Final report

May 31, 2019

0.0.1 Making DATA SET

0.0.2 &

```
In [157]: import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          import os
          import random
          import matplotlib as mpl
          import matplotlib.pyplot as plt
          mpl.rcParams['axes.unicode_minus'] = False

          %matplotlib inline
          import seaborn as sns
          from sklearn import preprocessing
          from sklearn.utils.class_weight import compute_sample_weight
          random.seed(1995)
          ticdata2000=pd.read_csv("ticdata2000.txt",engine='python',header=None,sep="\s+")
          ticdata2000.head()
          ticdata2000.shape
          ticeval2000=pd.read_csv("ticeval2000.txt",engine='python',header=None,sep="\s+")
          ticeval2000.head()
          ticeval2000.shape
          tictgts2000=pd.read_csv("tictgts2000.txt",engine='python',header=None,sep="\s+")
          tictgts2000.head()
          tictgts2000.shape
          ticdata=pd.concat([ticeval2000,tictgts2000],axis=1)
          ticdata.shape
          ticdata.head()
          ticdata.columns=ticdata2000.columns
          ticdata=pd.concat([ticdata,ticdata2000],axis=0,ignore_index=True) ###ticeval2000 tic
          names="MOSTYPE,MAANTHUI,MGEMOMV,MGEMLEEF,MOSHOOFD,MGODRK,MGODPR,MGODOV,MGODGE,MRELGE
          names=names.split(",")
          names=[x.strip() for x in names]
          ticdata.columns=names
          rawdata=ticdata.copy()
```

0.0.3

```
In [158]: ticdata.shape

Out[158]: (9822, 86)

In [159]: ticdata.head() ###

Out[159]:    MOSTYPE  MAANTHUI  MGEMOMV  MGEMLEEF  MOSHOOFD  MGODRK  MGODPR  MGODOV  \
          0       33         1        4         2         8       0       6       0
          1        6         1        3         2         2       0       5       0
          2       39         1        3         3         9       1       4       2
          3        9         1        2         3         3       2       3       2
          4       31         1        2         4         7       0       2       0

             MGODGE  MRELGE  ...  APERSONG  AGEZONG  AWAOREG  ABRAND  AZEILPL  APLEZIER  \
          0       3       5  ...         0        0        0       1        0         0
          1       4       5  ...         0        0        0       1        0         0
          2       3       5  ...         0        0        0       1        0         0
          3       4       5  ...         0        0        0       1        0         0
          4       7       9  ...         0        0        0       1        0         0

             AFIETS  AINBOED  ABYSTAND  CARAVAN
          0       0        0         0        0
          1       0        0         0        1
          2       0        0         0        0
          3       0        0         0        0
          4       0        0         0        0

          [5 rows x 86 columns]

In [160]: ticdata.tail() ###

Out[160]:       MOSTYPE  MAANTHUI  MGEMOMV  MGEMLEEF  MOSHOOFD  MGODRK  MGODPR  MGODOV  \
          9817       36         1        1         2         8       0       6       1
          9818       35         1        4         4         8       1       4       1
          9819       33         1        3         4         8       0       6       0
          9820       34         1        3         2         8       0       7       0
          9821       33         1        3         3         8       0       6       1

                MGODGE  MRELGE  ...  APERSONG  AGEZONG  AWAOREG  ABRAND  AZEILPL  \
          9817       2       1  ...         0        0        0       1        0
          9818       4       6  ...         0        0        0       1        0
          9819       3       5  ...         0        0        0       1        0
          9820       2       7  ...         0        0        0       0        0
          9821       2       7  ...         0        0        0       0        0

                APLEZIER  AFIETS  AINBOED  ABYSTAND  CARAVAN
          9817         0       0        0         0        0
```

```
9818        0       0        0        0        0
9819        0       0        0        0        1
9820        0       0        0        0        0
9821        0       0        0        0        0

[5 rows x 86 columns]
```

```
In [161]: ticdata.columns ###
```

```
Out[161]: Index(['MOSTYPE', 'MAANTHUI', 'MGEMOMV', 'MGEMLEEF', 'MOSHOOFD', 'MGODRK',
                 'MGODPR', 'MGODOV', 'MGODGE', 'MRELGE', 'MRELSA', 'MRELOV', 'MFALLEEN',
                 'MFGEKIND', 'MFWEKIND', 'MOPLHOOG', 'MOPLMIDD', 'MOPLLAAG', 'MBERHOOG',
                 'MBERZELF', 'MBERBOER', 'MBERMIDD', 'MBERARBG', 'MBERARBO', 'MSKA',
                 'MSKB1', 'MSKB2', 'MSKC', 'MSKD', 'MHHUUR', 'MHKOOP', 'MAUT1', 'MAUT2',
                 'MAUT0', 'MZFONDS', 'MZPART', 'MINKM30', 'MINK3045', 'MINK4575',
                 'MINK7512', 'MINK123M', 'MINKGEM', 'MKOOPKLA', 'PWAPART', 'PWABEDR',
                 'PWALAND', 'PPERSAUT', 'PBESAUT', 'PMOTSCO', 'PVRAAUT', 'PAANHANG',
                 'PTRACTOR', 'PWERKT', 'PBROM', 'PLEVEN', 'PPERSONG', 'PGEZONG',
                 'PWAOREG', 'PBRAND', 'PZEILPL', 'PPLEZIER', 'PFIETS', 'PINBOED',
                 'PBYSTAND', 'AWAPART', 'AWABEDR', 'AWALAND', 'APERSAUT', 'ABESAUT',
                 'AMOTSCO', 'AVRAAUT', 'AAANHANG', 'ATRACTOR', 'AWERKT', 'ABROM',
                 'ALEVEN', 'APERSONG', 'AGEZONG', 'AWAOREG', 'ABRAND', 'AZEILPL',
                 'APLEZIER', 'AFIETS', 'AINBOED', 'ABYSTAND', 'CARAVAN'],
                dtype='object')
```

0.0.4

```
In [162]: ticdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9822 entries, 0 to 9821
Data columns (total 86 columns):
MOSTYPE     9822 non-null int64
MAANTHUI    9822 non-null int64
MGEMOMV     9822 non-null int64
MGEMLEEF    9822 non-null int64
MOSHOOFD    9822 non-null int64
MGODRK      9822 non-null int64
MGODPR      9822 non-null int64
MGODOV      9822 non-null int64
MGODGE      9822 non-null int64
MRELGE      9822 non-null int64
MRELSA      9822 non-null int64
MRELOV      9822 non-null int64
MFALLEEN    9822 non-null int64
MFGEKIND    9822 non-null int64
MFWEKIND    9822 non-null int64
MOPLHOOG    9822 non-null int64
MOPLMIDD    9822 non-null int64
```

```
MOPLLAAG    9822 non-null int64
MBERHOOG    9822 non-null int64
MBERZELF    9822 non-null int64
MBERBOER    9822 non-null int64
MBERMIDD    9822 non-null int64
MBERARBG    9822 non-null int64
MBERARBO    9822 non-null int64
MSKA        9822 non-null int64
MSKB1       9822 non-null int64
MSKB2       9822 non-null int64
MSKC        9822 non-null int64
MSKD        9822 non-null int64
MHHUUR      9822 non-null int64
MHKOOP      9822 non-null int64
MAUT1       9822 non-null int64
MAUT2       9822 non-null int64
MAUT0       9822 non-null int64
MZFONDS     9822 non-null int64
MZPART      9822 non-null int64
MINKM30     9822 non-null int64
MINK3045    9822 non-null int64
MINK4575    9822 non-null int64
MINK7512    9822 non-null int64
MINK123M    9822 non-null int64
MINKGEM     9822 non-null int64
MKOOPKLA    9822 non-null int64
PWAPART     9822 non-null int64
PWABEDR     9822 non-null int64
PWALAND     9822 non-null int64
PPERSAUT    9822 non-null int64
PBESAUT     9822 non-null int64
PMOTSCO     9822 non-null int64
PVRAAUT     9822 non-null int64
PAANHANG    9822 non-null int64
PTRACTOR    9822 non-null int64
PWERKT      9822 non-null int64
PBROM       9822 non-null int64
PLEVEN      9822 non-null int64
PPERSONG    9822 non-null int64
PGEZONG     9822 non-null int64
PWAOREG     9822 non-null int64
PBRAND      9822 non-null int64
PZEILPL     9822 non-null int64
PPLEZIER    9822 non-null int64
PFIETS      9822 non-null int64
PINBOED     9822 non-null int64
PBYSTAND    9822 non-null int64
AWAPART     9822 non-null int64
```

```
AWABEDR     9822 non-null int64
AWALAND     9822 non-null int64
APERSAUT    9822 non-null int64
ABESAUT     9822 non-null int64
AMOTSCO     9822 non-null int64
AVRAAUT     9822 non-null int64
AAANHANG    9822 non-null int64
ATRACTOR    9822 non-null int64
AWERKT      9822 non-null int64
ABROM       9822 non-null int64
ALEVEN      9822 non-null int64
APERSONG    9822 non-null int64
AGEZONG     9822 non-null int64
AWAOREG     9822 non-null int64
ABRAND      9822 non-null int64
AZEILPL     9822 non-null int64
APLEZIER    9822 non-null int64
AFIETS      9822 non-null int64
AINBOED     9822 non-null int64
ABYSTAND    9822 non-null int64
CARAVAN     9822 non-null int64
dtypes: int64(86)
memory usage: 6.4 MB
```

```
In [163]: print('Missing values: %i' % ticdata.isnull().sum().sum()) ####    .
```

```
Missing values: 0
```

0.0.5

```
In [164]: plt.style.use('ggplot') ###ggplot style
```

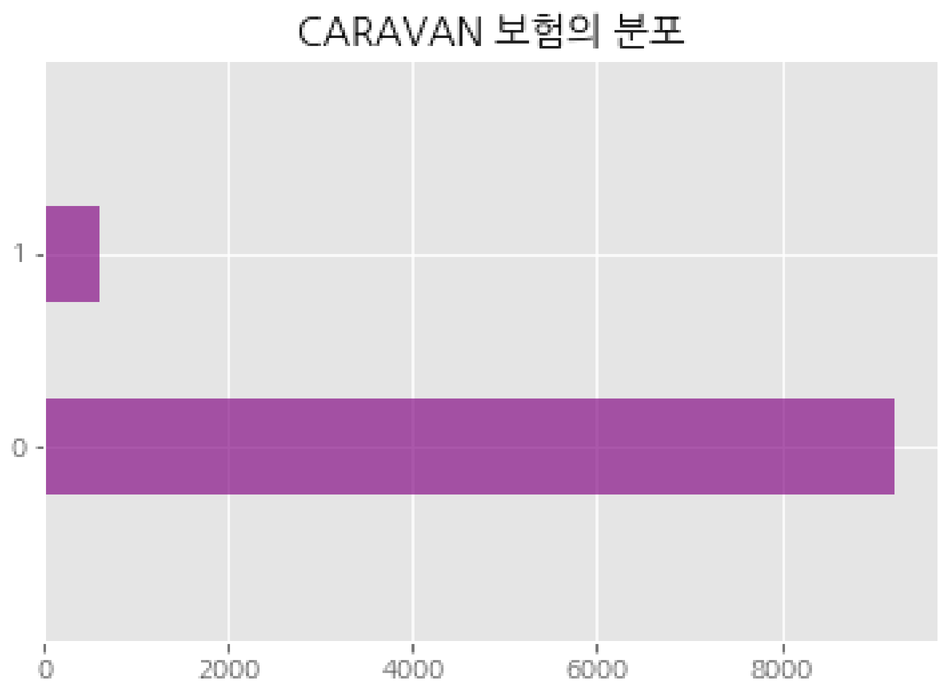
```
In [165]: import matplotlib.font_manager as fm
          print ('   : ', mpl.matplotlib_fname())###matplotlib
```

```
  :  C:\Users\jang\Anaconda3\lib\site-packages\matplotlib\mpl-data\matplotlibrc
```

```
In [166]: plt.rc('font', family='NanumGothic')###
```

```
In [167]: fig,ax= plt.subplots()
          ticdata.CARAVAN.value_counts().plot(kind='barh', color="purple", alpha=.65)
          ax.set_ylim(-1, len(ticdata.CARAVAN.value_counts()))
          plt.title("CARAVAN  ")
          print("CARAVAN  ")
          print(ticdata.CARAVAN.value_counts())
```

```
CARAVAN
0    9236
1     586
Name: CARAVAN, dtype: int64
```

