30           float64
fr_da_rech30            float64
cnt_da_rech90             int64
fr_da_rech90              int64
cnt_loans30               int64
amnt_loans30              int64
maxamnt_loans30         float64
medianamnt_loans30      float64
cnt_loans90             float64
amnt_loans90              int64
maxamnt_loans90           int64
medianamnt_loans90      float64
payback30               float64
payback90               float64
pcircle                  object
pdate                    object
dtype: object
```

In [9]:

```
df.describe()
```

Out[9]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_r
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	20
mean	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921	
std	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430	
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	
25%	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000	
50%	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000	
75%	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000	
max	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	5

8 rows × 33 columns

In [10]:

```
cat_df=df.select_dtypes(exclude=['object'])
cat_df
```

Out[10]:

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

209590	1	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	0.0	1539	5
209591	1	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	38.0	773	5
209592	1	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	0.0	7526	2

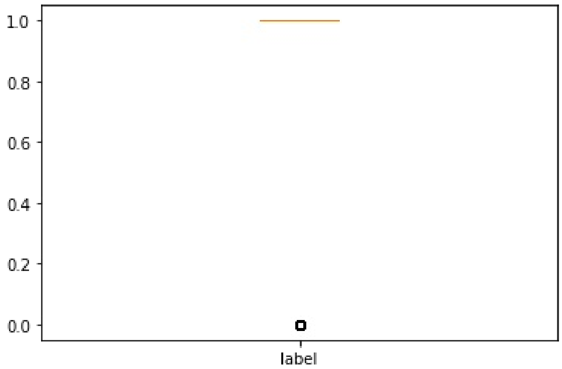
209593 rows × 33 columns

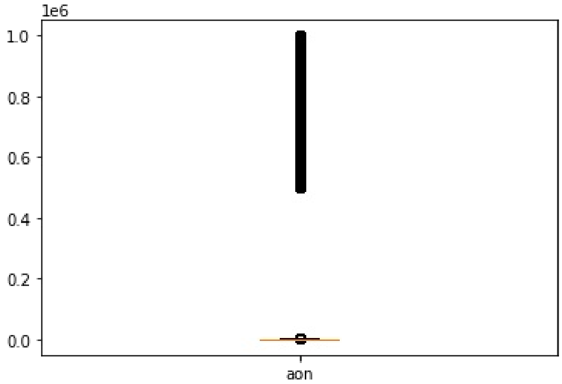
In [11]:

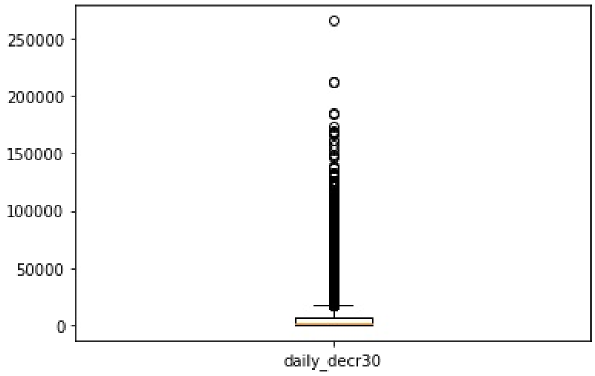
```
## detecting the outliers
```

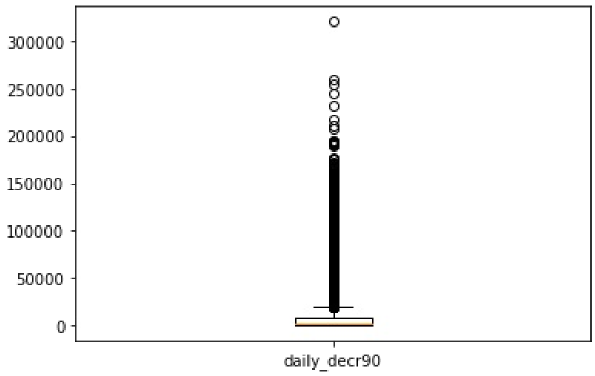
In [12]:

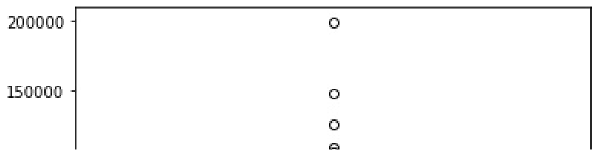
```
for i in cat_df:
    plt.boxplot(cat_df[i], labels = [i])
    plt.show()
```

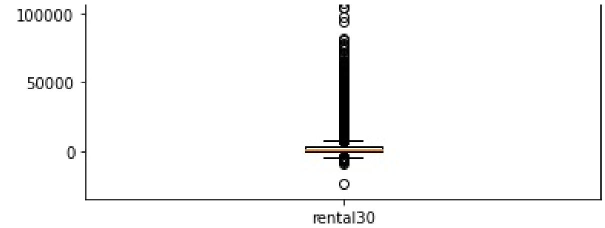


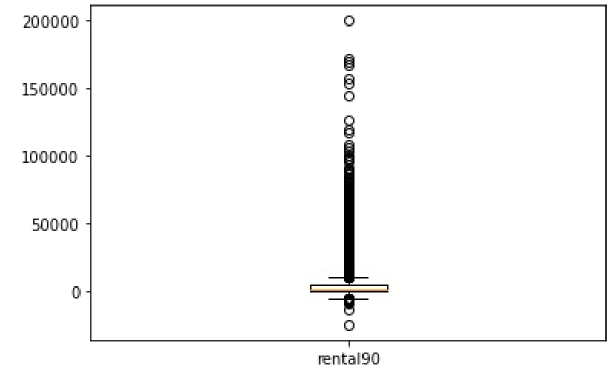


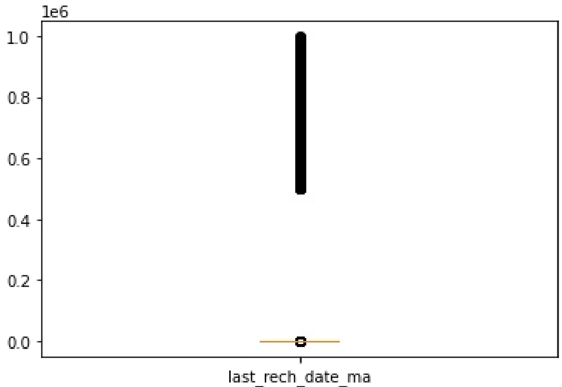


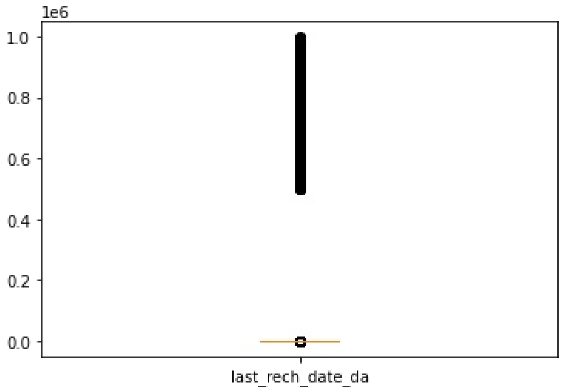


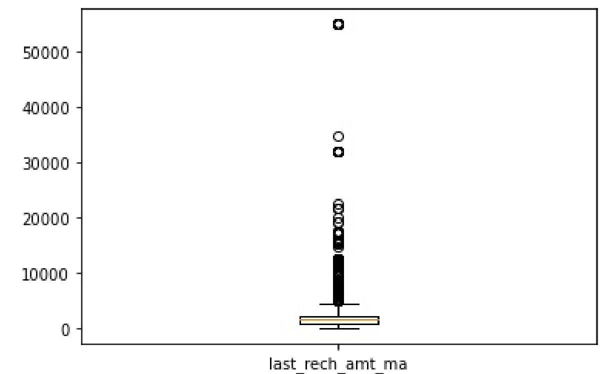


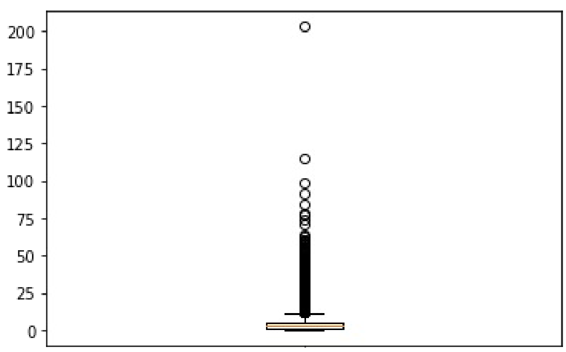

100000
50000
0
rental30


200000
150000
100000
50000
0
rental90

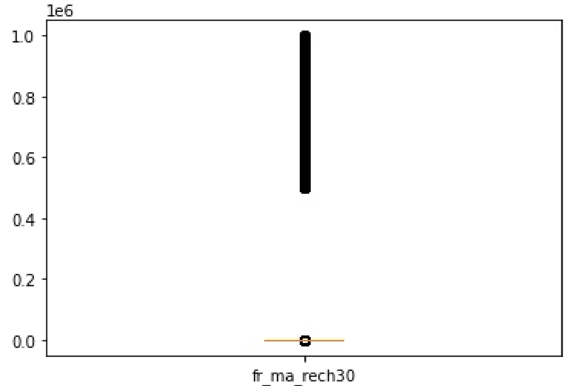

1e6
1.0
0.8
0.6
0.4
0.2
0.0
last_rech_date_ma

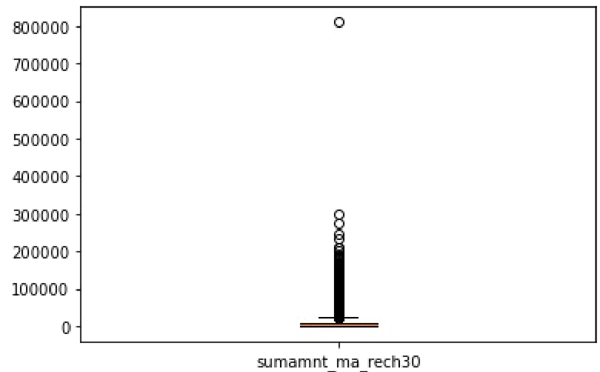

1e6
1.0
0.8
0.6
0.4
0.2
0.0
last_rech_date_da

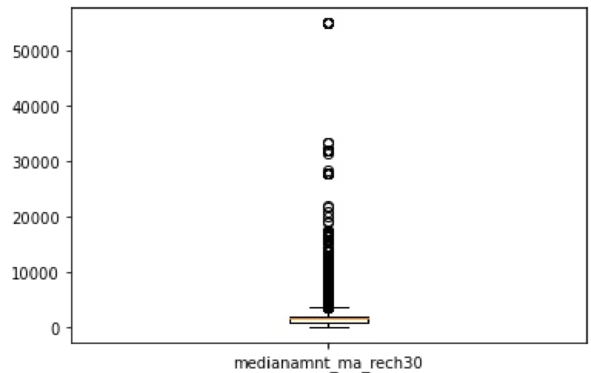

50000
40000
30000
20000
10000
0
last_rech_amt_ma

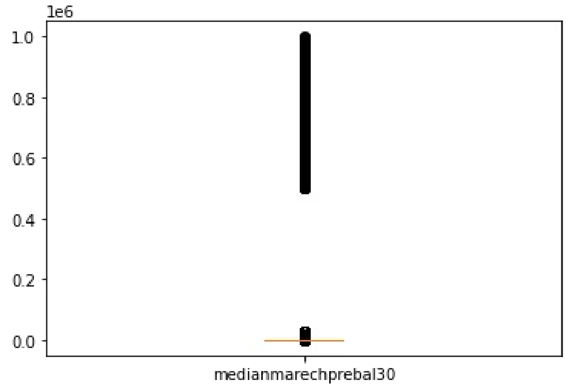

200
175
150
125
100
75
50
25
0

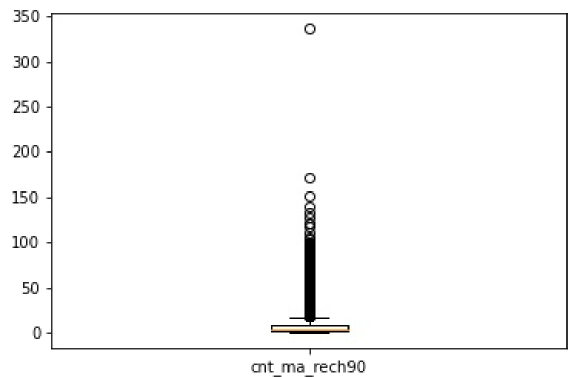
cnt_ma_rech30

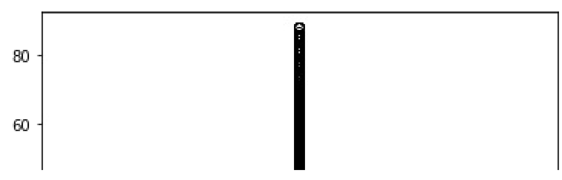


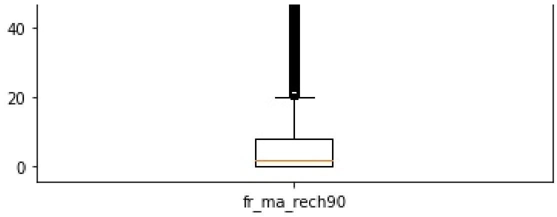


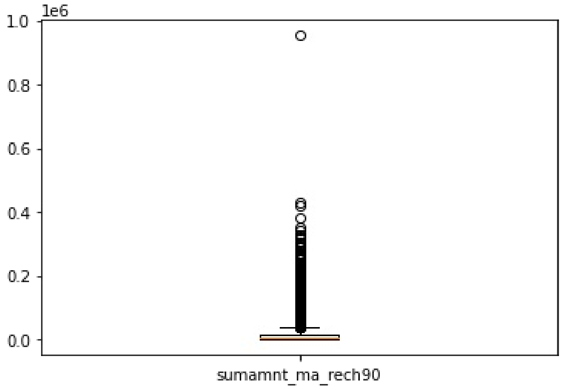


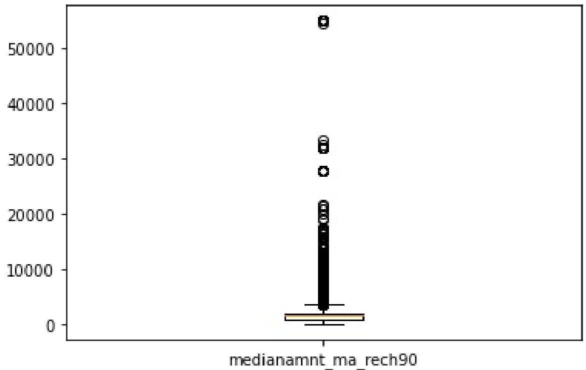


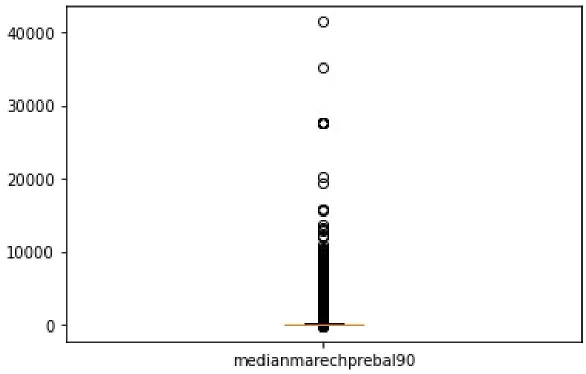