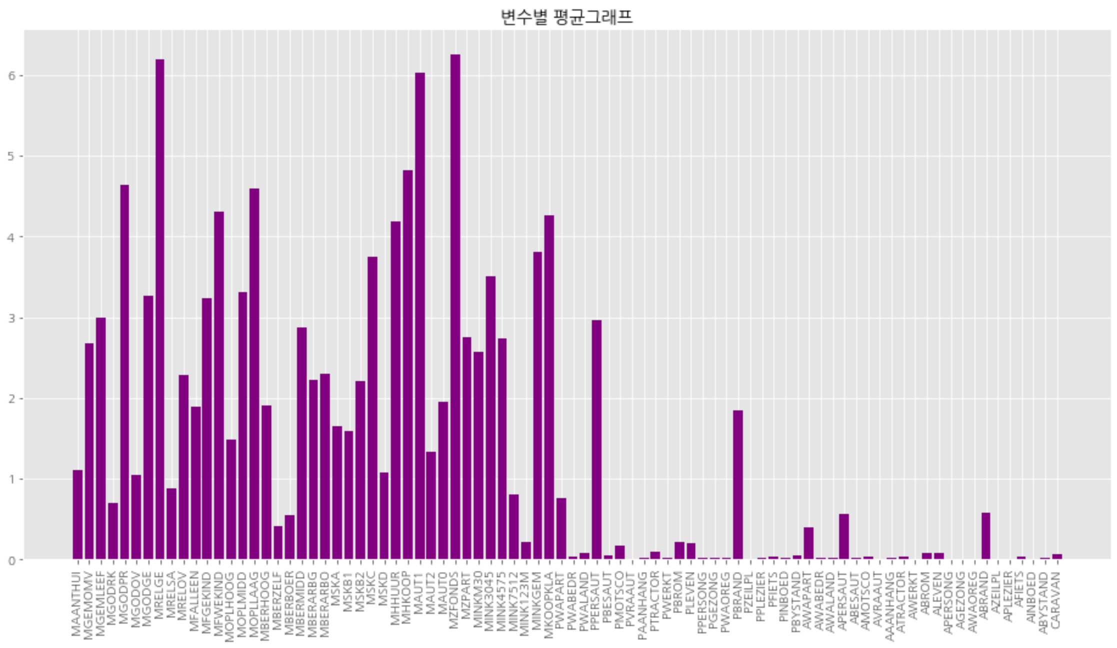


0.0.6 & &

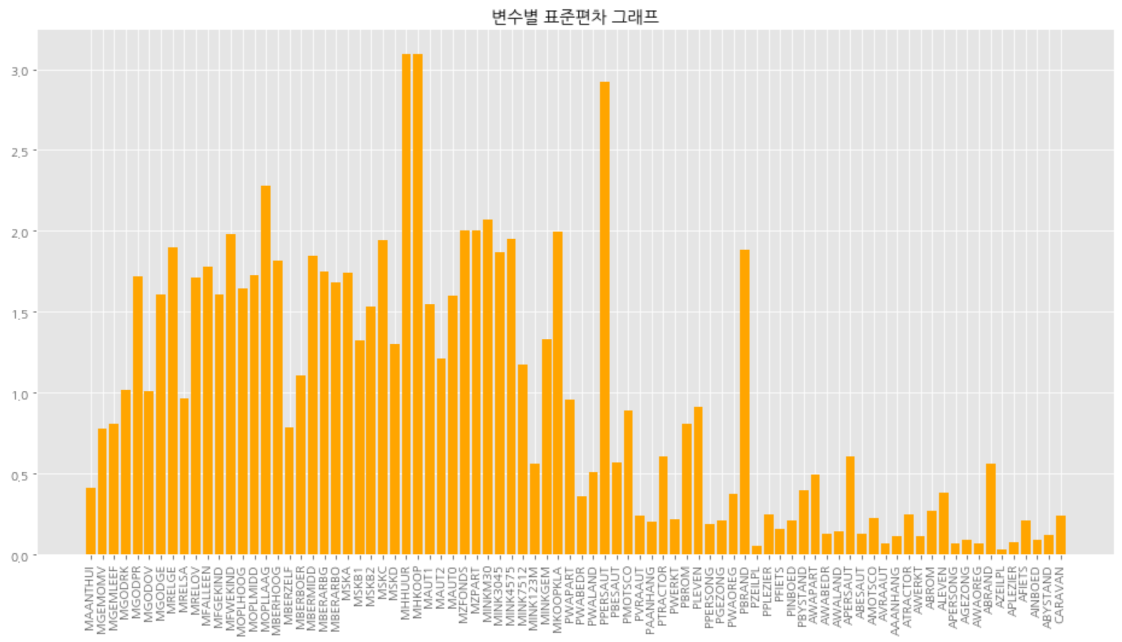
```
In [168]: continuous_ticdata=ticdata.drop(['MOSTYPE','MOSHOOFD'],1)
          stats=continuous_ticdata.describe()

In [169]: stats = continuous_ticdata.describe()
          print(stats) ###
```

```
          MAANTHUI      MGEMOMV     MGEMLEEF       MGODRK       MGODPR  \
count  9822.000000  9822.000000  9822.000000  9822.000000  9822.000000
mean      1.108735     2.677561     2.996437     0.700672     4.637650
std       0.412101     0.780701     0.804660     1.015107     1.721212
min       1.000000     1.000000     1.000000     0.000000     0.000000
25%       1.000000     2.000000     2.000000     0.000000     4.000000
50%       1.000000     3.000000     3.000000     0.000000     5.000000
75%       1.000000     3.000000     3.000000     1.000000     6.000000
max      10.000000     6.000000     6.000000     9.000000     9.000000

            MGODOV       MGODGE       MRELGE       MRELSA       MRELOV  ...  \
count  9822.000000  9822.000000  9822.000000  9822.000000  9822.000000  ...
mean      1.050092     3.262981     6.188964     0.873142     2.286602  ...
std       1.011156     1.606287     1.896070     0.961955     1.710674  ...
min       0.000000     0.000000     0.000000     0.000000     0.000000  ...
25%       0.000000     2.000000     5.000000     0.000000     1.000000  ...
50%       1.000000     3.000000     6.000000     1.000000     2.000000  ...
75%       2.000000     4.000000     7.000000     1.000000     3.000000  ...
max       5.000000     9.000000     9.000000     7.000000     9.000000  ...

          APERSONG      AGEZONG      AWAOREG       ABRAND      AZEILPL  \
count  9822.000000  9822.000000  9822.000000  9822.000000  9822.000000
mean      0.004582     0.007941     0.004276     0.574018     0.000916
std       0.067535     0.088764     0.071224     0.561255     0.030258
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     1.000000     0.000000
75%       0.000000     0.000000     0.000000     1.000000     0.000000
max       1.000000     1.000000     2.000000     7.000000     1.000000

          APLEZIER      AFIETS      AINBOED     ABYSTAND      CARAVAN
count  9822.000000  9822.00000  9822.000000  9822.000000  9822.000000
mean      0.005091     0.03146     0.008450     0.013846     0.059662
std       0.077996     0.20907     0.092647     0.117728     0.236872
min       0.000000     0.00000     0.000000     0.000000     0.000000
25%       0.000000     0.00000     0.000000     0.000000     0.000000
50%       0.000000     0.00000     0.000000     0.000000     0.000000
75%       0.000000     0.00000     0.000000     0.000000     0.000000
max       2.000000     4.00000     2.000000     2.000000     1.000000

[8 rows x 84 columns]
```

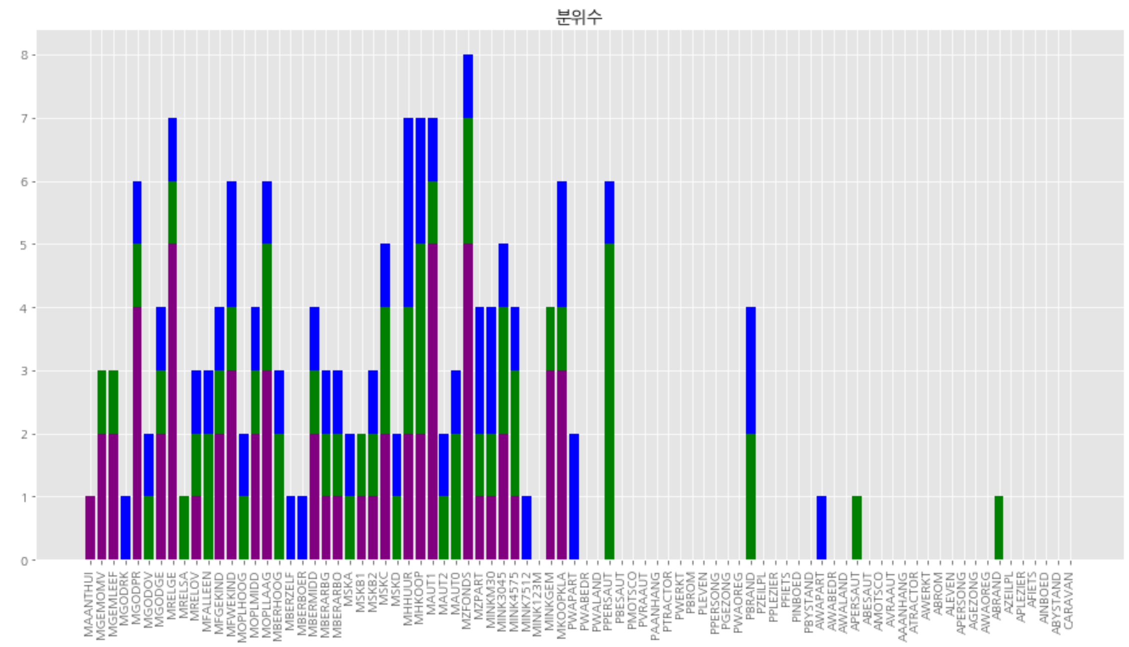
```
In [170]: ###
          fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = continuous_ticdata.columns
          x_pos = np.arange(len(objects))
          ax1 = plt.bar(x_pos, stats.loc['mean'],color="purple" ,alpha=1)
          plt.xticks(x_pos, objects)
          plt.xticks(rotation=90);
          plt.title(' ', size=14)

Out[170]: Text(0.5, 1.0, ' ')
```



```
In [171]: ###
          fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = continuous_ticdata.columns
          x_pos = np.arange(len(objects))
          ax1 = plt.bar(x_pos, stats.loc['std'],color="orange" ,alpha=1)
          plt.xticks(x_pos, objects)
          plt.xticks(rotation=90);
          plt.title('   ', size=14)

Out[171]: Text(0.5, 1.0, '   ')
```



```
In [172]: #
          fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = continuous_ticdata.columns
          x_pos = np.arange(len(objects))
          ax1 = plt.bar(x_pos, stats.loc['75%'],color="blue" ,alpha=1)
          ax1 = plt.bar(x_pos, stats.loc['50%'],color="green" ,alpha=1)
          ax1 = plt.bar(x_pos, stats.loc['25%'],color="purple" ,alpha=1)
          plt.xticks(x_pos, objects)
          plt.xticks(rotation=90);
          plt.title('', size=14)

Out[172]: Text(0.5, 1.0, '')
```



0.0.7 75th 0 0

```
In [173]: num_zeros = []
          for i in range(0, len(ticdata.columns)):
              num_nonzero = len(ticdata.iloc[:,i].nonzero()[0])
              num_zeros.append(ticdata.shape[0] - num_nonzero)

C:\Users\jang\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: FutureWarning: Series.nonze
  This is separate from the ipykernel package so we can avoid doing imports until
```

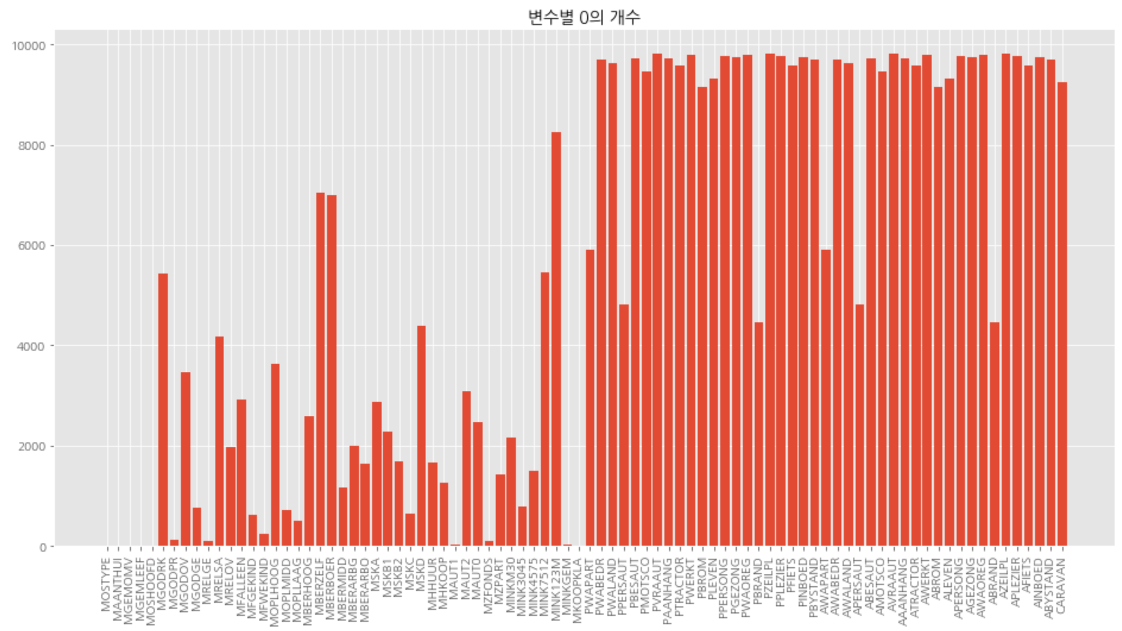
```
In [174]: num_zeros=pd.Series(num_zeros)
          num_zeros.index=ticdata.columns ##
          print(num_zeros)

MOSTYPE        0
MAANTHUI       0
MGEMOMV        0
MGEMLEEF       0
MOSHOOFD       0
MGODRK      5420
MGODPR       127
MGODOV      3460
MGODGE       773
MRELGE       108
MRELSA      4185
MRELOV      1981
```

```
MFALLEEN    2916
MFGEKIND     613
MFWEKIND     243
MOPLHOOG    3621
MOPLMIDD     711
MOPLLAAG     494
MBERHOOG    2576
MBERZELF    7031
MBERBOER    6985
MBERMIDD    1164
MBERARBG    1995
MBERARBO    1636
MSKA        2871
MSKB1       2275
MSKB2       1694
MSKC         634
MSKD        4376
MHHUUR      1663
            ...
PGEZONG     9744
PWAOREG     9784
PBRAND      4464
PZEILPL     9813
PPLEZIER    9777
PFIETS      9573
PINBOED     9740
PBYSTAND    9687
AWAPART     5903
AWABEDR     9688
AWALAND     9613
APERSAUT    4825
ABESAUT     9730
AMOTSCO     9460
AVRAAUT     9808
AAANHANG    9719
ATRACTOR    9576
AWERKT      9790
ABROM       9150
ALEVEN      9308
APERSONG    9777
AGEZONG     9744
AWAOREG     9784
ABRAND      4464
AZEILPL     9813
APLEZIER    9777
AFIETS      9573
AINBOED     9740
ABYSTAND    9687
```

```
CARAVAN     9236
Length: 86, dtype: int64
```

```
In [175]: # Plot number of zero values for each feature in order.
          fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = ticdata.columns
          x_pos = np.arange(len(objects))
          ax1 = plt.bar(x_pos, num_zeros)
          plt.xticks(x_pos, objects)
          plt.xticks(rotation=90);
          plt.title(' 0 ', size=14)
```

```
Out[175]: Text(0.5, 1.0, ' 0 ')
```