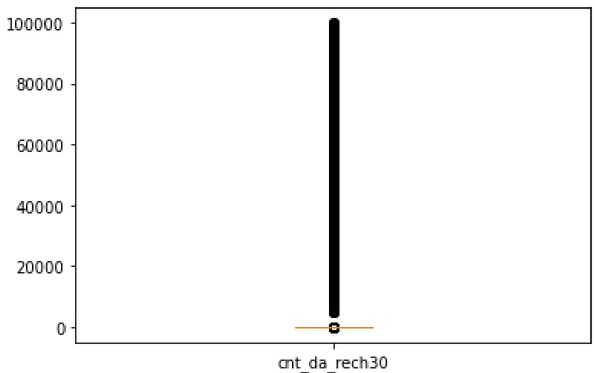


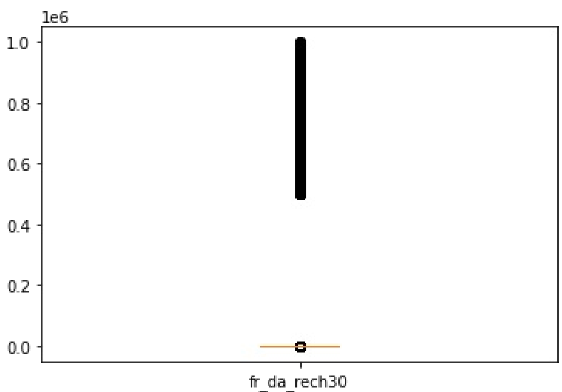

0
20
40
fr_ma_rech90

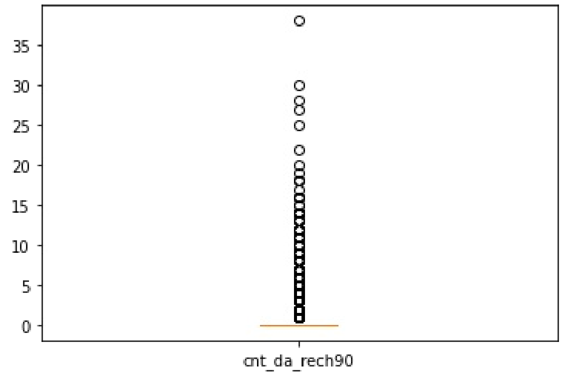

1e6
0.0
0.2
0.4
0.6
0.8
1.0
sumamnt_ma_rech90

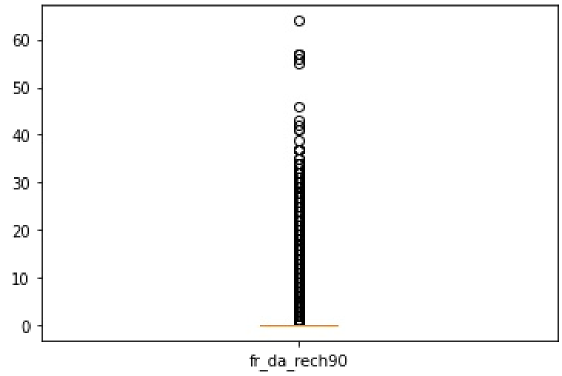

0
10000
20000
30000
40000
50000
medianamnt_ma_rech90

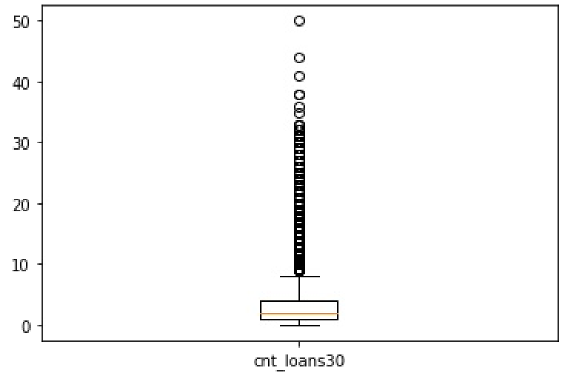

0
10000
20000
30000
40000
medianmarechprebal90

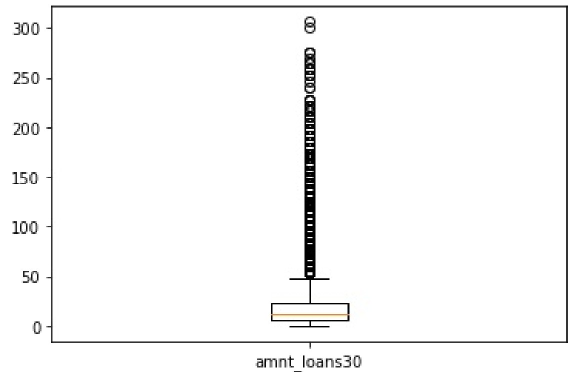

0
20000
40000
60000
80000
100000
cnt_da_rech30

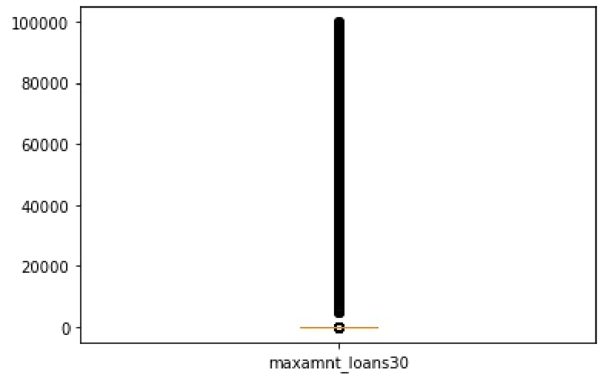

1e6
0.0
0.2
0.4
0.6
0.8
1.0
fr_da_rech30

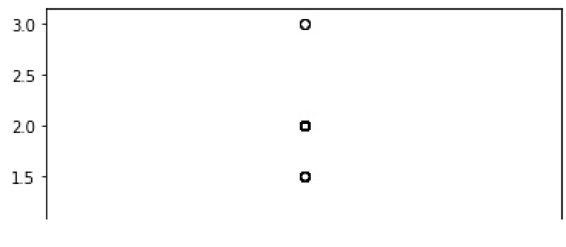

35
30
25
20
15
10
5
0
cnt_da_rech90

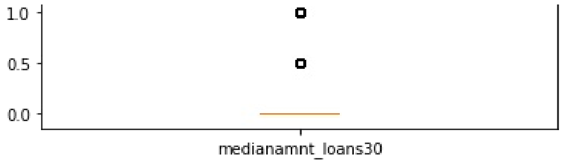

60
50
40
30
20
10
0
fr_da_rech90

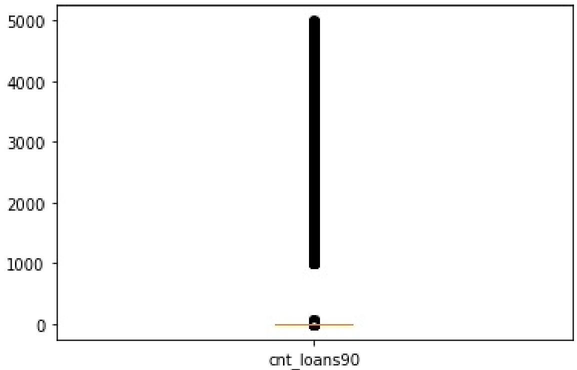

50
40
30
20
10
0
cnt_loans30

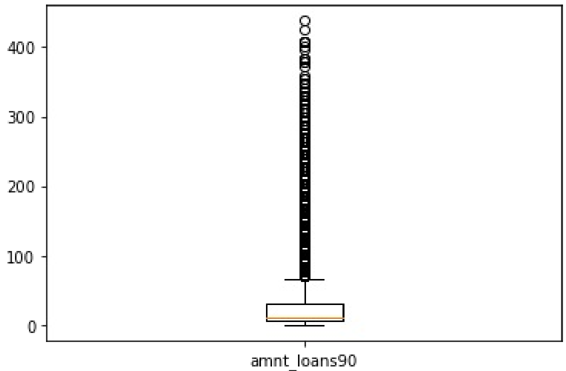

300
250
200
150
100
50
0
amnt_loans30

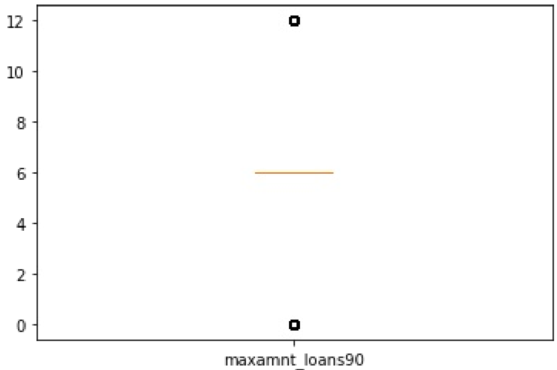

100000
80000
60000
40000
20000
0
maxamnt_loans30

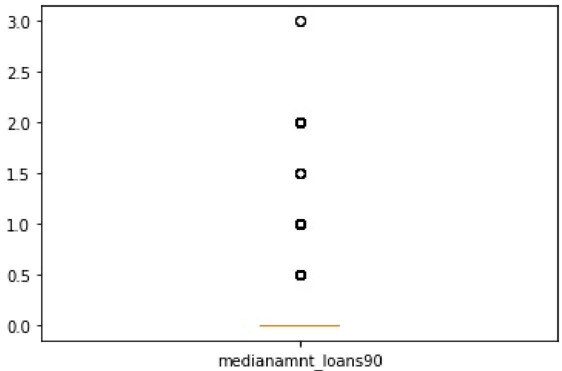

3.0
2.5
2.0
1.5

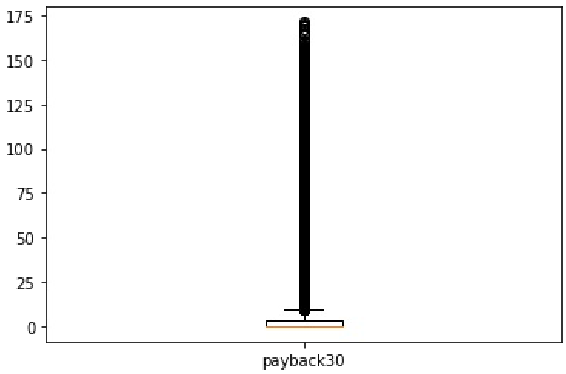

1.0
0.5
0.0
medianamnt_loans30

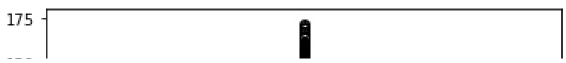

5000
4000
3000
2000
1000
0
cnt_loans90

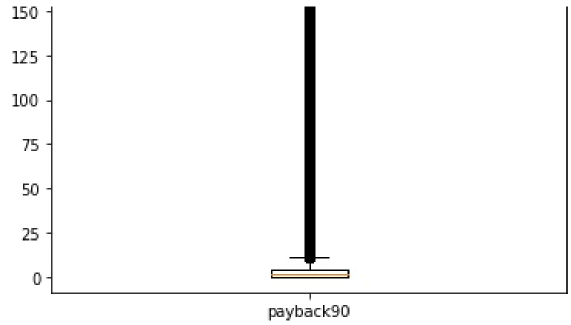

400
300
200
100
0
amnt_loans90


12
10
8
6
4
2
0
maxamnt_loans90


3.0
2.5
2.0
1.5
1.0
0.5
0.0
medianamnt_loans90


175
150
125
100
75
50
25
0
payback30


175



In [13]:
```
## removing the outliers using iqr
```

In [14]:
```
out_vars=['aon','daily_decr30','daily_decr90','rental30','rental90','last_rech_date_ma','last_rech_date_da','last
```

In [15]:
```
def outlierTreat(x):
    upper = x.quantile(.75) + 1.5 * (x.quantile(.75) - x.quantile(.25))
    lower = x.quantile(.25) - 1.5 * (x.quantile(.75) - x.quantile(.25))
    return x.clip(lower, upper)
```

In [16]:
```
cat_df.loc[:, out_vars] = cat_df.loc[:, out_vars].apply(outlierTreat)
cat_df.loc[:, out_vars].describe()
```

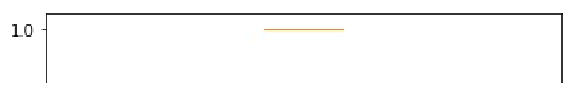
Out[16]:

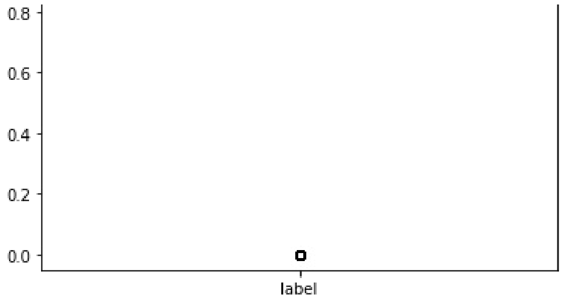
	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cr
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.0	209593.000000	2
mean	668.405825	4468.105346	4855.261718	2197.069184	2769.147968	4.837991	0.0	1719.109131	
std	514.130937	5859.401770	6438.219389	2552.708718	3251.352605	5.071356	0.0	1345.846661	
min	-48.000000	-93.012667	-93.012667	-4334.360000	-5552.035000	-8.000000	0.0	0.000000	
25%	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.0	770.000000	
50%	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.0	1539.000000	
75%	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.0	2309.000000	
max	2086.000000	18046.340000	19442.937000	7971.720000	10054.085000	16.000000	0.0	4617.500000	

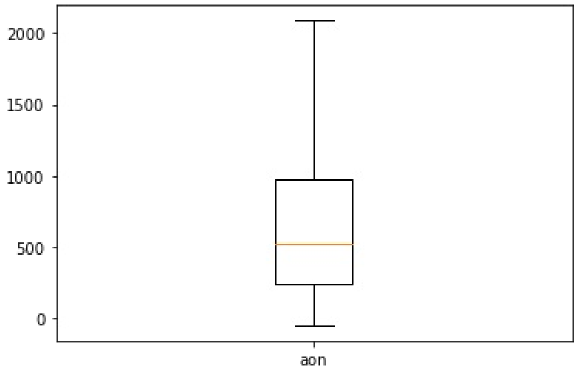
8 rows × 32 columns

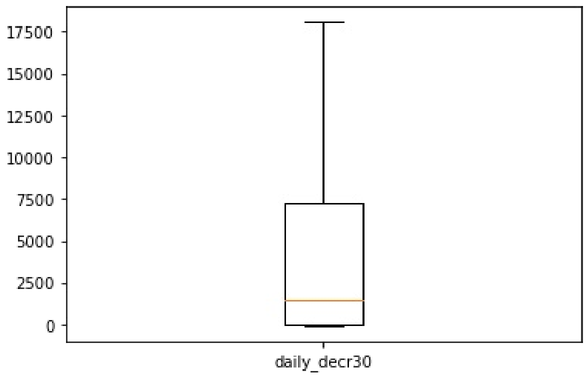
In [17]:
```
## after removal of outliers
```

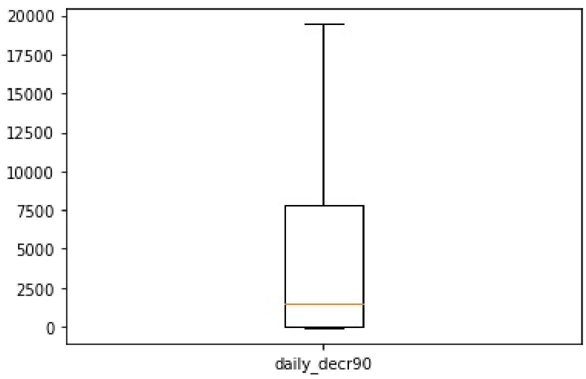
In [18]:
```
for i in cat_df:
    plt.boxplot(cat_df[i], labels = [i])
    plt.show()
```

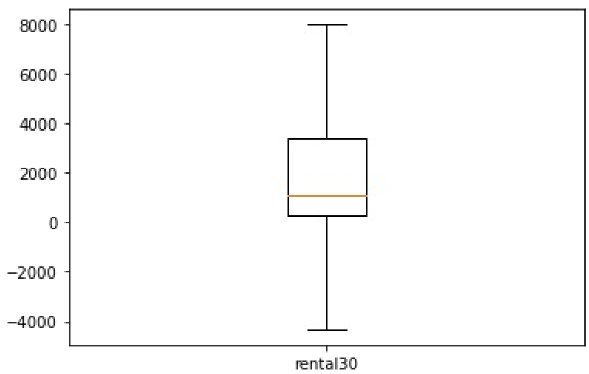


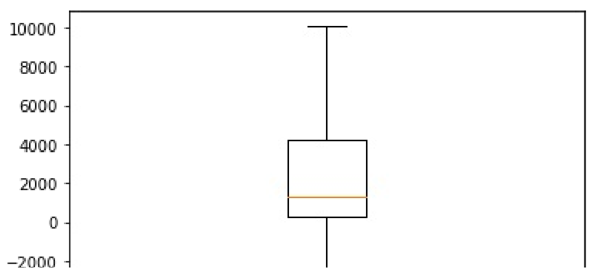

0.8
0.6
0.4
0.2
0.0
label

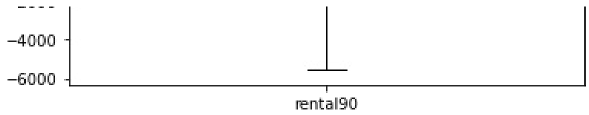

2000
1500
1000
500
0
aon

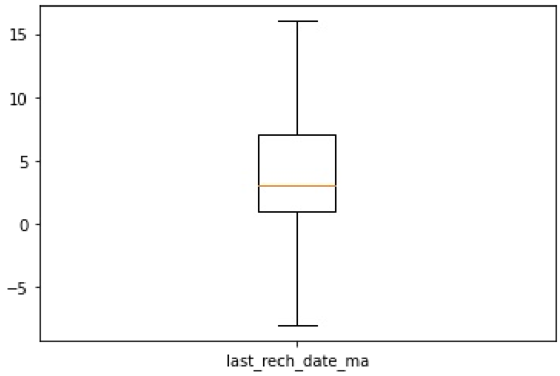

17500
15000
12500
10000
7500
5000
2500
0
daily_decr30

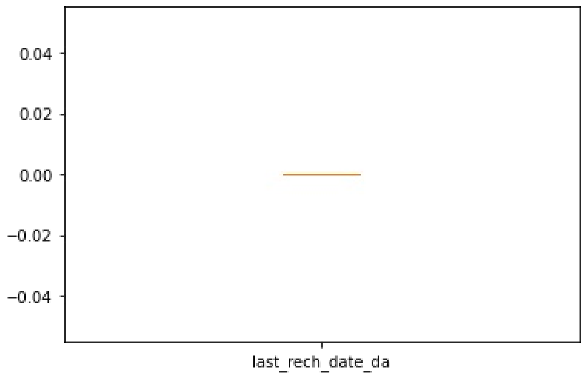

20000
17500
15000
12500
10000
7500
5000
2500
0
daily_decr90

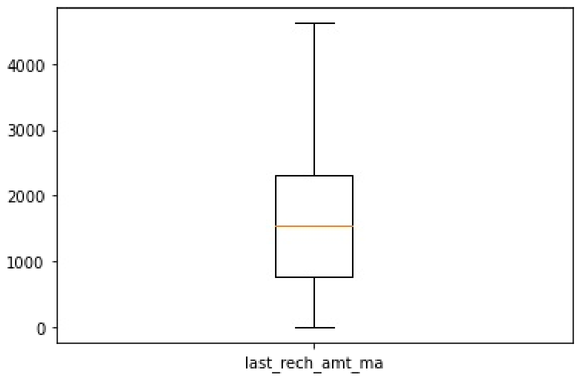

8000
6000
4000
2000
0
−2000
−4000
rental30

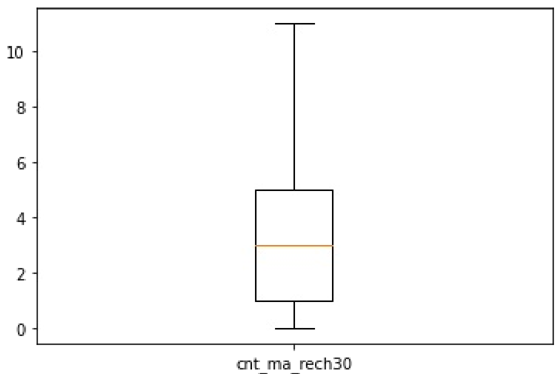

10000
8000
6000
4000
2000
0
−2000

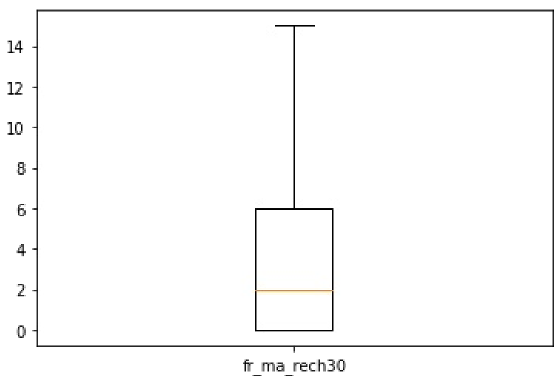

−4000
−6000
rental90

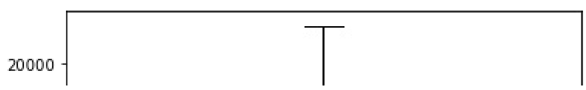

15
10
5
0
−5
last_rech_date_ma

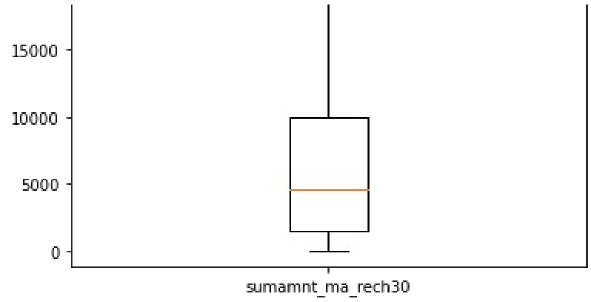

0.04
0.02
0.00
−0.02
−0.04
last_rech_date_da

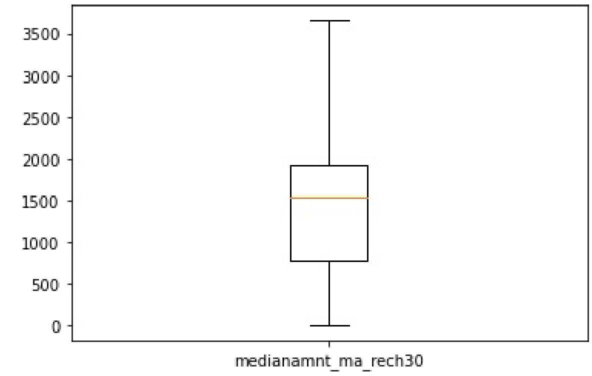

4000
3000
2000
1000
0
last_rech_amt_ma

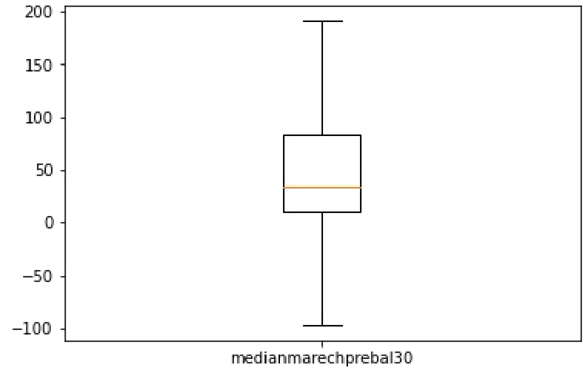

10
8
6
4
2
0
cnt_ma_rech30

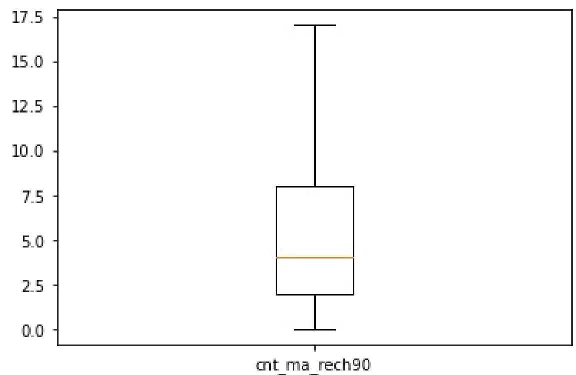

14
12
10
8
6
4
2
0
fr_ma_rech30

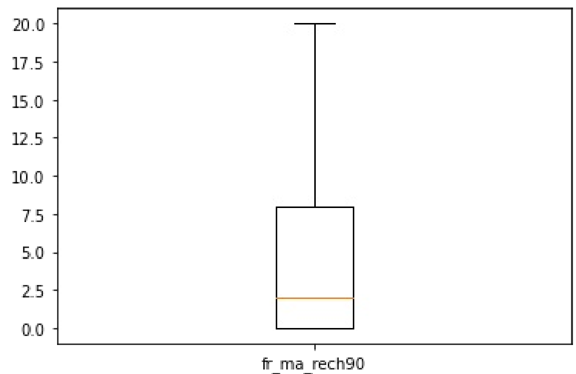

20000

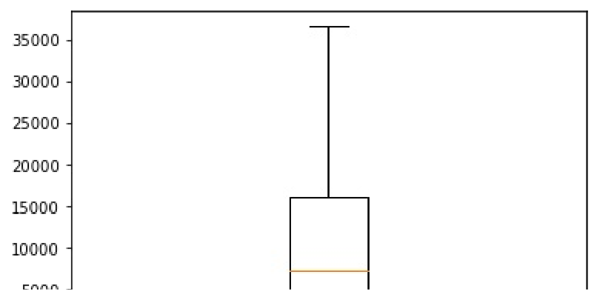

15000
10000
5000
0
sumamnt_ma_rech30

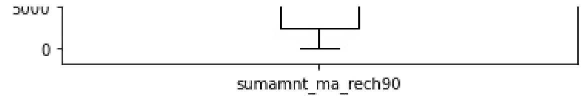

3500
3000
2500
2000
1500
1000
500
0
medianamnt_ma_rech30

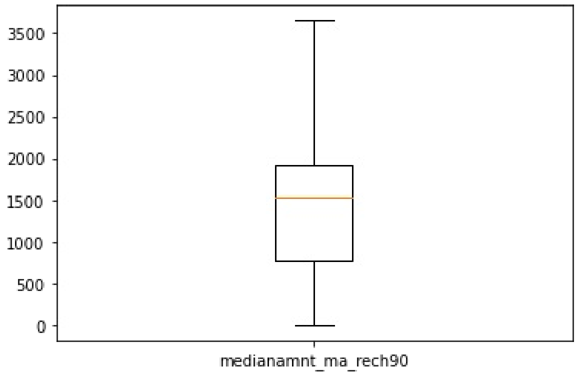

200
150
100
50
0
−50
−100
medianmarechprebal30

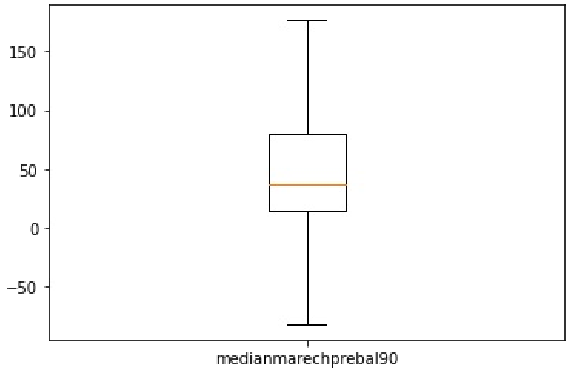

17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
cnt_ma_rech90

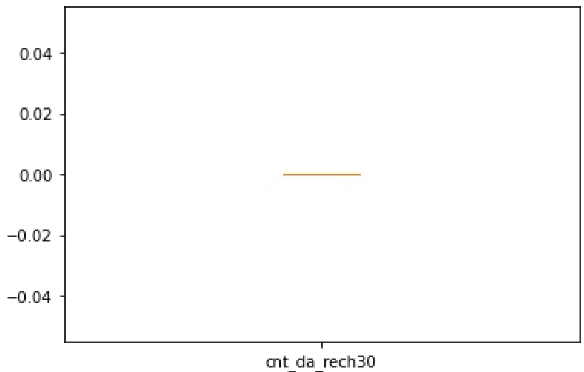

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
fr_ma_rech90

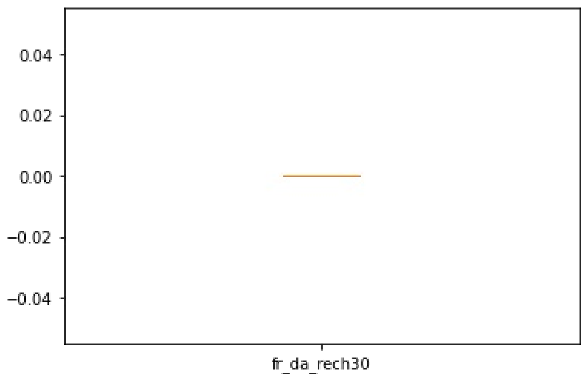

35000
30000
25000
20000
15000
10000

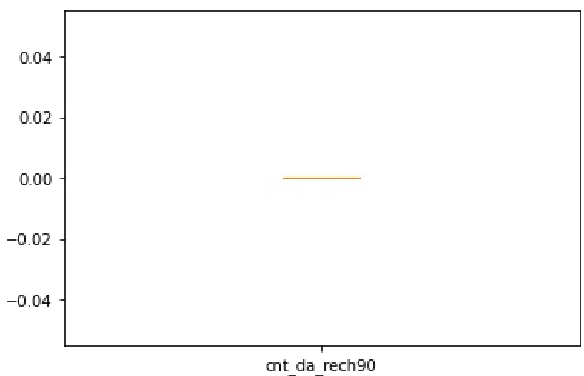

sumamnt_ma_rech90

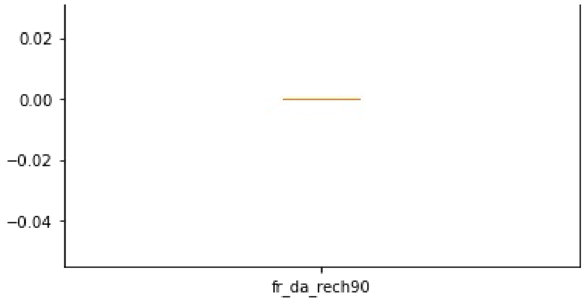

medianamnt_ma_rech90

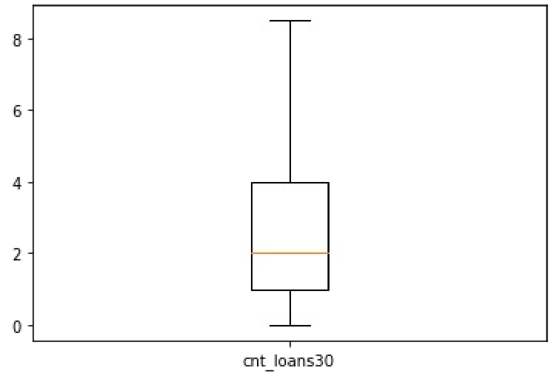

medianmarechprebal90

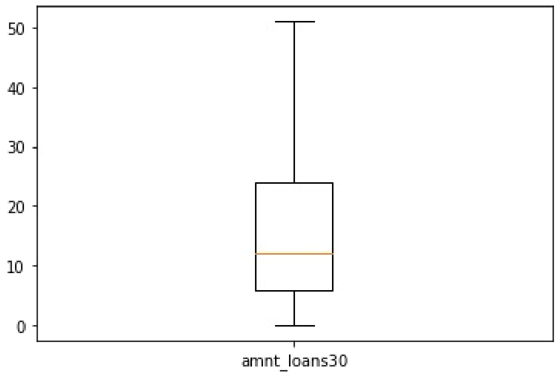

cnt_da_rech30

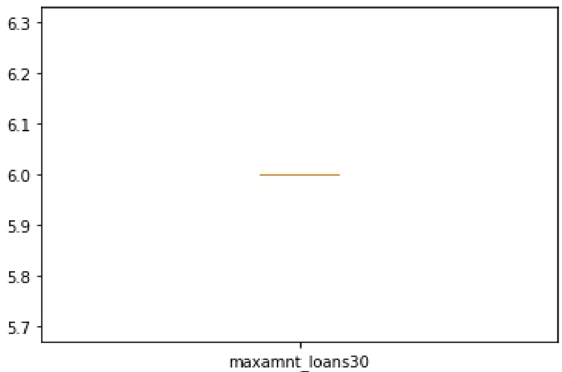

fr_da_rech30


cnt_da_rech90

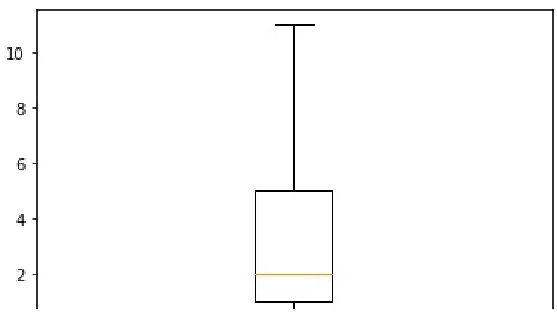

0.02
0.00
−0.02
−0.04
fr_da_rech90

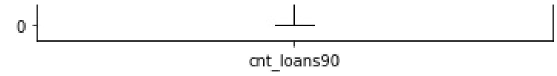

8
6
4
2
0
cnt_loans30

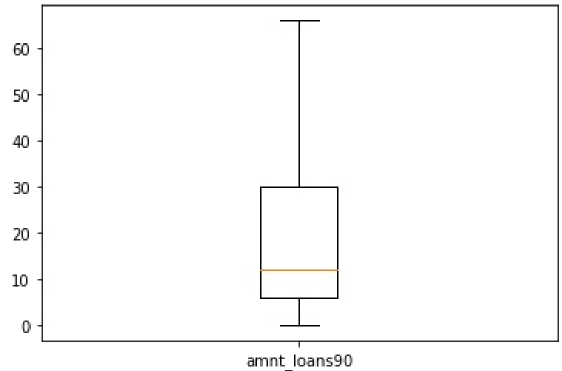

50
40
30
20
10
0
amnt_loans30


6.3
6.2
6.1
6.0
5.9
5.8
5.7
maxamnt_loans30

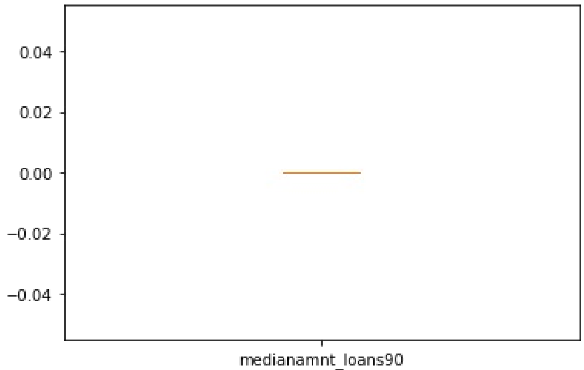

0.04
0.02
0.00
−0.02
−0.04
medianamnt_loans30

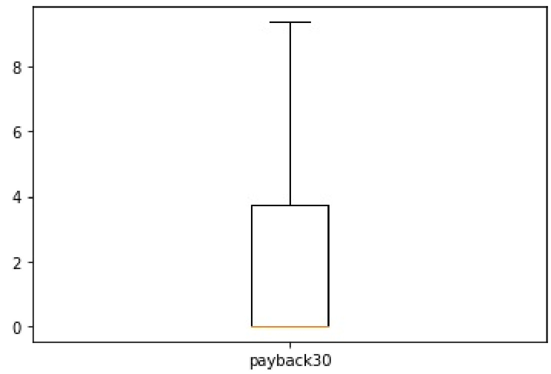

10
8
6
4
2

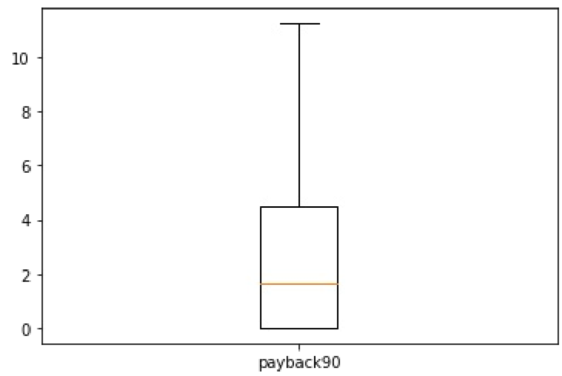

0
cnt_loans90


60
50
40
30
20
10
0
amnt_loans90


6.3
6.2
6.1
6.0
5.9
5.8
5.7
maxamnt_loans90


0.04
0.02
0.00
−0.02
−0.04
medianamnt_loans90


8
6
4
2
0
payback30


10
8
6
4
2
0
payback90

In [19]:

```python
for col in cat_df.columns:
    print(f"{col}: /n{cat_df[col].unique()}/n")
```

```
label: /n[0 1]/n
aon: /n[ 272.  712.  535. ... 2051. 2082. 2038.]/n
daily_decr30: /n[ 3055.05       12122.          1398.         ... 11843.11166667
 12488.22833333  4489.362     ]/n
daily_decr90: /n[ 3065.15       12124.75        1398.         ...   151.87233333
 12574.37        4534.82      ]/n
rental30: /n[ 220.13 3691.26  900.13 ... 5861.83  411.83  483.92]/n
rental90: /n[ 260.13 3691.26  900.13 ... 1728.36 8893.2   984.58]/n
last_rech_date_ma: /n[ 2. 16.  3.  4. 13.  1. 11.  8.  0.  6. 15.  7.  5. 10. 14.  9. -8. 12.
 -5. -3. -6. -4. -2. -1. -7.]/n
last_rech_date_da: /n[0.]/n
last_rech_amt_ma: /n[1539.  4617.5  947.  2309.  3178.   773.  1547.   770.     0.  4048.
  173.  1924.  2320.  2593.  1720.  3193.  1333.  4067.  3467.   777.
 1933.  1554.   790.  1580.  4340.  3466.   769.   946.  4066.  4047.
  954.   177.  1546.  1923.  1538.   772. ]/n
cnt_ma_rech30: /n[ 2  1  0  7  4  3  5 11  6  9 10  8]/n
fr_ma_rech30: /n[15.  0.  2. 10.  3.  1.  5.  6.  8. 12. 11.  4.  9. 13.  7. 14.]/n
sumamnt_ma_rech30: /n[ 3078.  5787.  1539. ... 14307. 13167. 12154.]/n
medianamnt_ma_rech30: /n[1539.  3655.     0.  2309.  3178.   773.  1156.   770.   771.5 1543.
  473.  1160.  1547.  1154.5 1924.  3178.5 2020.  2793.5 3613.  1243.
 2593.  1928.  2410.5 2320.   173.   947.  1158.5  856.  1720.  2551.
 3322.5  860.  1541.  1247.  1731.5 2120.   471.5 1353.  2743.5 1633.5
 2409.  1539.5 1347.  1158.  3622.5 2370.  3467.  1928.5 1974.  1821.
 2358.5 1176.5 2314.5 1933.   858.5 3228.  2986.  2014.5 3278.5 2070.
 1333.  2803.  3193.  2116.5  777.  2362.5  560.  2797.5 1246.5 3620.5
 2366.  1975.5 1929.5 1545.  1735.5 1628.  3330.  1250.5 2066.  3185.5
 1348.5 3280.   780.  2497.5  773.5 3188.  1933.5 1554.  1736.  2555.5
 1629.5 1053.  1580.  3136.5 1983.  1546.5 3466.  2807.  2843.5 1162.
 2507.  2545.  1822.  3624.5 1351.5 1740.  2751.  2756.5 2121.  2888.
 3239.5 1140.  2449.  3184.   769.   562.  1163.5 3292.  2456.5 2503.
 2884.   781.5 1245.  1550.5 2893.5 2062.5 3380.5 2856.5  946.5 3232.
 1435.5 2749.  1538.5 2418.5 2451.  1773.5 2110.5 1165.5 1333.5 1683.
 3618.5 2118.5 3560.  1440.   862.   775.  3367.  2420.  2885.5 1981.5
 2855.  2207.  2845.  2255.5 1241.  1559.5  946.  2992.  2126.5  475.
 2932.  3193.5  950.5 2410.  3320.5 2893.  2510.5 3177.5 2408.5 1436.
  954.  3295.5 1931.5 1675.5  190.  3630.  2990.5  790.  2690.5 3420.5
 3566.5 3376.5 2993.5 1681.5 1563.5  477.  3573.5 2558.5 1048.5 1254.
 1051.5 1637.  1770.  1826.5 2801.  1633.   863.5 3293.5 2695.5 3622.
 2556.5 2156.5 2943.5 2980.   177.  2793.  2802.5  772.5 1175.   772.
 2501.  2258.5 2939.5 3243.5 2985.5 2995.5]/n
medianmarechprebal30: /n[ 7.5  61.04 66.32 ... 86.74 48.24 71.37]/n
cnt_ma_rech90: /n[ 2  1  8  9  4  7  0  3 17 10  6 11 16 15  5 14 13 12]/n
fr_ma_rech90: /n[20  0  2  3  1  5 10  8 12 15  7 18  4  9 11  6 17 13 14 16 19]/n
sumamnt_ma_rech90: /n[ 3078.  5787.  1539. ... 35951. 36422. 17941.]/n
medianamnt_ma_rech90: /n[1539.  3650.5  947.  2888.  3178.   773.  1156.  1720.     0.   771.5
  770.  3178.5 1543.  1160.  2309.  1247.  1924.  2803.  3278.5  173.
 2020.  3193.  1243.  1547.  3613.  2358.5 1928.  1541.  2410.5 2320.
 1731.5  473.  1154.5 1158.5  856.  2593.  1333.  3467.   471.5 1353.
 3569.  1929.5 1633.5  858.5 2409.  2314.5 2551.  1546.5 1347.  2743.5
 2793.5 3622.5 2749.   860.  2893.  1539.5 2370.  1736.  2507.  1821.
  772.5 2797.5 1928.5 1683.  2451.  2014.5 1974.  3293.5 2362.5 1176.5
 1629.5 2751.  1933.  2556.5 1628.  1933.5 2066.  3228.  2503.  2986.
 3239.5  868.5 2070.  1246.5 3188.   560.   777.  2593.5 3280.  3620.5
 2366.  3184.  1545.  3030.  1975.5 1554.  2120.  1735.5 3330.   954.
 3292.  2116.5  780.  1158.  3080.5 1436.  1245.  2555.5 1981.5 2263.
 2756.5 1053.  2885.5 1348.5 3232.  1580.  3136.5 1983.  3193.  3618.5
 1333.5 3367.  3630.  1351.5 2456.5 2156.5 2807.  2843.5 1162.   790.
 1822.  1675.5 1740.  2062.5 2121.  3185.5 2893.5 1770.  1140.  2497.5
 2884.  1383.  2643.5 3322.5 1629.   769.  2695.5 1440.   177.  2980.
 1628.5 2420.  1051.5 2856.5 2845.   773.5 1163.5 3320.5  781.5 1550.5
 2449.   946.5  946.  2110.5  775.  2418.5 1048.5 2700.  1435.5 2939.5
 1241.   862.  2118.5 2993.5 2992.  3376.5 3560.  2793.   950.5 2563.
 2558.5 2255.5 2855.  2207.  3624.5 1633.  2400.  3324.5 1155.5 3380.5
 2990.5  859.5 1542.5  863.5  475.  2932.  2410.  3318.5 1826.5 2408.5
  753.  1168.5 3295.5 1637.  2126.5  481.5 2690.5 1681.5 2943.5 1559.5
 3466.  2122.   477.  3573.5 1985.  1254.  1526.5 1250.5  473.5 2514.
 3622.  1931.5 1546.   876.5 1927.5  175.  1548.5  963.5 1538.5 2501.
```