0.0.8 MOSTYPE,MOSHOOFD

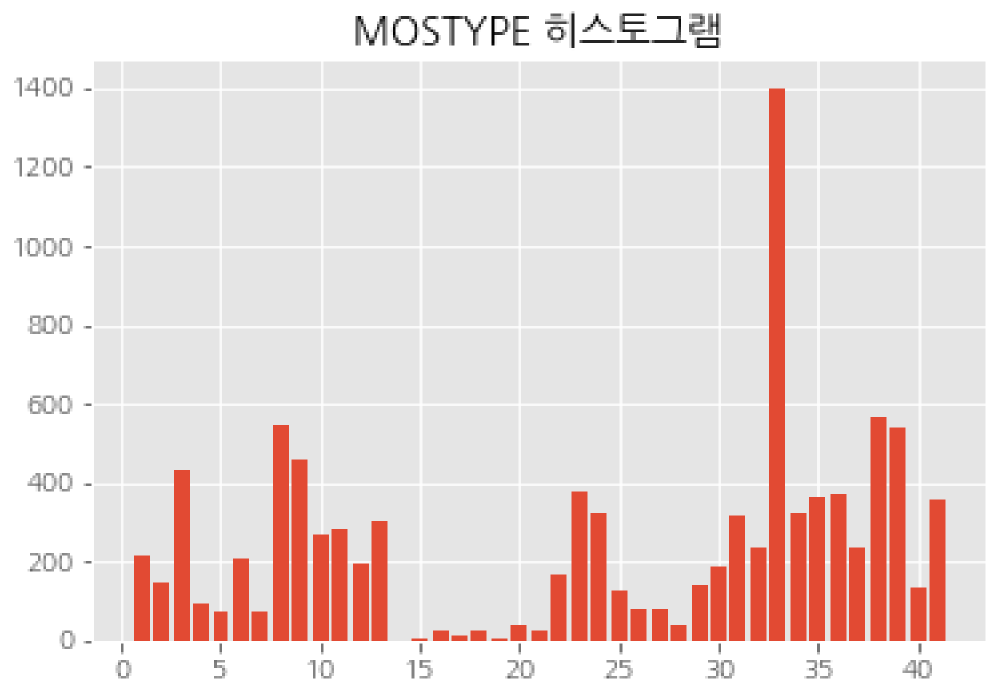
```
In [176]: ticdata["MOSTYPE"]=ticdata["MOSTYPE"].astype('category')

          ticdata["MOSHOOFD"]=ticdata["MOSHOOFD"].astype('category')
```

```
In [177]: Mostype_hist=pd.Series(ticdata['MOSTYPE'].value_counts())
          Mostype_hist=Mostype_hist.sort_index()
          print(Mostype_hist)
```

```
1       218
2       148
```

```
3      433
4       90
5       70
6      209
7       72
8      546
9      460
10     271
11     286
12     194
13     302
15       7
16      25
17      13
18      27
19       7
20      42
21      29
22     169
23     376
24     324
25     129
26      79
27      77
28      41
29     139
30     190
31     318
32     234
33    1401
34     325
35     362
36     373
37     233
38     569
39     542
40     137
41     355
Name: MOSTYPE, dtype: int64
```

```
In [178]: plt.bar(Mostype_hist.index,Mostype_hist)
          plt.title("MOSTYPE ")

Out[178]: Text(0.5, 1.0, 'MOSTYPE ')
```

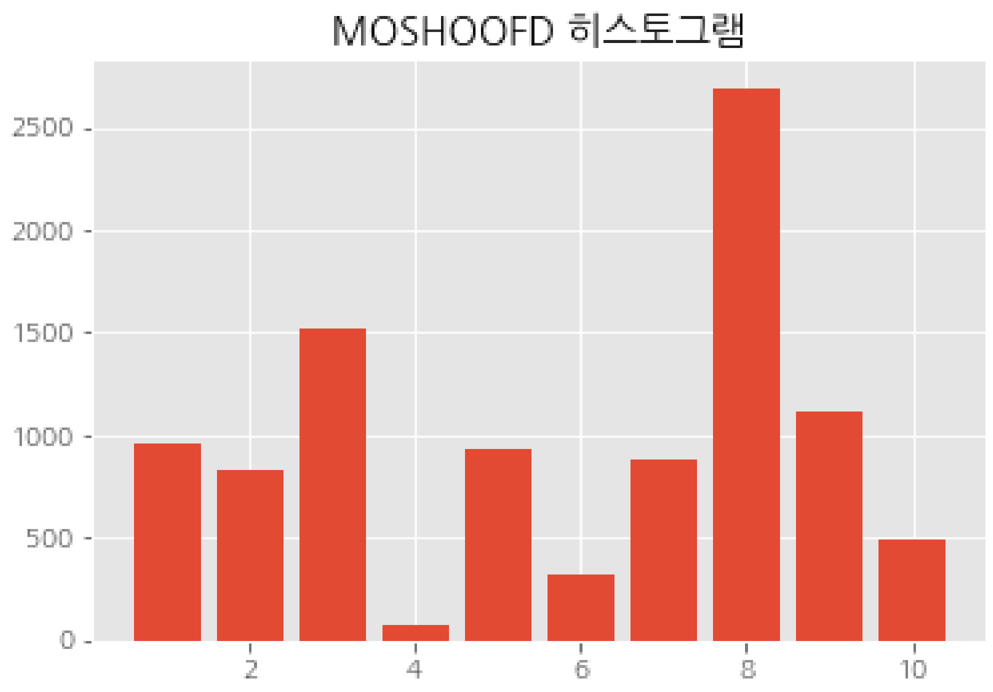


```
In [179]: MoshooFD_hist=pd.Series(ticdata['MOSHOOFD'].value_counts())
          MoshooFD_hist=MoshooFD_hist.sort_index()
          print(MoshooFD_hist)
```

```
1      959
2      827
3     1513
4       79
5      940
6      326
7      881
8     2694
9     1111
10     492
Name: MOSHOOFD, dtype: int64
```

```
In [180]: plt.bar(MoshooFD_hist.index,MoshooFD_hist)
          plt.title("MOSHOOFD ")
```

```
Out[180]: Text(0.5, 1.0, 'MOSHOOFD ')
```



0.0.9 ( )

```
In [181]: stat=ticdata.describe()
```

```
In [182]: stat.loc[['min','max']].T
```

```
Out[182]:           min   max
          MAANTHUI  1.0  10.0
          MGEMOMV   1.0   6.0
          MGEMLEEF  1.0   6.0
          MGODRK    0.0   9.0
          MGODPR    0.0   9.0
          MGODOV    0.0   5.0
          MGODGE    0.0   9.0
          MRELGE    0.0   9.0
          MRELSA    0.0   7.0
          MRELOV    0.0   9.0
          MFALLEEN  0.0   9.0
          MFGEKIND  0.0   9.0
          MFWEKIND  0.0   9.0
          MOPLHOOG  0.0   9.0
          MOPLMIDD  0.0   9.0
          MOPLLAAG  0.0   9.0
          MBERHOOG  0.0   9.0
          MBERZELF  0.0   5.0
```

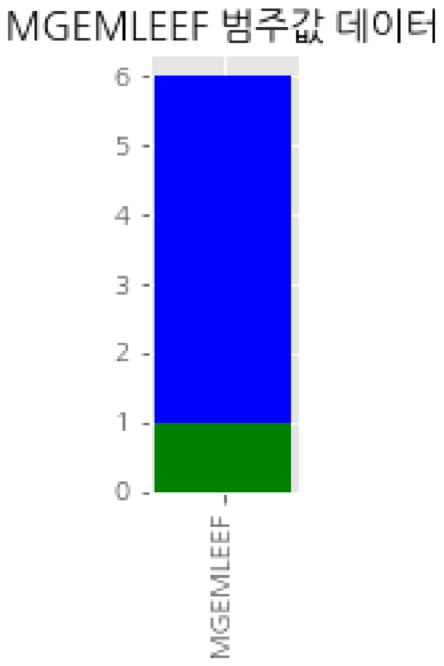
```
MBERBOER  0.0   9.0
MBERMIDD  0.0   9.0
MBERARBG  0.0   9.0
MBERARBO  0.0   9.0
MSKA      0.0   9.0
MSKB1     0.0   9.0
MSKB2     0.0   9.0
MSKC      0.0   9.0
MSKD      0.0   9.0
MHHUUR    0.0   9.0
MHKOOP    0.0   9.0
MAUT1     0.0   9.0
...       ...   ...
PGEZONG   0.0   3.0
PWAOREG   0.0   7.0
PBRAND    0.0   8.0
PZEILPL   0.0   3.0
PPLEZIER  0.0   6.0
PFIETS    0.0   1.0
PINBOED   0.0   6.0
PBYSTAND  0.0   5.0
AWAPART   0.0   2.0
AWABEDR   0.0   5.0
AWALAND   0.0   1.0
APERSAUT  0.0  12.0
ABESAUT   0.0   5.0
AMOTSCO   0.0   8.0
AVRAAUT   0.0   4.0
AAANHANG  0.0   3.0
ATRACTOR  0.0   6.0
AWERKT    0.0   6.0
ABROM     0.0   3.0
ALEVEN    0.0   8.0
APERSONG  0.0   1.0
AGEZONG   0.0   1.0
AWAOREG   0.0   2.0
ABRAND    0.0   7.0
AZEILPL   0.0   1.0
APLEZIER  0.0   2.0
AFIETS    0.0   4.0
AINBOED   0.0   2.0
ABYSTAND  0.0   2.0
CARAVAN   0.0   1.0

[84 rows x 2 columns]
```

```
In [183]: L3columns=ticdata.columns[5:43]
```

```
In [184]: L4columns=ticdata.columns[43:64]
In [185]: one_to_12columns=ticdata.columns[64:86]
```

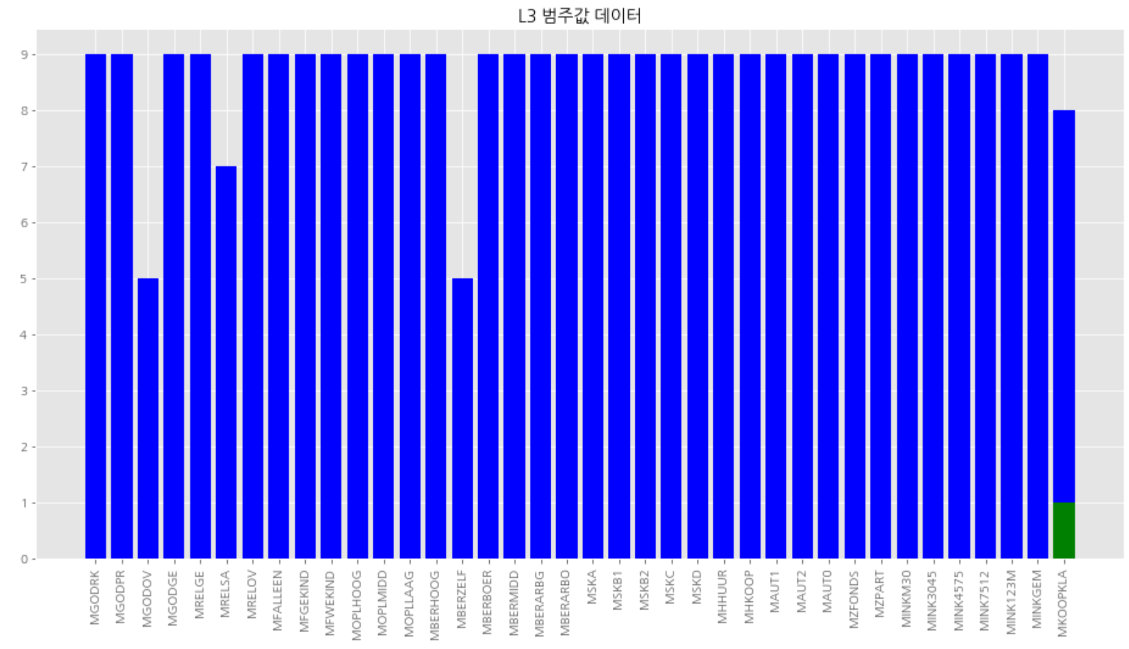
```
In [186]: fig = plt.figure(figsize=(1,3))
          ax1 = fig.add_subplot(111)
          objects = ['MGEMLEEF']
          x_pos = np.arange(len(objects))
          ax1 = plt.bar(x_pos, stats['MGEMLEEF'].loc['max'],color="blue" ,alpha=1)
          ax1 = plt.bar(x_pos, stats['MGEMLEEF'].loc['min'],color="green" ,alpha=1)
          plt.xticks(x_pos, objects)
          plt.xticks(rotation=90);
          plt.yticks(range(7))
          plt.title('MGEMLEEF  ', size=14)
```

```
Out[186]: Text(0.5, 1.0, 'MGEMLEEF  ')
```

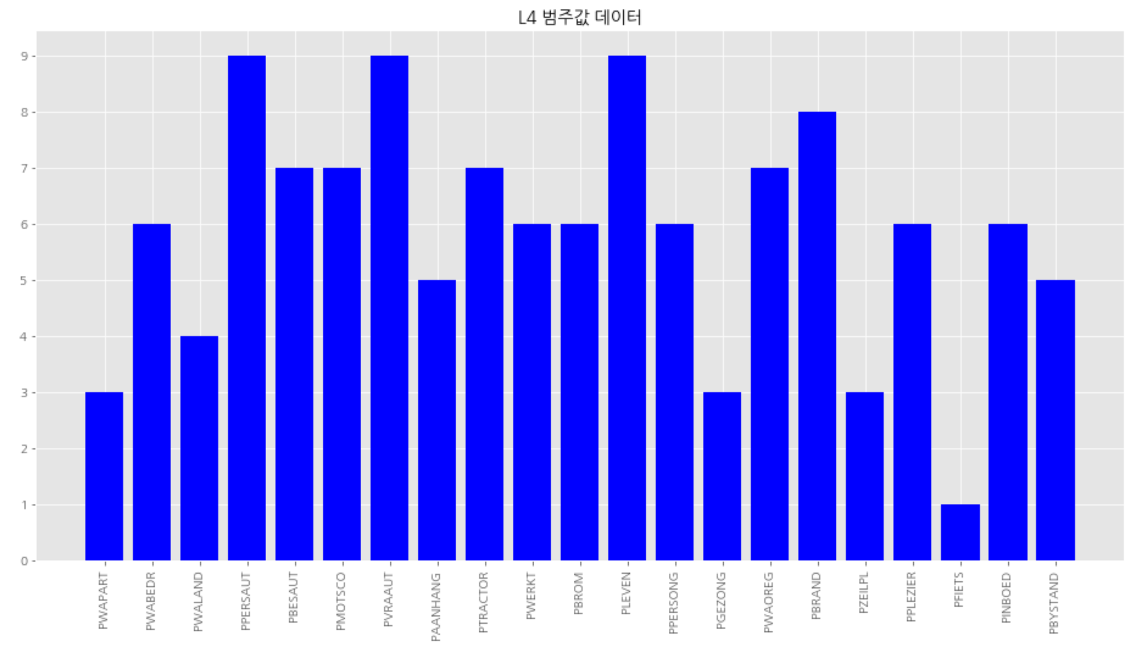


```
In [187]: fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = ticdata[L3columns].columns
          x_pos = np.arange(len(objects))
          ax1 = plt.bar(x_pos, stats[L3columns].loc['max'],color="blue" ,alpha=1)
          ax1 = plt.bar(x_pos, stats[L3columns].loc['min'],color="green" ,alpha=1)
          plt.xticks(x_pos, objects)
          plt.xticks(rotation=90);
          plt.yticks(range(10))
          plt.title('L3  ', size=14)
```