```
 2258.5 2073.5  772.  1923.   858.  3420.5 3243.5 2985.5 3288.5  490.
 3425.  2995.5 3432.  3282.  1165.5]/n
medianmarechprebal90: /n[  7.5   61.04  66.32 ...  49.82  27.66 118.97]/n
cnt_da_rech30: /n[0.]/n
fr_da_rech30: /n[0.]/n
cnt_da_rech90: /n[0]/n
fr_da_rech90: /n[0]/n
cnt_loans30: /n[2.  1.  7.  3.  4.  5.  8.  6.  8.5 0. ]/n
amnt_loans30: /n[12  6 42 18 24 30 48 51 36  0]/n
maxamnt_loans30: /n[6.]/n
medianamnt_loans30: /n[0.]/n
cnt_loans90: /n[ 2.  1.  7.  3.  4.  5.  8. 11.  6. 10.  9.  0.]/n
amnt_loans90: /n[12  6 42 18 24 30 48 66 36 60 54  0]/n
maxamnt_loans90: /n[6]/n
medianamnt_loans90: /n[0.]/n
payback30: /n[9.375      0.         2.33333333 6.         2.66666667 4.
 1.33333333 2.6        5.         1.8        1.375      2.
 7.5        3.66666667 1.57142857 2.72727273 3.25       9.
 6.75       2.16666667 4.25       4.8        2.5        1.5
 3.14285714 3.33333333 3.16666667 2.25       8.         5.33333333
 2.8        3.6        4.5        5.4        4.83333333 4.66666667
 2.375      3.         3.5        5.5        7.         5.66666667
 1.90909091 1.66666667 4.33333333 1.71428571 5.28571429 3.71428571
 6.25       1.         1.83333333 1.88888889 7.16666667 1.76923077
 1.75       3.2        4.6        7.25       1.25       6.66666667
 5.8        6.5        8.33333333 2.625      1.125      5.75
 2.14285714 6.33333333 9.33333333 3.4        4.4        1.16666667
 2.75       1.6        7.8        2.4        3.8        1.86956522
 1.2        4.16666667 2.2        5.2        7.66666667 8.5
 2.7        1.42857143 1.85714286 2.57142857 3.22222222 3.11764706
 4.71428571 7.55555556 5.25       3.625      3.85714286 3.75
 4.2        5.6        4.125      1.28571429 1.46666667 1.36842105
 9.25       2.44444444 8.66666667 2.83333333 6.16666667 2.61538462
 3.125      6.4        2.61111111 3.11111111 1.72727273 7.33333333
 2.27272727 2.85714286 6.6        4.75       1.875      1.55555556
 8.25       2.08333333 1.4        2.54545455 1.58823529 2.69230769
 6.28571429 1.86363636 1.63636364 1.53333333 2.88888889 3.83333333
 4.14285714 2.18181818 2.875      1.23809524 4.85714286 1.92307692
 1.625      1.78947368 1.47368421 2.42857143 2.13333333 4.42857143
 3.57142857 2.06666667 6.2        1.36363636 7.4        2.71428571
 4.625      2.22222222 3.42857143 6.83333333 5.42857143 5.83333333
 9.2        5.77777778 3.28571429 2.28571429 2.30769231 1.58333333
 3.375      3.44444444 5.375      1.77777778 7.75       2.9
 3.27272727 6.8        2.38461538 8.75       2.15384615 1.26666667
 4.57142857 1.45454545 4.28571429 3.55555556 2.3125     4.22222222
 2.81818182 1.22222222 8.14285714 2.55555556 2.53333333 1.38461538
 2.3        2.125      3.09090909 5.71428571 1.94736842 2.63636364
 2.45454545 4.875      1.9        3.3        2.90909091 1.7
 3.875      3.63636364 3.1        2.58333333 8.83333333 2.09090909
 2.11111111 1.81818182 1.52380952 1.53846154 4.11111111 2.36363636
 3.77777778 1.27272727 7.6        2.77777778 1.76470588 2.46666667
 2.05       4.18181818 5.44444444 8.875      5.85714286 1.46153846
 1.3        2.35294118 3.81818182 7.83333333 2.41666667 1.94117647
 2.1875     6.42857143 1.68181818 1.44444444 2.91666667 3.07142857
 2.07142857 1.8125     5.16666667 1.46428571 1.73333333 3.9
 1.93333333 3.23076923 2.26666667 1.54545455 8.4        2.76923077
 3.72727273 3.54545455 1.11111111 6.85714286 1.84615385 1.91666667
 2.1        3.08333333 5.57142857 1.41666667 3.7        2.64285714
 1.78571429 5.14285714 1.35714286 1.94444444 1.69230769 2.07692308
 2.92307692 6.875      5.625      1.07142857 2.78571429 2.6875
 1.31578947 9.28571429 8.2        2.46153846 4.55555556 8.8
 5.22222222 1.61538462 9.11111111 2.0625     7.2        8.6
 2.53846154 1.18181818 6.625      6.375      7.14285714 6.14285714
 1.30769231 2.36842105 4.44444444 1.86206897 3.58333333 1.14285714
 2.23076923 3.21428571 1.64285714 1.24       7.22222222 1.35
 1.43243243 1.15789474 4.27272727 2.41176471 5.11111111 1.4375
 1.52941176 8.42857143 1.47058824 1.95       2.23529412 1.1
 2.47058824 1.1875     7.42857143 1.62962963 5.69230769 1.88235294
 1.45       1.92857143 1.23076923 1.68421053 7.71428571 1.95238095
 1.07692308 4.9        3.88888889 1.80769231 1.76666667 4.05882353
 1.85       4.375      2.21428571 2.84615385 3.36363636 3.15384615
 2.04347826 7.11111111 1.76190476 6.3        1.72       3.90909091
 2.35714286 1.76       1.6875     1.73684211 1.9375     1.82352941
 2.92857143 7.125      8.28571429 6.27272727 3.07692308 1.59259259
 3.46666667 1.27777778 8.85714286 4.77777778 1.39285714 2.5625
 5.875      1.82608696 4.1        1.09090909 3.18181818 6.11111111
 1.45833333 3.73684211 5.90909091 1.3125     2.38888889 2.58823529
 8.625      1.23529412 1.5625     7.28571429 1.64705882 1.68
 5.36363636 3.41666667 6.55555556 1.70588235 2.17647059 6.57142857
 1.19230769 1.44827586 2.70588235 5.7        8.22222222 5.125
 1.35294118 1.21052632 1.61111111 1.32142857 1.29411765 1.41176471
 1.69565217 7.72727273 1.15625    2.73333333 1.7037037  7.77777778
 3.45454545 6.1875     4.07692308 2.05882353 2.4375     1.52631579
```