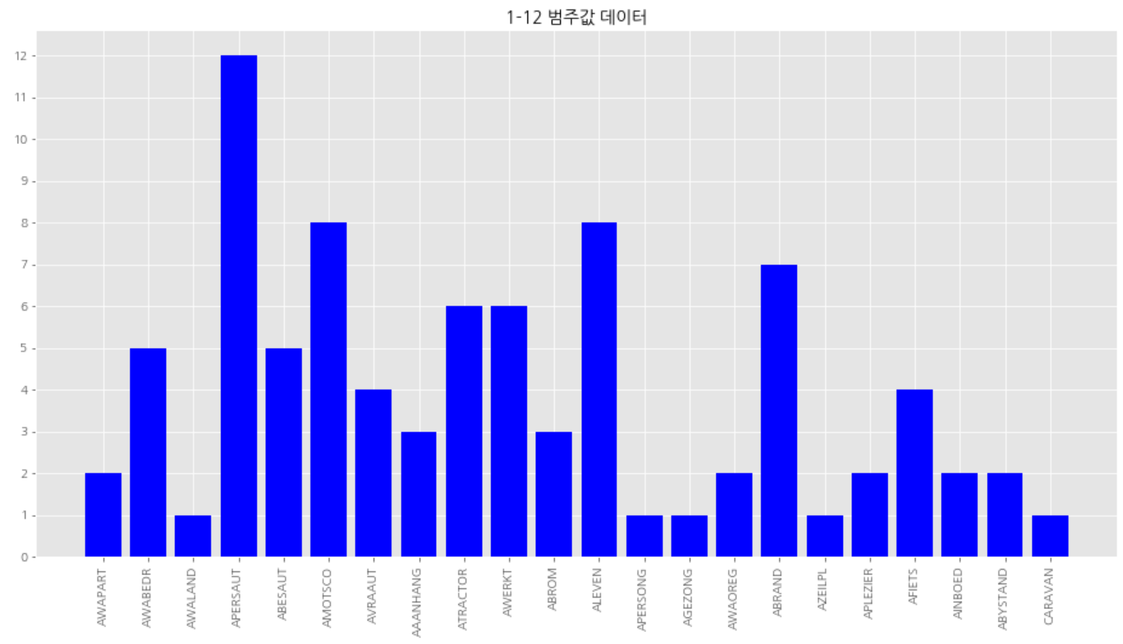
```
Out[187]: Text(0.5, 1.0, 'L3  ')
```



```
In [188]: fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = ticdata[L4columns].columns
          x_pos = np.arange(len(objects))
          ax1 = plt.bar(x_pos, stats[L4columns].loc['max'],color="blue" ,alpha=1)
          ax1 = plt.bar(x_pos, stats[L4columns].loc['min'],color="green" ,alpha=1)
          plt.xticks(x_pos, objects)
          plt.xticks(rotation=90);
          plt.yticks(range(10))
          plt.title('L4  ', size=14)

Out[188]: Text(0.5, 1.0, 'L4  ')
```



```
In [189]: fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = ticdata[one_to_12columns].columns
          x_pos = np.arange(len(objects))
          ax1 = plt.bar(x_pos, stats[one_to_12columns].loc['max'],color="blue" ,alpha=1)
          ax1 = plt.bar(x_pos, stats[one_to_12columns].loc['min'],color="green" ,alpha=1)
          plt.xticks(x_pos, objects)
          plt.xticks(rotation=90);
          plt.yticks(range(13))
          plt.title('1-12  ', size=14)

Out[189]: Text(0.5, 1.0, '1-12  ')
```



0.0.10

```
In [190]: ###MGMLEEF
          ticdata.loc[ticdata['MGEMLEEF']==1,'MGEMLEEF']=25
          ticdata.loc[ticdata['MGEMLEEF']==2,'MGEMLEEF']=35
          ticdata.loc[ticdata['MGEMLEEF']==3,'MGEMLEEF']=45
          ticdata.loc[ticdata['MGEMLEEF']==4,'MGEMLEEF']=55
          ticdata.loc[ticdata['MGEMLEEF']==5,'MGEMLEEF']=65
          ticdata.loc[ticdata['MGEMLEEF']==6,'MGEMLEEF']=75
```

```
In [191]: ###L3
          for i in ticdata.columns[5:43] :
              ticdata.loc[ticdata.loc[:,i]==0,i]=0
              ticdata.loc[ticdata.loc[:,i]==1,i]=5
              ticdata.loc[ticdata.loc[:,i]==2,i]=17
              ticdata.loc[ticdata.loc[:,i]==3,i]=30
              ticdata.loc[ticdata.loc[:,i]==4,i]=43
              ticdata.loc[ticdata.loc[:,i]==5,i]=56
              ticdata.loc[ticdata.loc[:,i]==6,i]=69
              ticdata.loc[ticdata.loc[:,i]==7,i]=82
              ticdata.loc[ticdata.loc[:,i]==8,i]=95
              ticdata.loc[ticdata.loc[:,i]==9,i]=100
```

```
In [192]: ###L4
          for i in ticdata.columns[43:64] :
              ticdata.loc[ticdata.loc[:,i]==0,i]=0
              ticdata.loc[ticdata.loc[:,i]==1,i]=25
```

```
        ticdata.loc[ticdata.loc[:,i]==2,i]=75
        ticdata.loc[ticdata.loc[:,i]==3,i]=150
        ticdata.loc[ticdata.loc[:,i]==4,i]=350
        ticdata.loc[ticdata.loc[:,i]==5,i]=750
        ticdata.loc[ticdata.loc[:,i]==6,i]=3000
        ticdata.loc[ticdata.loc[:,i]==7,i]=7500
        ticdata.loc[ticdata.loc[:,i]==8,i]=15000
        ticdata.loc[ticdata.loc[:,i]==9,i]=20000
```

0.0.11 :

```
In [193]: allticdata=ticdata.copy()
```

```
In [194]: raw_sample=allticdata ####
          raw_sample_0=raw_sample[raw_sample['CARAVAN']==0]
          raw_sample_1=raw_sample[raw_sample['CARAVAN']==1]
```

```
In [195]: corrdf=raw_sample.corr().stack()['CARAVAN']
```

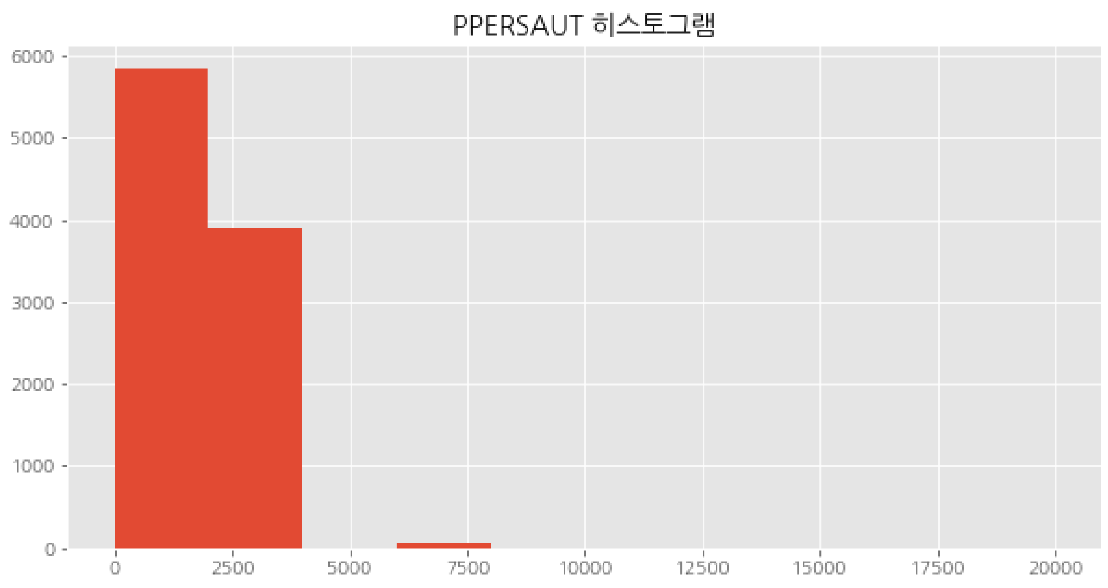
```
In [196]: np.round(abs(corrdf).sort_values()[::-1],2)
```

```
Out[196]: CARAVAN     1.00
          PPERSAUT    0.14
          APERSAUT    0.13
          PWAPART     0.10
          MKOOPKLA    0.09
          AWAPART     0.09
          MINKGEM     0.09
          APLEZIER    0.08
          MOPLLAAG    0.08
          MAUT1       0.07
          MAUT0       0.07
          MHHUUR      0.07
          MRELGE      0.07
          MHKOOP      0.06
          MZFONDS     0.06
          ABRAND      0.06
          MINKM30     0.06
          PPLEZIER    0.06
          MBERBOER    0.06
          MRELOV      0.05
          PBYSTAND    0.05
          ABYSTAND    0.05
          PGEZONG     0.05
          ALEVEN      0.05
          MGEMOMV     0.05
          MOPLMIDD    0.04
          MOPLHOOG    0.04
          AGEZONG     0.04
```

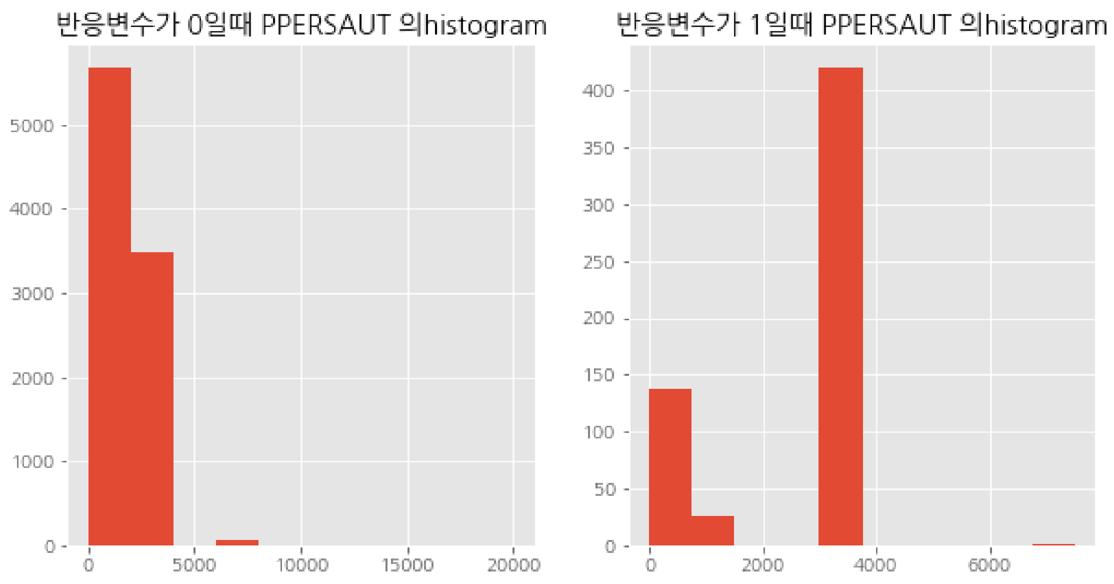
```
MINK4575    0.04
MSKC        0.04
            ...
AAANHANG    0.01
AWERKT      0.01
MRELSA      0.01
ABESAUT     0.01
PBRAND      0.01
PMOTSCO     0.01
MBERARBG    0.01
PBESAUT     0.01
AWAOREG     0.01
PWABEDR     0.01
PWAOREG     0.01
AVRAAUT     0.01
MSKB2       0.01
MAUT2       0.01
PWERKT      0.01
PAANHANG    0.01
PVRAAUT     0.01
PTRACTOR    0.01
PPERSONG    0.01
MSKB1       0.01
MINK3045    0.01
MINK123M    0.00
APERSONG    0.00
MGEMLEEF    0.00
AWABEDR     0.00
PINBOED     0.00
PLEVEN      0.00
MAANTHUI    0.00
AMOTSCO     0.00
MFGEKIND    0.00
Length: 84, dtype: float64
```

0.0.12 PPERSAUT ( )

```
In [197]: plt.figure(figsize=(10,5))
          plt.hist(raw_sample["PPERSAUT"])
          plt.title("PPERSAUT ")

Out[197]: Text(0.5, 1.0, 'PPERSAUT ')
```



```
In [198]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(raw_sample_0["PPERSAUT"])
          axes1.set_title(" 0 PPERSAUT histogram")
          axes2=plt.subplot(1,2,2)
          axes2.hist(raw_sample_1["PPERSAUT"])
          axes2.set_title(" 1 PPERSAUT histogram")

Out[198]: Text(0.5, 1.0, ' 1 PPERSAUT histogram')
```



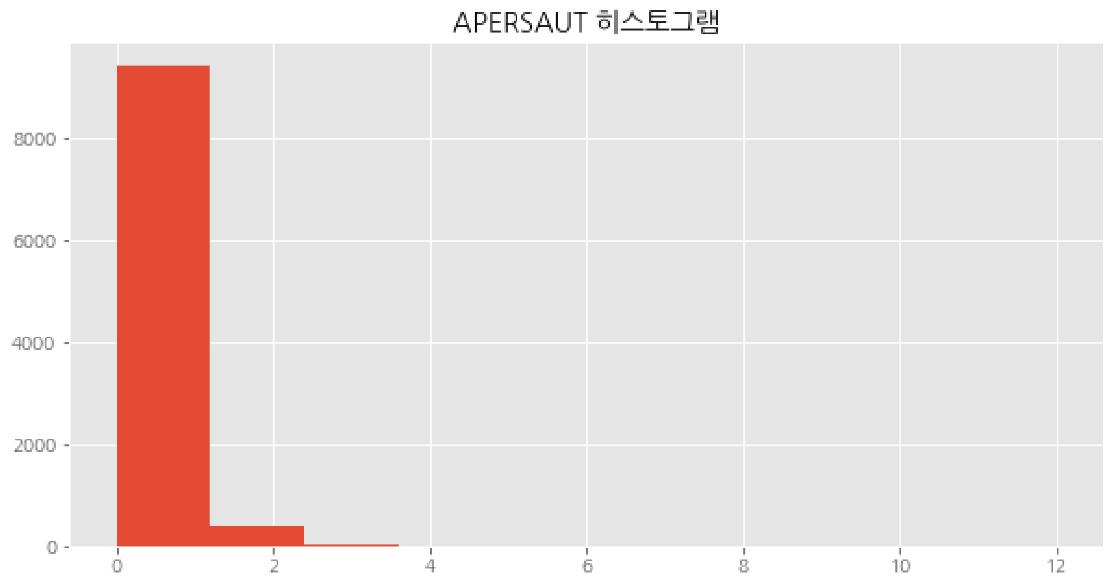
```
In [199]: PPERSAUT_table=pd.concat([raw_sample_0['PPERSAUT'].value_counts(),raw_sample_1['PPERS
          PPERSAUT_table.columns=[0,1]
          PPERSAUT_table=PPERSAUT_table.fillna(1)
          for i in range(PPERSAUT_table.shape[0]):
              print("PPERSAUT ",PPERSAUT_table.index[i],"",PPERSAUT_table.sum(1).iloc[i]," ",PI
```

```
PPERSAUT  0  4825.0   0.029443140601664176
PPERSAUT  350  5.0   0.25
PPERSAUT  750  1013.0   0.02634245187436677
PPERSAUT  3000  3910.0   0.12034383954154727
PPERSAUT  7500  64.0   0.03225806451612903
PPERSAUT  15000  6.0   0.2
PPERSAUT  20000  2.0   1.0
```