```
 2.31578947 1.56       6.63636364 4.72727273 1.18918919 2.29411765
 7.57142857 8.64       1.30434783 2.27777778 3.30769231 1.65
 2.04545455 5.1        1.28       1.40909091 8.57142857 3.91666667
 1.21428571 7.07692308 3.61538462 2.64705882 1.73076923 1.64
 1.11764706 2.05263158 2.19047619 2.8125     1.60869565 1.86666667
 8.71428571 5.3        1.525      1.38095238 8.16666667 3.52941176
 2.56521739 6.22222222 6.21428571 1.55172414 1.61904762 8.7
 1.21212121 1.51851852 3.13333333 4.54545455 4.09090909 1.84210526
 2.82352941 2.26315789 1.31818182 6.15384615 1.05555556 2.11764706
 1.95454545 6.77777778 6.71428571 1.08       2.15       1.80952381
 1.17647059 6.10526316 1.89473684 7.85714286 4.11764706 1.15
 4.36363636 1.72222222 9.22222222 2.47368421 1.34782609 7.27272727
 4.08333333 9.16666667 1.47619048 4.7        1.95652174 9.09090909
 8.1        3.29411765 1.59090909 1.79310345 1.91304348 1.08333333
 9.3        5.46153846 1.52173913 5.55555556 4.3        6.88888889
 1.65217391 1.77272727 2.13636364 9.08333333 2.86666667 1.29166667
 2.34482759 1.38888889 4.81818182 1.79166667 8.375      7.09090909
 2.52631579 2.35       7.44444444 8.88888889 5.15384615 8.06666667
 1.32       1.48275862 1.20833333 6.0625     2.45       1.67857143
 3.1875     6.44444444 2.21052632 6.38461538 1.54166667 2.23809524
 1.63157895 3.76923077 1.04166667 4.15384615 1.96551724 1.57692308
 7.875      5.58823529 1.42105263 3.84615385 6.125      1.88
 1.70833333 6.90909091 2.93333333 7.9        5.27272727 4.38461538
 8.53846154 3.38461538 1.47826087 1.5483871  8.44444444 2.94736842
 3.57894737 1.95833333 1.35135135 1.82142857 4.84615385 1.2173913
 1.43478261 2.94117647 9.125      5.78571429 1.7826087  7.375
 4.63636364 1.06666667 4.45454545 1.26315789 4.88888889 3.35714286
 4.07142857 3.46153846 8.78947368 2.05555556 1.65384615 2.10526316
 5.63636364 7.625      1.22727273 2.29032258 1.57894737 5.61538462]/n
payback90: /n[11.25        0.          2.33333333 ...  1.97297297  1.53571429
  5.23529412]/n
```

In [20]:

```python
cat=df.select_dtypes(include=['object'])
cat
```

Out[20]:

	pcircle	pdate
0	UPW	2016-07-20
1	UPW	2016-08-10
2	UPW	2016-08-19
3	UPW	2016-06-06
4	UPW	2016-06-22
...	...	...
209588	UPW	2016-06-17
209589	UPW	2016-06-12
209590	UPW	2016-07-29
209591	UPW	2016-07-25
209592	UPW	2016-07-07

209593 rows × 2 columns

In [21]:

```python
for col in cat.columns:
    print(f"{col}: /n{cat[col].unique()}/n")
```

```
pcircle: /n['UPW']/n
pdate: /n['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']/n
```

In [22]:
```
cat.shape
```

Out[22]: (209593, 2)

In [23]:
```
cat=pd.get_dummies(cat,drop_first=True)
cat
```

Out[23]:

	pdate_2016-06-02	pdate_2016-06-03	pdate_2016-06-04	pdate_2016-06-05	pdate_2016-06-06	pdate_2016-06-07	pdate_2016-06-08	pdate_2016-06-09	pdate_2016-06-10	pdate_2016-06-11	...
0	0	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	0	...
3	0	0	0	0	1	0	0	0	0	0	...
4	0	0	0	0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
209588	0	0	0	0	0	0	0	0	0	0	...
209589	0	0	0	0	0	0	0	0	0	0	...
209590	0	0	0	0	0	0	0	0	0	0	...
209591	0	0	0	0	0	0	0	0	0	0	...
209592	0	0	0	0	0	0	0	0	0	0	...

209593 rows × 81 columns

In [24]:
```
## concatenating the dataframe
```

In [25]:
```
x1=pd.concat([cat_df,cat],axis=1)
x1.head()
```

Out[25]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_rech30	...	pda
0	0	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539.0	2	...	
1	1	712.0	12122.000000	12124.750000	3691.26	3691.26	16.0	0.0	4617.5	1	...	
2	1	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539.0	1	...	
3	1	241.0	21.228000	21.228000	159.42	159.42	16.0	0.0	947.0	0	...	
4	1	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309.0	7	...	

5 rows × 114 columns

In [26]:
```
## shape of the x
```

In [27]:
```
x1.shape
```

Out[27]: (209593, 114)

In [28]:
```
x=x1.drop(['label'],axis=1)
x.head()
```

Out[28]:

	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
0	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539.0	2	15.0
1	712.0	12122.000000	12124.750000	3691.26	3691.26	16.0	0.0	4617.5	1	0.0
2	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539.0	1	0.0
3	241.0	21.228000	21.228000	159.42	159.42	16.0	0.0	947.0	0	0.0
4	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309.0	7	2.0

5 rows × 113 columns

In [29]:
```
y=x1['label']
y.head()
```

Out[29]:
```
0    0
1    1
2    1
3    1
4    1
Name: label, dtype: int64
```

In [30]:
```
y.shape
```

Out[30]:
```
(209593,)
```

In [ ]:

In [31]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [32]:
```
import imblearn
```

In [33]:
```
print(imblearn.__version__)
```