0.0.13 APERSAUT ( )

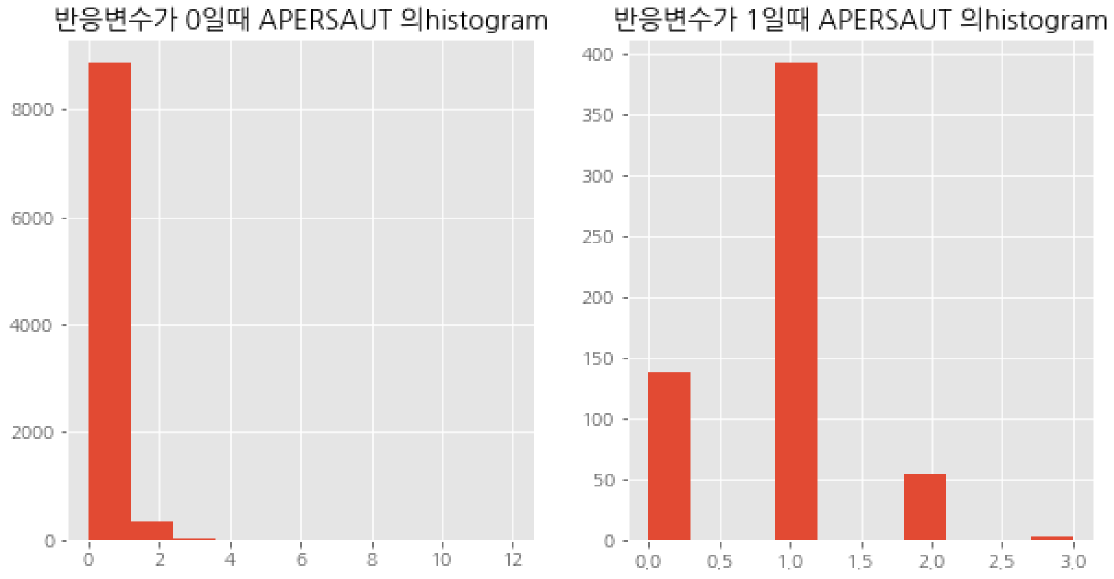
```
In [200]: plt.figure(figsize=(10,5))
          plt.hist(raw_sample["APERSAUT"])
          plt.title("APERSAUT ")
```

```
Out[200]: Text(0.5, 1.0, 'APERSAUT ')
```



```
In [201]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(raw_sample_0["APERSAUT"])
          axes1.set_title(" 0 APERSAUT histogram")
          axes2=plt.subplot(1,2,2)
          axes2.hist(raw_sample_1["APERSAUT"])
          axes2.set_title(" 1 APERSAUT histogram")
```

```
Out[201]: Text(0.5, 1.0, ' 1 APERSAUT histogram')
```

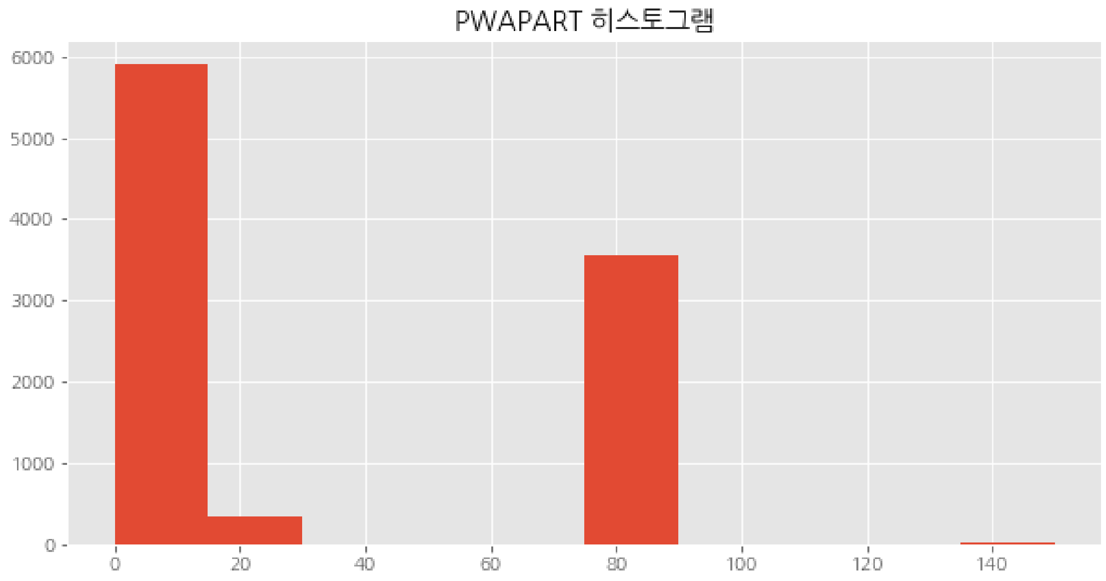


```
In [202]: APERSAUT_table=pd.concat([raw_sample_0['APERSAUT'].value_counts(),raw_sample_1['APERS
          APERSAUT_table.columns=[0,1]
          APERSAUT_table=APERSAUT_table.fillna(1)
          for i in range(APERSAUT_table.shape[0]):
              print("APERSAUT ",APERSAUT_table.index[i],"",APERSAUT_table.sum(1).iloc[i]," ",AI
```

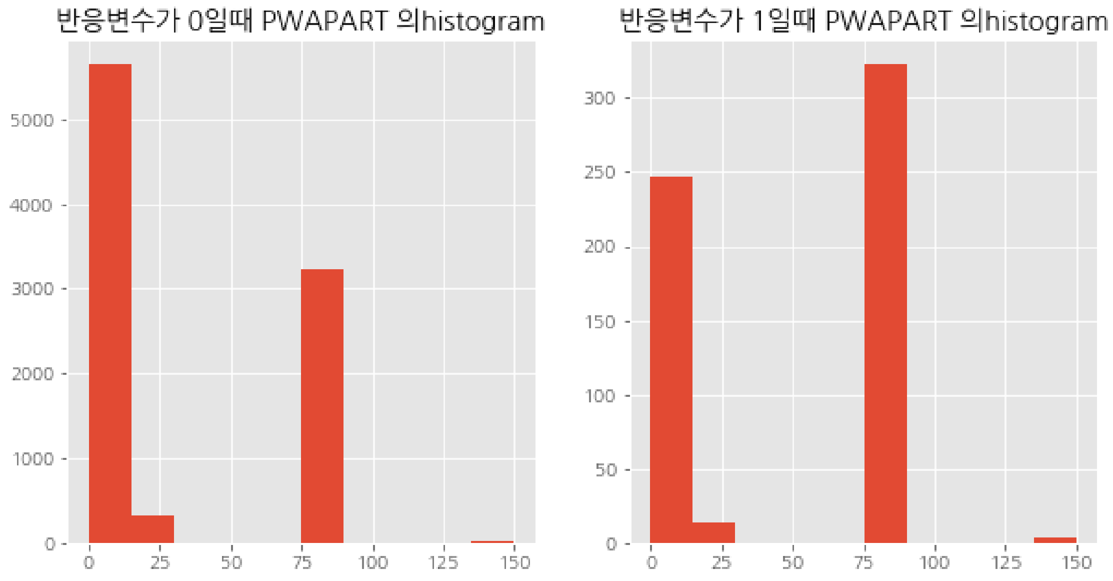
```
APERSAUT  0  4825.0   0.029443140601664176
APERSAUT  1  4580.0   0.0936007640878701
APERSAUT  2  384.0   0.16363636363636364
APERSAUT  3  21.0   0.10526315789473684
APERSAUT  4  9.0   0.125
APERSAUT  5  2.0   1.0
APERSAUT  6  2.0   1.0
APERSAUT  7  2.0   1.0
APERSAUT  12  2.0   1.0
```

0.0.14 PWAPART ()

```
In [203]: plt.figure(figsize=(10,5))
          plt.hist(raw_sample["PWAPART"])
          plt.title("PWAPART ")

Out[203]: Text(0.5, 1.0, 'PWAPART ')
```



```
In [204]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(raw_sample_0["PWAPART"])
          axes1.set_title(" 0 PWAPART histogram")
          axes2=plt.subplot(1,2,2)
          axes2.hist(raw_sample_1["PWAPART"])
          axes2.set_title(" 1 PWAPART histogram")

Out[204]: Text(0.5, 1.0, ' 1 PWAPART histogram')
```



```
In [205]: PWAPART_table=pd.concat([raw_sample_0['PWAPART'].value_counts(),raw_sample_1['PWAPAR
          PWAPART_table.columns=[0,1]
          PWAPART_table=PWAPART_table.fillna(1)
          for i in range(PWAPART_table.shape[0]):
              print("PWAPART ",PWAPART_table.index[i],"",PWAPART_table.sum(1).iloc[i]," ",PWAP
```

```
PWAPART  0  5903   0.04367043847241867
PWAPART  25  341   0.039634146341463415
PWAPART  75  3562   0.09972213646187095
PWAPART  150  16   0.23076923076923078
```

0.0.15 AWAPART

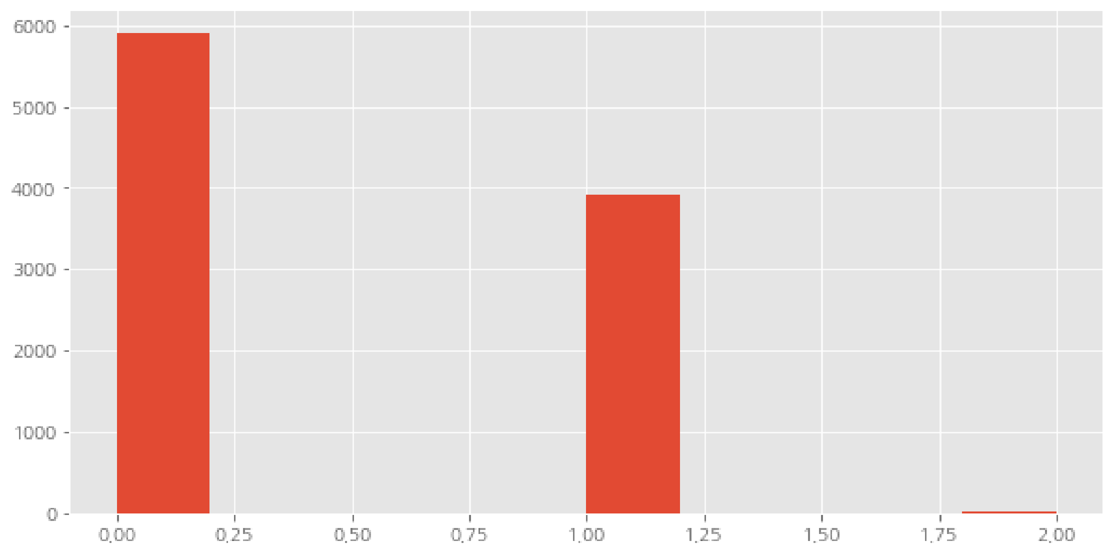
```
In [206]: raw_sample["AWAPART"]
```

```
Out[206]: 0       1
          1       1
          2       1
          3       1
          4       1
          5       0
          6       1
          7       1
          8       1
          9       1
          10      0
          11      1
          12      0
          13      1
          14      0
          15      0
          16      1
          17      1
          18      0
          19      0
          20      0
          21      1
          22      0
          23      1
          24      1
          25      1
          26      0
          27      0
          28      0
          29      1
                 ..
          9792    1
```

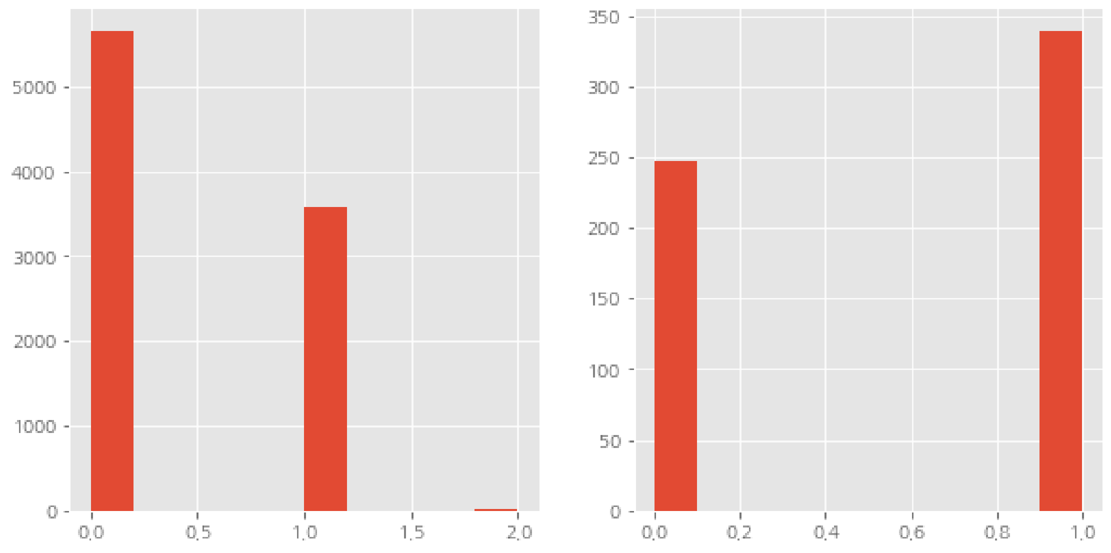
```
9793    0
9794    0
9795    0
9796    0
9797    0
9798    0
9799    1
9800    1
9801    0
9802    0
9803    0
9804    1
9805    0
9806    0
9807    1
9808    1
9809    0
9810    0
9811    0
9812    0
9813    1
9814    1
9815    1
9816    0
9817    1
9818    0
9819    1
9820    0
9821    1
Name: AWAPART, Length: 9822, dtype: int64
```

```
In [207]: plt.figure(figsize=(10,5))
          plt.hist(raw_sample["AWAPART"])
```

```
Out[207]: (array([5903.,    0.,    0.,    0.,    0., 3909.,    0.,    0.,    0.,
                    10.]),
           array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. ]),
           <a list of 10 Patch objects>)
```