```
0.9.1
```

In [34]:
```
## under sampling
```

In [35]:
```
from imblearn.under_sampling import NearMiss

under=NearMiss(version=1,n_neighbors=3)

x_smote,y_smote=under.fit_resample(x_train,y_train)
```

In [36]:
```
##x_smote.value_counts()
```

In [37]:
```
y_smote.value_counts()
```

Out[37]:
```
0    17470
1    17470
Name: label, dtype: int64
```

In [38]:
```
y_train.value_counts()
```

Out[38]:
```
1    122957
0     17470
Name: label, dtype: int64
```

In [39]:
```
x_smote.shape
```

Out[39]:
```
(34940, 113)
```

In [40]:
```
y_smote.shape
```

Out[40]: (34940,)

In [41]:
```
from sklearn.model_selection import train_test_split
x_train1,x_test1,y_train1,y_test1=train_test_split(x_smote,y_smote,test_size=0.33,random_state=42)
```

In [42]:
```
x_train1.shape
```

Out[42]: (23409, 113)

In [43]:
```
y_test1.shape
```

Out[43]: (11531,)

In [44]:
```
x_test1.shape
```

Out[44]: (11531, 113)

In [45]:
```
y_train1.shape
```

Out[45]: (23409,)

In [46]:
```
### standardization of dataset
```

In [47]:
```
from sklearn.preprocessing import MinMaxScaler
min=MinMaxScaler()
z=min.fit_transform(x_smote)
z
```

Out[47]:
```
array([[0.48453608, 0.00214257, 0.001989  , ..., 0.        , 0.        ,
        0.        ],
       [0.30506092, 0.45458775, 0.42562259, ..., 0.        , 0.        ,
        0.        ],
       [0.50749766, 0.00232132, 0.00215494, ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.24086223, 0.05538201, 0.05141241, ..., 0.        , 0.        ,
        0.        ],
       [0.44095595, 0.08683916, 0.08061482, ..., 0.        , 0.        ,
        1.        ],
       [0.1710403 , 0.00311221, 0.00288914, ..., 0.        , 0.        ,
        0.        ]])
```

In [48]:
```
## modelling phase and training starts
```

In [49]:
```
from sklearn.model_selection import train_test_split,cross_val_score

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

In [50]:
```
## RandomForestClassifier
```

In [51]:
```
model=RandomForestClassifier()
model.fit(x_train1,y_train1)
p=model.predict(x_test1)
s=cross_val_score(model,x_smote,y_smote,cv=10)
```

```
print('Accuracy',np.round(accuracy_score(p,y_test1),4))
print('----------------------------------------------------------')
print('Mean of Cross Validation Score',np.round(s.mean(),4))
print('----------------------------------------------------------')
print('Confusion Matrix')
print(confusion_matrix(p,y_test1))
print('----------------------------------------------------------')
print('Classification Report')
print(classification_report(p,y_test1))
```

```
Accuracy 0.869
----------------------------------------------------------
Mean of Cross Validation Score 0.8453
----------------------------------------------------------
Confusion Matrix
[[4916  655]
 [ 856 5104]]
----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.88      0.87      5571
           1       0.89      0.86      0.87      5960

    accuracy                           0.87     11531
   macro avg       0.87      0.87      0.87     11531
weighted avg       0.87      0.87      0.87     11531
```

In [53]:

```
## decisiontreeClassifier
```

In [54]:

```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()
```

In [55]:

```
model.fit(x_train1,y_train1)
p=model.predict(x_test1)
s=cross_val_score(model,x_smote,y_smote,cv=10)
```

In [56]:

```
print('Accuracy',np.round(accuracy_score(p,y_test1),4))
print('----------------------------------------------------------')
print('Mean of Cross Validation Score',np.round(s.mean(),4))
print('----------------------------------------------------------')
print('Confusion Matrix')
print(confusion_matrix(p,y_test1))
print('----------------------------------------------------------')
print('Classification Report')
print(classification_report(p,y_test1))
```

```
Accuracy 0.8186
----------------------------------------------------------
Mean of Cross Validation Score 0.7966
----------------------------------------------------------
Confusion Matrix
[[4715 1035]
 [1057 4724]]
----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       0.82      0.82      0.82      5750
           1       0.82      0.82      0.82      5781

    accuracy                           0.82     11531
   macro avg       0.82      0.82      0.82     11531
weighted avg       0.82      0.82      0.82     11531
```

In [57]:

```
## kneighborsClassifier
```

In [58]:

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier()
```

In [59]:

```
model.fit(x_train1,y_train1)
```

```
p=model.predict(x_test1)
s=cross_val_score(model,x_smote,y_smote,cv=10)
```

In [60]:

```
print('Accuracy',np.round(accuracy_score(p,y_test1),4))
print('-----------------------------------------------------------')
print('Mean of Cross Validation Score',np.round(s.mean(),4))
print('-----------------------------------------------------------')
print('Confusion Matrix')
print(confusion_matrix(p,y_test1))
print('-----------------------------------------------------------')
print('Classification Report')
print(classification_report(p,y_test1))
```

```
Accuracy 0.7609
----------------------------------------------------------
Mean of Cross Validation Score 0.6955
----------------------------------------------------------
Confusion Matrix
[[4137 1122]
 [1635 4637]]
----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       0.72      0.79      0.75      5259
           1       0.81      0.74      0.77      6272

    accuracy                           0.76     11531
   macro avg       0.76      0.76      0.76     11531
weighted avg       0.76      0.76      0.76     11531
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [61]:

```
## gradientBoosting classifier
```

In [62]:

```
from sklearn.ensemble import GradientBoostingClassifier
model=GradientBoostingClassifier()
```

In [63]:

```
## training
```

In [64]:

```
model.fit(x_train1,y_train1)
p=model.predict(x_test1)
s=cross_val_score(model,x_smote,y_smote,cv=10)
```

In [65]:

```
print('Accuracy',np.round(accuracy_score(p,y_test1),4))
print('-----------------------------------------------------------')
print('Mean of Cross Validation Score',np.round(s.mean(),4))
print('-----------------------------------------------------------')
print('Confusion Matrix')
print(confusion_matrix(p,y_test1))
print('-----------------------------------------------------------')
print('Classification Report')
print(classification_report(p,y_test1))
```

```
Accuracy 0.8459
----------------------------------------------------------
Mean of Cross Validation Score 0.842
----------------------------------------------------------
Confusion Matrix
[[4667  672]
 [1105 5087]]
```

```
----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       0.81      0.87      0.84      5339
           1       0.88      0.82      0.85      6192

    accuracy                           0.85     11531
   macro avg       0.85      0.85      0.85     11531
weighted avg       0.85      0.85      0.85     11531
```

In [66]:
```
## ada boost classifier
```

In [67]:
```
from sklearn.ensemble import AdaBoostClassifier
model=AdaBoostClassifier()
```

In [68]:
```
model.fit(x_train1,y_train1)
p=model.predict(x_test1)
s=cross_val_score(model,x_smote,y_smote,cv=10)
```

In [69]:
```
print('Accuracy',np.round(accuracy_score(p,y_test1),4))
print('----------------------------------------------------------')
print('Mean of Cross Validation Score',np.round(s.mean(),4))
print('----------------------------------------------------------')
print('Confusion Matrix')
print(confusion_matrix(p,y_test1))
print('----------------------------------------------------------')
print('Classification Report')
print(classification_report(p,y_test1))
```

```
Accuracy 0.8419
----------------------------------------------------------
Mean of Cross Validation Score 0.8356
----------------------------------------------------------
Confusion Matrix
[[4753  804]
 [1019 4955]]
----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       0.82      0.86      0.84      5557
           1       0.86      0.83      0.84      5974

    accuracy                           0.84     11531
   macro avg       0.84      0.84      0.84     11531
weighted avg       0.84      0.84      0.84     11531
```

In [70]:
```
## hyper paramter tuning for random forest classifier
```

In [71]:
```
params={'n_estimators':[100, 300, 500, 700],
        'min_samples_split':[1,2,3,4],
        'min_samples_leaf':[1,2,3,4],
        'max_depth':[None,1,2,3,4,5,6,7,8,9,10,15,20,25,30,35,40]}
```

In [72]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [73]:
```
g=RandomizedSearchCV(RandomForestClassifier(),params,cv=10)
```

In [74]:
```
g.fit(x_train1,y_train1)
```

Out[74]:

▸ **RandomizedSearchCV**

▸ **estimator: RandomForestClassifier**

▸ RandomForestClassifier

In [75]:
```
print(g.best_estimator_)
print(g.best_params_)
print(g.best_score_)
```

```
RandomForestClassifier(max_depth=40, min_samples_leaf=3, min_samples_split=3)
{'n_estimators': 100, 'min_samples_split': 3, 'min_samples_leaf': 3, 'max_depth': 40}
0.8639419015177238
```

In [76]:
```
m=RandomForestClassifier(max_depth=40, min_samples_leaf=3, min_samples_split=3,n_estimators=100)
m.fit(x_train1,y_train1)
p=m.predict(x_test1)
score=cross_val_score(m,x_smote,y_smote,cv=10)
```

In [77]:
```
print('Accuracy',np.round(accuracy_score(p,y_test1),4))
print('-----------------------------------------------------------')
print('Mean of Cross Validation Score',np.round(s.mean(),4))
print('-----------------------------------------------------------')
print('Confusion Matrix')
print(confusion_matrix(p,y_test1))
print('-----------------------------------------------------------')
print('Classification Report')
print(classification_report(p,y_test1))
```

```
Accuracy 0.8641
-----------------------------------------------------------
Mean of Cross Validation Score 0.8356
-----------------------------------------------------------
Confusion Matrix
[[4889  684]
 [ 883 5075]]
-----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.88      0.86      5573
           1       0.88      0.85      0.87      5958

    accuracy                           0.86     11531
   macro avg       0.86      0.86      0.86     11531
weighted avg       0.86      0.86      0.86     11531
```

In [ ]:
```
## testing the model
```

In [82]:
```
import numpy as np
```

In [90]:
```
a=np.array(y_test1)
a
```

Out[90]:
```
array([0, 1, 0, ..., 0, 1, 0], dtype=int64)
```

In [91]:
```
pred=np.array(m.predict(x_test1))
pred
```

Out[91]:
```
array([0, 1, 0, ..., 0, 1, 0], dtype=int64)
```

In [92]:
```
df_com=pd.DataFrame({'predicted':pred,'actual':a},index=range(len(a)))
```

In [89]:
```
df_com
```

Out[89]:

	predicted	actual
0	0	0
1	1	1

2	0	0
3	0	0
4	0	0
...	...	...
11526	0	0
11527	0	0
11528	0	0
11529	1	1
11530	0	0

11531 rows × 2 columns

In [93]:

```
## saving the model
```

In [94]:

```
import pickle
```

In [96]:

```
filename='micro_credit_defaulter.pkl'
```

In [97]:

```
pickle.dump(m,open(filename,'wb'))
```

In [ ]:

```

```

Loading [MathJax]/extensions/Safe.js