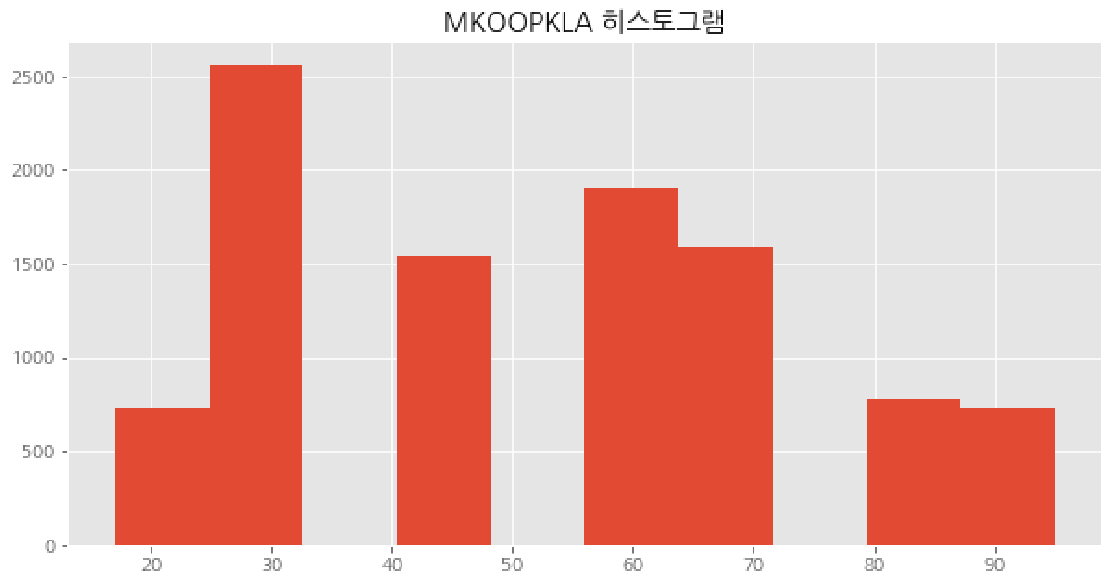
```
In [208]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(raw_sample_0["AWAPART"])
          axes2=plt.subplot(1,2,2)
          axes2.hist(raw_sample_1["AWAPART"])

Out[208]: (array([247.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 339.]),
           array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
           <a list of 10 Patch objects>)
```



```
In [209]: raw_sample['AWAPART'].unique()

Out[209]: array([1, 0, 2], dtype=int64)

In [210]: AWAPART_table=pd.concat([raw_sample_0['AWAPART'].value_counts(),raw_sample_1['AWAPAR
          AWAPART_table.columns=[0,1]
          AWAPART_table=AWAPART_table.fillna(1)
          for i in range(AWAPART_table.shape[0]):
              print("AWAPART ",AWAPART_table.index[i],"",AWAPART_table.sum(1).iloc[i]," ",AWAP

AWAPART  0  5903.0   0.04367043847241867
AWAPART  1  3909.0   0.0949579831932773
AWAPART  2  11.0   0.1
```

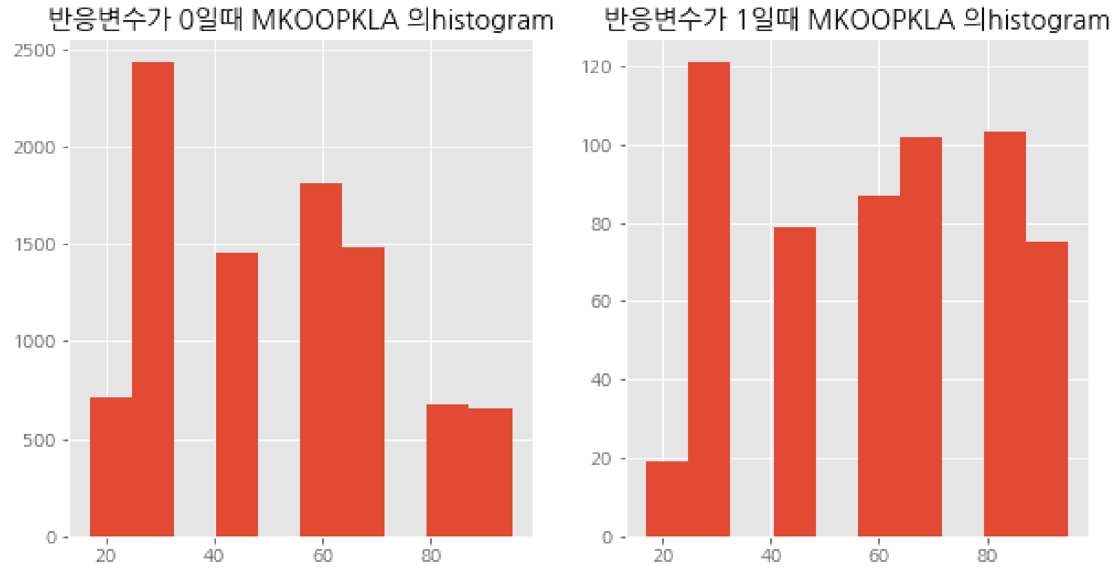
0.0.16 MKOOPKLA ()

```
In [211]: plt.figure(figsize=(10,5))
          plt.hist(raw_sample["MKOOPKLA"])
          plt.title("MKOOPKLA ")

Out[211]: Text(0.5, 1.0, 'MKOOPKLA ')
```



```
In [212]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(raw_sample_0["MKOOPKLA"])
          axes1.set_title(" 0 MKOOPKLA histogram")
          axes2=plt.subplot(1,2,2)
          axes2.hist(raw_sample_1["MKOOPKLA"])
          axes2.set_title(" 1 MKOOPKLA histogram")
```

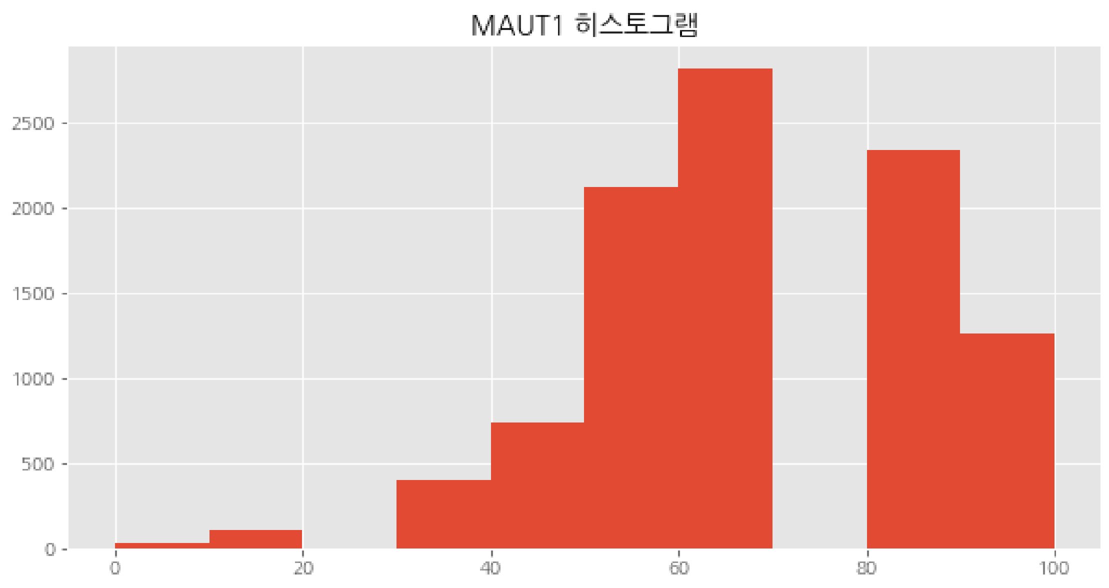
```
Out[212]: Text(0.5, 1.0, ' 1 MKOOPKLA histogram')
```



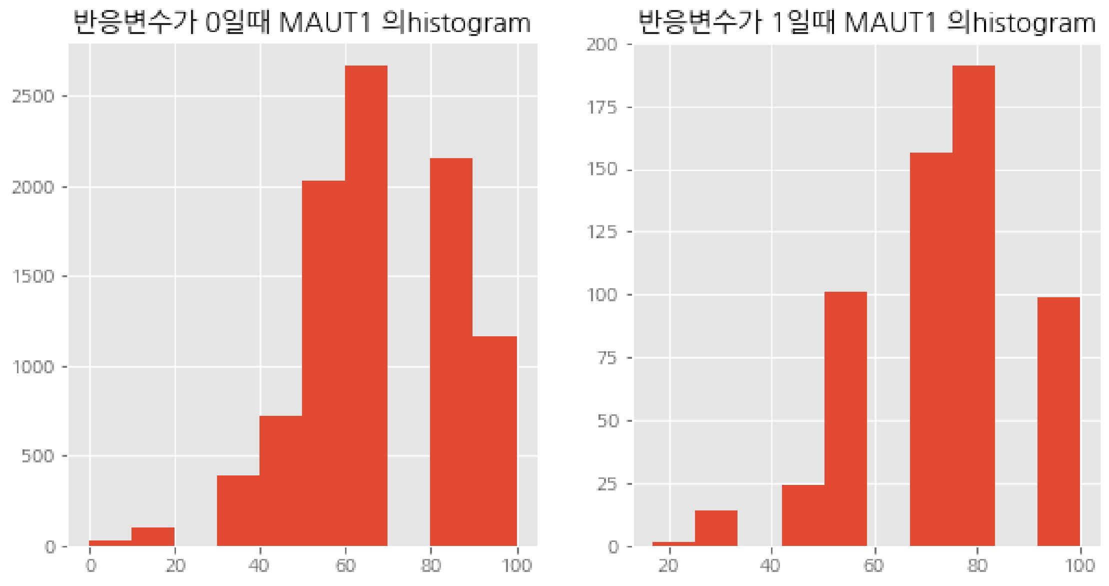
```
In [213]: MKOOPKLA_table=pd.concat([raw_sample_0['MKOOPKLA'].value_counts(),raw_sample_1['MKOO
          MKOOPKLA_table.columns=[0,1]
          MKOOPKLA_table=MKOOPKLA_table.fillna(1)
          for i in range(MKOOPKLA_table.shape[0]):
              print("MKOOPKLA ",MKOOPKLA_table.index[i],"",MKOOPKLA_table.sum(1).iloc[i]," ",MK

MKOOPKLA  17  731   0.026685393258426966
MKOOPKLA  30  2556   0.04969199178644764
MKOOPKLA  43  1539   0.05410958904109589
MKOOPKLA  56  1902   0.047933884297520664
MKOOPKLA  69  1587   0.06868686868686869
MKOOPKLA  82  777   0.15281899109792285
MKOOPKLA  95  730   0.11450381679389313
```

0.0.17 MAUT1 ()

```
In [214]: plt.figure(figsize=(10,5))
          plt.hist(raw_sample["MAUT1"])
          plt.title("MAUT1 ")

Out[214]: Text(0.5, 1.0, 'MAUT1 ')
```



```
In [215]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(raw_sample_0["MAUT1"])
          axes1.set_title(" 0 MAUT1 histogram")
          axes2=plt.subplot(1,2,2)
          axes2.hist(raw_sample_1["MAUT1"])
          axes2.set_title(" 1 MAUT1 histogram")

Out[215]: Text(0.5, 1.0, ' 1 MAUT1 histogram')
```



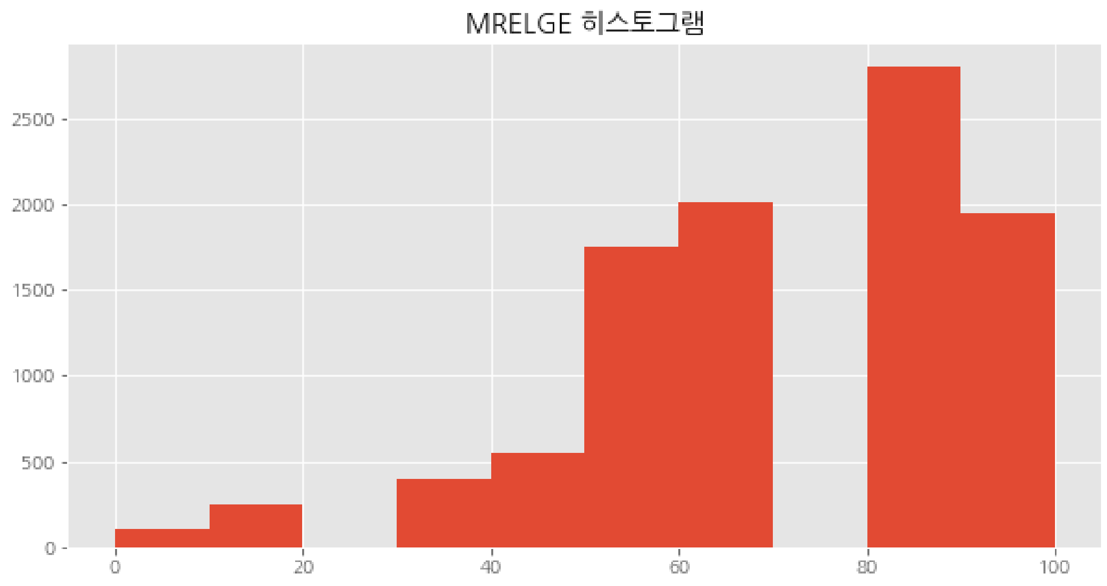
```
In [216]: MAUT1_table=pd.concat([raw_sample_0['MAUT1'].value_counts(),raw_sample_1['MAUT1'].val
          MAUT1_table.columns=[0,1]
          MAUT1_table=MAUT1_table.fillna(1)
          for i in range(MAUT1_table.shape[0]):
              print("MAUT1 ",MAUT1_table.index[i],"",MAUT1_table.sum(1).iloc[i]," ",MAUT1_tabl
```

```
MAUT1  0  31.0   0.03333333333333333
MAUT1  17  102.0   0.009900990099009901
MAUT1  30  400.0   0.03626943005181347
MAUT1  43  740.0   0.0335195530726257
MAUT1  56  2126.0   0.04987654320987654
MAUT1  69  2822.0   0.058514628657164294
MAUT1  82  2338.0   0.08896134140661388
MAUT1  95  435.0   0.06356968215158924
MAUT1  100  829.0   0.09656084656084656
```

0.0.18 MRELGE ()

```
In [217]: plt.figure(figsize=(10,5))
          plt.hist(raw_sample["MRELGE"])
          plt.title("MRELGE ")
```

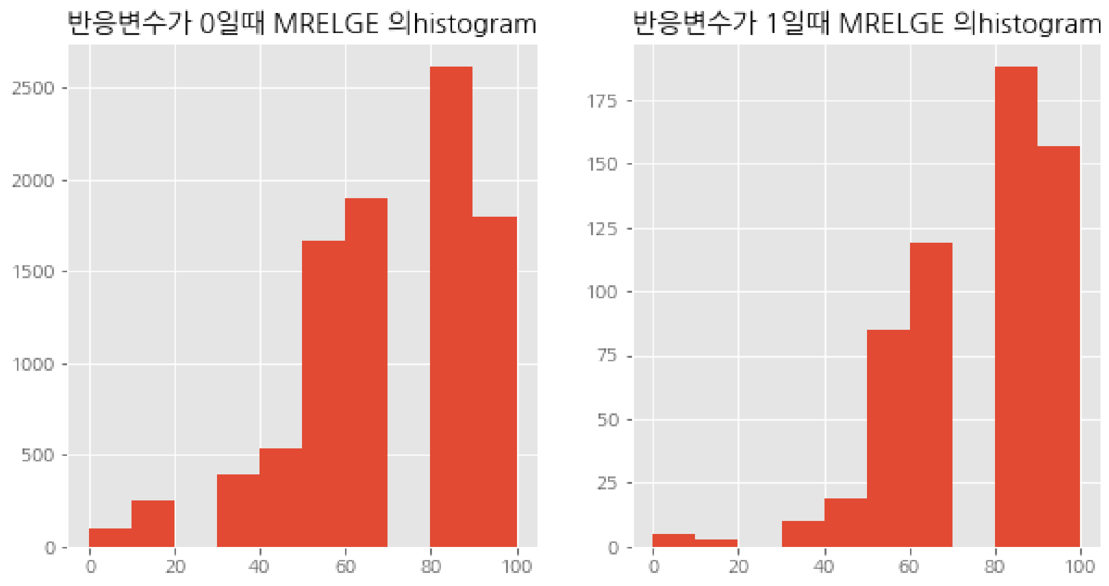
```
Out[217]: Text(0.5, 1.0, 'MRELGE ')
```



```
In [218]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(raw_sample_0["MRELGE"])
```

```
        axes1.set_title(" 0 MRELGE histogram")
        axes2=plt.subplot(1,2,2)
        axes2.hist(raw_sample_1["MRELGE"])
        axes2.set_title(" 1 MRELGE histogram")
```

```
Out[218]: Text(0.5, 1.0, ' 1 MRELGE histogram')
```



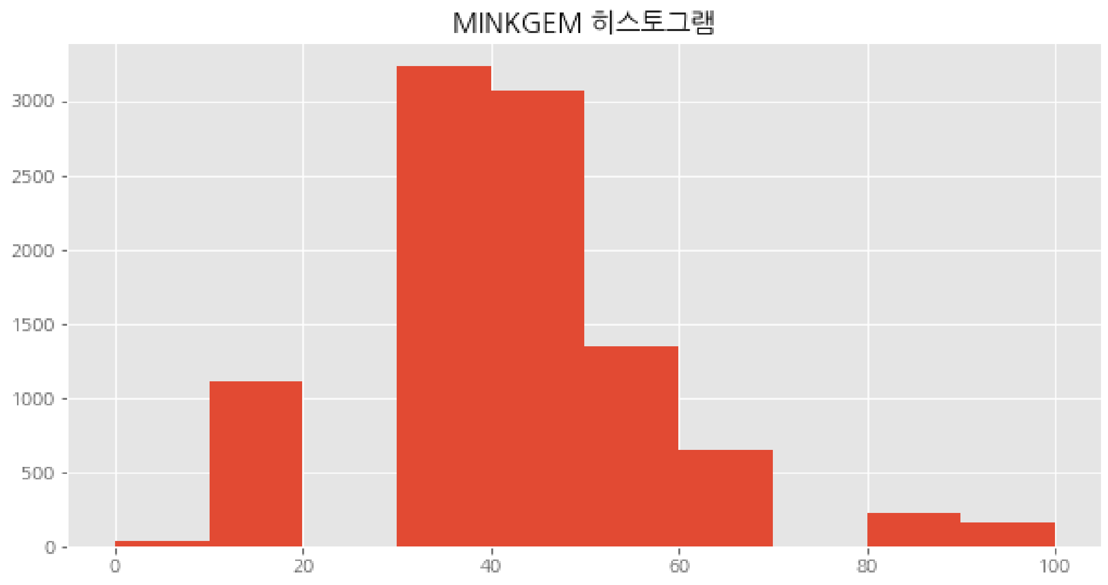
```
In [219]: MRELGE_table=pd.concat([raw_sample_0['MRELGE'].value_counts(),raw_sample_1['MRELGE']
          MRELGE_table.columns=[0,1]
          MRELGE_table=MRELGE_table.fillna(1)
          for i in range(MRELGE_table.shape[0]):
              print("MRELGE ",MRELGE_table.index[i],"",MRELGE_table.sum(1).iloc[i]," ",MRELGE_
```

```
MRELGE  0  108   0.04854368932038835
MRELGE  17  252   0.012048192771084338
MRELGE  30  402   0.025510204081632654
MRELGE  43  550   0.035781544256120526
MRELGE  56  1747   0.05114320096269555
MRELGE  69  2015   0.06276371308016877
MRELGE  82  2800   0.07197549770290965
MRELGE  95  603   0.08064516129032258
MRELGE  100  1345   0.0908353609083536
```

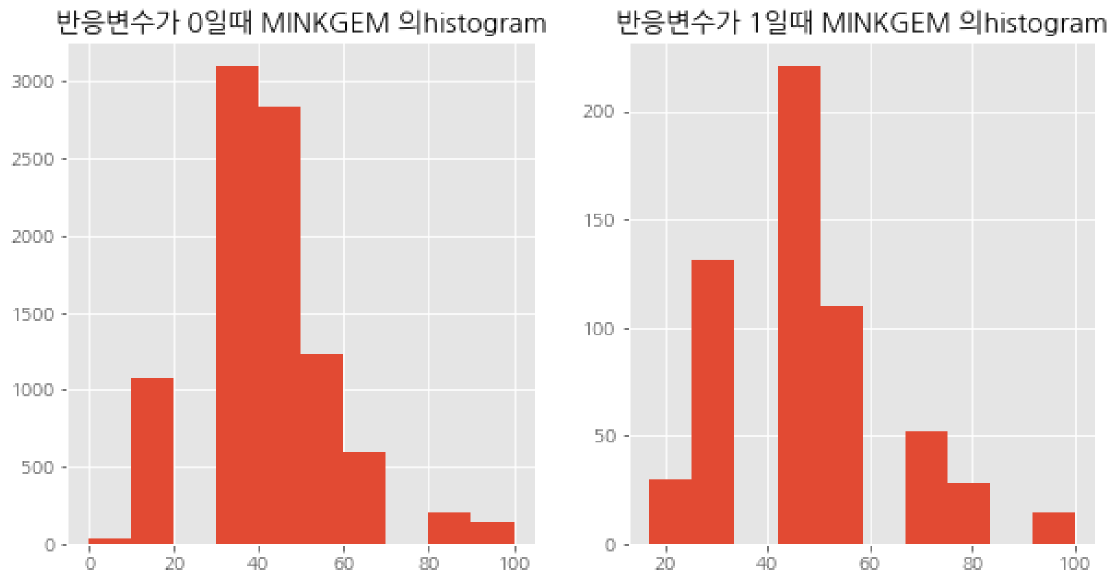
0.0.19 MINKGEM ()

```
In [220]: plt.figure(figsize=(10,5))
          plt.hist(raw_sample["MINKGEM"])
          plt.title("MINKGEM ")
```

```
Out[220]: Text(0.5, 1.0, 'MINKGEM ')
```



```
In [221]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(raw_sample_0["MINKGEM"])
          axes1.set_title(" 0 MINKGEM histogram")
          axes2=plt.subplot(1,2,2)
          axes2.hist(raw_sample_1["MINKGEM"])
          axes2.set_title(" 1 MINKGEM histogram")

Out[221]: Text(0.5, 1.0, ' 1 MINKGEM histogram')
```



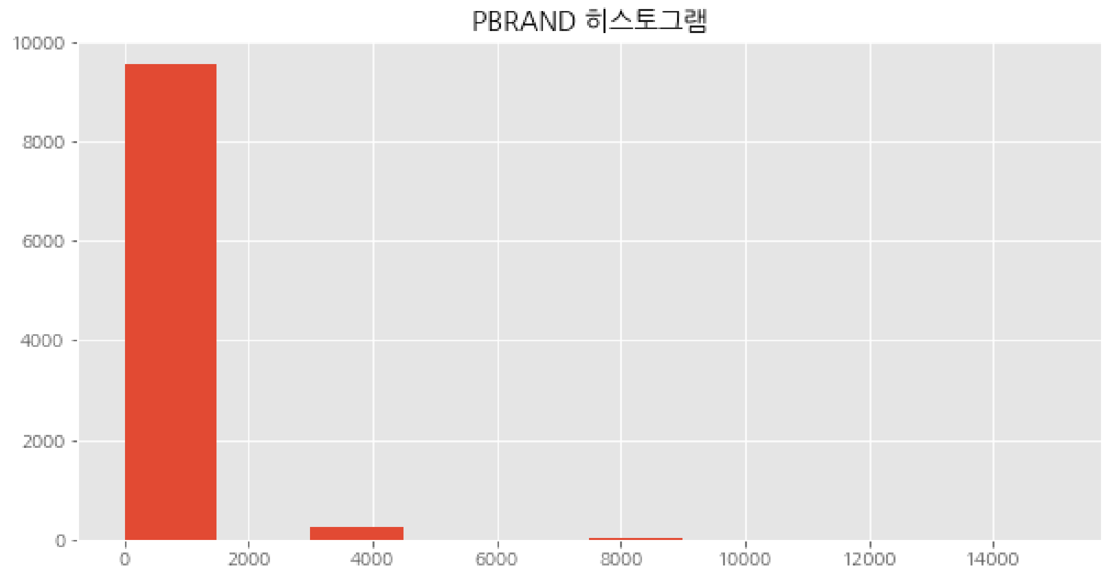
```
In [222]: MINKGEM_table=pd.concat([raw_sample_0['MINKGEM'].value_counts(),raw_sample_1['MINKGEM
          MINKGEM_table.columns=[0,1]
          MINKGEM_table=MINKGEM_table.fillna(1)
          for i in range(MINKGEM_table.shape[0]):
              print("MINKGEM ",MINKGEM_table.index[i],"",MINKGEM_table.sum(1).iloc[i]," ",MINKG
```

```
MINKGEM  0  39.0   0.02631578947368421
MINKGEM  17  1110.0   0.027777777777777776
MINKGEM  30  3232.0   0.042244437278297325
MINKGEM  43  3063.0   0.07776213933849402
MINKGEM  56  1346.0   0.0889967637540453
MINKGEM  69  646.0   0.08754208754208755
MINKGEM  82  228.0   0.14
MINKGEM  95  121.0   0.11009174311926606
MINKGEM  100  38.0   0.05555555555555555
```

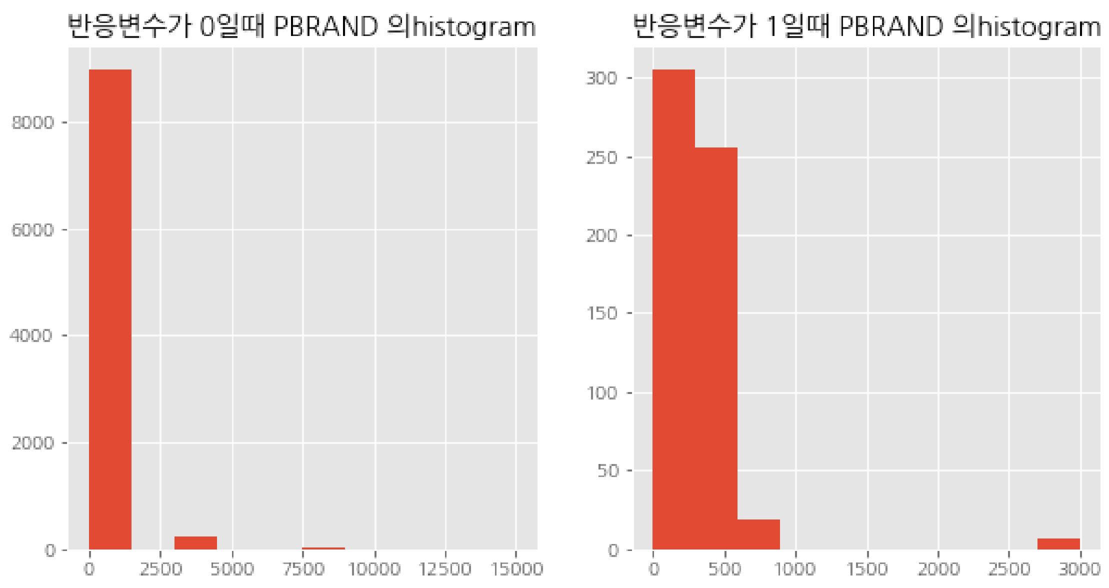
0.0.20 PBRAND ( )

```
In [223]: plt.figure(figsize=(10,5))
          plt.hist(raw_sample["PBRAND"])
          plt.title("PBRAND ")
```

```
Out[223]: Text(0.5, 1.0, 'PBRAND ')
```



```
In [224]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(raw_sample_0["PBRAND"])
          axes1.set_title(" 0 PBRAND histogram")
          axes2=plt.subplot(1,2,2)
          axes2.hist(raw_sample_1["PBRAND"])
          axes2.set_title(" 1 PBRAND histogram")

Out[224]: Text(0.5, 1.0, ' 1 PBRAND histogram')
```



```
In [225]: PBRAND_table=pd.concat([raw_sample_0['PBRAND'].value_counts(),raw_sample_1['PBRAND']
          PBRAND_table.columns=[0,1]
          PBRAND_table=PBRAND_table.fillna(1)
          for i in range(PBRAND_table.shape[0]):
              print("PBRAND ",PBRAND_table.index[i],"",PBRAND_table.sum(1).iloc[i]," ",PBRAND_

PBRAND  0  4464.0   0.042260098062106004
PBRAND  25  245.0   0.012396694214876033
PBRAND  75  901.0   0.01807909604519774
PBRAND  150  1541.0   0.07311977715877438
PBRAND  350  2142.0   0.13513513513513514
PBRAND  750  263.0   0.0778688524590164
PBRAND  3000  252.0   0.02857142857142857
PBRAND  7500  13.0   0.08333333333333333
PBRAND  15000  3.0   0.5
```

, , 3 , 3 , , , . (,)

0.0.21 (100) PCA

```
In [226]: allticdata=ticdata.copy()


          from sklearn.decomposition import PCA
          doPCA=True
          if doPCA:

              pca=PCA(n_components=2)
              pca.fit(allticdata.iloc[:,[15,16,17]])
              print("[[[Edu]]]")
              print("Edu    ")
              print(np.round(pca.explained_variance_ratio_,2))
              print("Edu   ")
              print(np.round(pca.components_,2)) ####  , ,##
              education=pca.transform(allticdata.iloc[:,[15,16,17]])
              education=pd.DataFrame(education)
              education.columns=["education1","education2"]

              ##### pca

              religion_pca=PCA(n_components=2)
              religion_pca.fit(allticdata.iloc[:,5:9])
              print("[[[religion]]]")
              print("religion   ")
              print(np.round(religion_pca.explained_variance_ratio_,2))#####    ###
              print("religion  ")
              print(np.round(religion_pca.components_,2))
              religion=religion_pca.transform(allticdata.iloc[:,[5,6,7,8]])
              religion=pd.DataFrame(religion)
              religion.columns=["religion1","religion2"]

              ####married 10~12

              married_pca=PCA(n_components=2)
              married_pca.fit(allticdata.iloc[:,9:12])
              print("[[[married]]]")
              print("married   ")
              print(np.round(married_pca.explained_variance_ratio_,2)) ###
              print("married  ")
              print(np.round(married_pca.components_,2))
              married=married_pca.transform(allticdata.iloc[:,[9,10,11]])
              married=pd.DataFrame(married)
              married.columns=["married1","married2"]

              ###single pca 13~15
```

```
single_pca=PCA(n_components=2)
single_pca.fit(allticdata.iloc[:,12:15])
print("[[[single]]]")
print("single   ")
print(np.round(single_pca.explained_variance_ratio_)) ###
print("single  ")
print(np.round(single_pca.components_,2))
single=single_pca.transform(allticdata.iloc[:,[12,13,14]])
single=pd.DataFrame(single)
single.columns=["single1","single2"]

######job pca(19~24)

job_pca=PCA(n_components=4)
job_pca.fit(allticdata.iloc[:,18:24])
print("job")
print(job_pca.explained_variance_ratio_)
print(job_pca.components_) ### ,      ,    ,
job=job_pca.transform(allticdata.iloc[:,18:24])

job=pd.DataFrame(job)

job.columns=["job_1","job_2","job_3","job_4"]

###### pca 25~29
###zip code

rank_pca=PCA(n_components=3)
rank_pca.fit(allticdata.iloc[:,24:29])
print("rank")
print(rank_pca.explained_variance_ratio_)
print(rank_pca.components_)  #CLASS C   , C  CLASS B   A
rank=rank_pca.transform(allticdata.iloc[:,24:29])
rank=pd.DataFrame(rank)
rank.columns=["rank_1","rank_2","rank_3"]

### RENT HOUSE PCA 30~31

rent_pca=PCA(n_components=1)
rent_pca.fit(allticdata.iloc[:,29:31])
print("[[[rent]]]")
print("rent   ")
print(np.round(rent_pca.explained_variance_ratio_,2))
print("rent  ")
print(np.round(rent_pca.components_,2))  ###renthouse house owener
rent=rent_pca.transform(allticdata.iloc[:,29:31])
rent=pd.DataFrame(rent)
rent.columns=["rent1"]
```

```
#### number of cars 32~34

cars_pca=PCA(n_components=2)
cars_pca.fit(allticdata.iloc[:,31:34])
print("[[[cars]]]")
print("car   ")
print(np.round(cars_pca.explained_variance_ratio_,2))
print("car  ")
print(np.round(cars_pca.components_,2))  ###   ,   0 1
cars=cars_pca.transform(allticdata.iloc[:,31:34])
cars=pd.DataFrame(cars)
cars.columns=["cars_1","cars_2"]

######   vs  pca 35~36

insurance_pca=PCA(n_components=1)
insurance_pca.fit(allticdata.iloc[:,34:36])
print("[[[insurance]]]")
print(np.round(insurance_pca.explained_variance_ratio_,2))
print(np.round(insurance_pca.components_,2)) ####   99%
insurance=insurance_pca.transform(allticdata.iloc[:,34:36])
insurance=pd.DataFrame(insurance)
insurance.columns=["insurance"]

##### pca 37~41
income_pca=PCA(n_components=3)
income_pca.fit(allticdata.iloc[:,36:41])
print("")
print(income_pca.explained_variance_ratio_)
print(income_pca.components_) ###    ,  30000  30000
income=income_pca.transform(allticdata.iloc[:,36:41])

income=pd.DataFrame(income)

income.columns=["income_1","income_2","income_3"]
###  pca     86-> 65

del_columns=allticdata.columns[5:41]

newticdata= allticdata.drop(del_columns, 1)
newticdata=pd.concat([newticdata,education,religion,married,single,rent,cars,insu

print(newticdata.columns)
print(newticdata.shape)
ticdata=newticdata
```

```
[[[Edu]]]
Edu
[0.58 0.31]
Edu
[[ 0.46  0.42 -0.78]
 [ 0.84 -0.48  0.25]]
[[[religion]]]
religion
[0.43 0.29]
religion
[[ 0.66 -0.43  0.55  0.27]
 [-0.3  -0.58 -0.41  0.63]]
[[[married]]]
married
[0.53 0.37]
married
[[-0.63  0.49  0.6 ]
 [-0.35 -0.87  0.34]]
[[[single]]]
single
[0. 0.]
single
[[ 0.58  0.3  -0.76]
 [ 0.73 -0.6   0.33]]
job
[0.22626969 0.19353952 0.17167808 0.16047774]
[[ 0.66092412  0.52557248  0.44536847 -0.18635891 -0.2307893   0.0246264 ]
 [-0.47031279  0.36894209  0.35651519 -0.46095187  0.54866256 -0.0455775 ]
 [-0.40884887  0.46918509  0.18445234  0.6262075  -0.28656277 -0.32315823]
 [-0.36814884  0.00079291  0.19306448 -0.13385575 -0.49142416  0.75350968]]
rank
[0.28558312 0.23535335 0.19953239]
[[ 0.79203099  0.19834751  0.11335954 -0.53169297  0.19441548]
 [-0.0776944   0.25697902  0.02815417  0.31840433  0.90870864]
 [-0.37514195  0.6838593   0.55473915 -0.24303018 -0.15749853]]
[[[rent]]]
rent
[0.92]
rent
[[ 0.72 -0.69]]
[[[cars]]]
car
[0.48 0.38]
car
[[-0.6   0.44  0.66]
 [ 0.11  0.87 -0.48]]
[[[insurance]]]
[0.87]
```

```
[[-0.71  0.7 ]]

[0.28429674 0.26660565 0.22538032]
[[-0.633707   -0.17907549  0.61952228  0.38626269  0.18259434]
 [ 0.62602838 -0.65713548  0.19937055  0.3020715   0.21275718]
 [-0.00788117  0.28860809 -0.48885668  0.77862785  0.26720978]]
Index(['MOSTYPE', 'MAANTHUI', 'MGEMOMV', 'MGEMLEEF', 'MOSHOOFD', 'MINKGEM',
       'MKOOPKLA', 'PWAPART', 'PWABEDR', 'PWALAND', 'PPERSAUT', 'PBESAUT',
       'PMOTSCO', 'PVRAAUT', 'PAANHANG', 'PTRACTOR', 'PWERKT', 'PBROM',
       'PLEVEN', 'PPERSONG', 'PGEZONG', 'PWAOREG', 'PBRAND', 'PZEILPL',
       'PPLEZIER', 'PFIETS', 'PINBOED', 'PBYSTAND', 'AWAPART', 'AWABEDR',
       'AWALAND', 'APERSAUT', 'ABESAUT', 'AMOTSCO', 'AVRAAUT', 'AAANHANG',
       'ATRACTOR', 'AWERKT', 'ABROM', 'ALEVEN', 'APERSONG', 'AGEZONG',
       'AWAOREG', 'ABRAND', 'AZEILPL', 'APLEZIER', 'AFIETS', 'AINBOED',
       'ABYSTAND', 'CARAVAN', 'education1', 'education2', 'religion1',
       'religion2', 'married1', 'married2', 'single1', 'single2', 'rent1',
       'cars_1', 'cars_2', 'insurance', 'income_1', 'income_2', 'income_3',
       'job_1', 'job_2', 'job_3', 'job_4', 'rank_1', 'rank_2', 'rank_3'],
      dtype='object')
(9822, 72)
```

0.0.22

```
In [227]: a=list(ticdata.columns)
          a.remove('CARAVAN')
          a.remove('MOSTYPE')
          a.remove('MOSHOOFD')
```

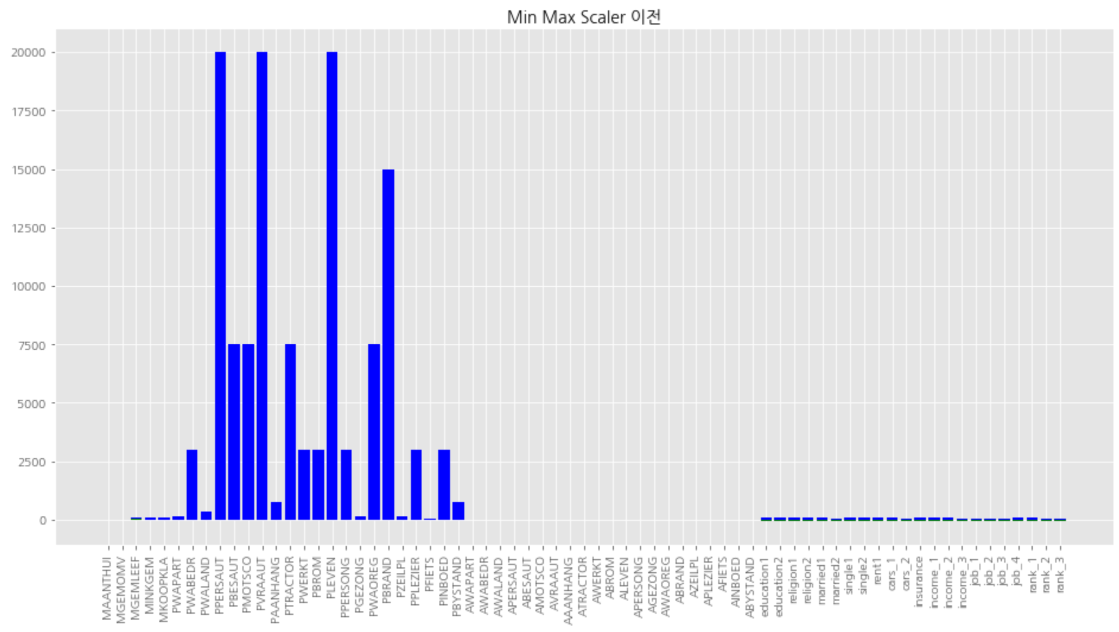
```
In [228]: ticdata_continuous=ticdata[a]
```

```
In [229]: stat=ticdata.describe()
          print(stat)
```

```
          MAANTHUI      MGEMOMV     MGEMLEEF      MINKGEM     MKOOPKLA  \
count  9822.000000  9822.000000  9822.000000  9822.000000  9822.000000
mean      1.108735     2.677561    44.964366    40.875585    51.350336
std       0.412101     0.780701     8.046598    16.837012    22.087471
min       1.000000     1.000000    25.000000     0.000000    17.000000
25%       1.000000     2.000000    35.000000    30.000000    30.000000
50%       1.000000     3.000000    45.000000    43.000000    56.000000
75%       1.000000     3.000000    45.000000    43.000000    69.000000
max      10.000000     6.000000    75.000000   100.000000    95.000000

           PWAPART      PWABEDR      PWALAND     PPERSAUT      PBESAUT  ...   \
count  9822.000000  9822.000000  9822.000000   9822.000000  9822.000000  ...
mean     28.311444     3.812869     5.312055   1330.294237    26.038485  ...
std      36.012237    72.743143    39.163783   1546.992525   297.103273  ...
min       0.000000     0.000000     0.000000      0.000000     0.000000  ...
```

```
25%       0.000000     0.000000     0.000000      0.000000     0.000000  ...
50%       0.000000     0.000000     0.000000    750.000000     0.000000  ...
75%      75.000000     0.000000     0.000000   3000.000000     0.000000  ...
max     150.000000  3000.000000   350.000000  20000.000000  7500.000000  ...

           income_1      income_2      income_3         job_1         job_2  \
count  9.822000e+03  9.822000e+03  9.822000e+03  9.822000e+03  9.822000e+03
mean  -1.595502e-15 -4.760101e-16  3.045597e-16 -1.663865e-16 -5.988106e-16
std    2.684151e+01  2.599295e+01  2.389896e+01  2.605783e+01  2.409960e+01
min   -7.236192e+01 -7.366003e+01 -6.003931e+01 -5.046348e+01 -6.791229e+01
25%   -1.817819e+01 -1.780362e+01 -1.645377e+01 -2.218664e+01 -1.800779e+01
50%    2.950805e+00  2.115539e+00 -2.433784e+00 -5.660454e-01 -7.640535e-01
75%    1.641644e+01  1.754246e+01  1.770717e+01  1.890935e+01  1.888679e+01
max    7.948196e+01  7.960697e+01  6.670915e+01  7.273837e+01  6.886258e+01

              job_3         job_4        rank_1        rank_2        rank_3
count  9.822000e+03  9.822000e+03  9.822000e+03  9.822000e+03  9.822000e+03
mean  -1.069214e-15 -6.971956e-16  1.115151e-15  6.727802e-17 -1.995553e-15
std    2.269773e+01  2.194484e+01  2.721778e+01  2.470851e+01  2.275061e+01
min   -5.557963e+01 -6.601747e+01 -6.656006e+01 -4.778245e+01 -5.907896e+01
25%   -1.524578e+01 -1.466223e+01 -1.863463e+01 -2.097174e+01 -1.813365e+01
50%   -2.348593e+00  1.123479e+00  1.132804e+00 -5.227916e+00  4.133134e-01
75%    1.375432e+01  1.460771e+01  1.981489e+01  2.381186e+01  1.651443e+01
max    6.847663e+01  7.932332e+01  9.019504e+01  6.829920e+01  7.317956e+01

[8 rows x 70 columns]
```

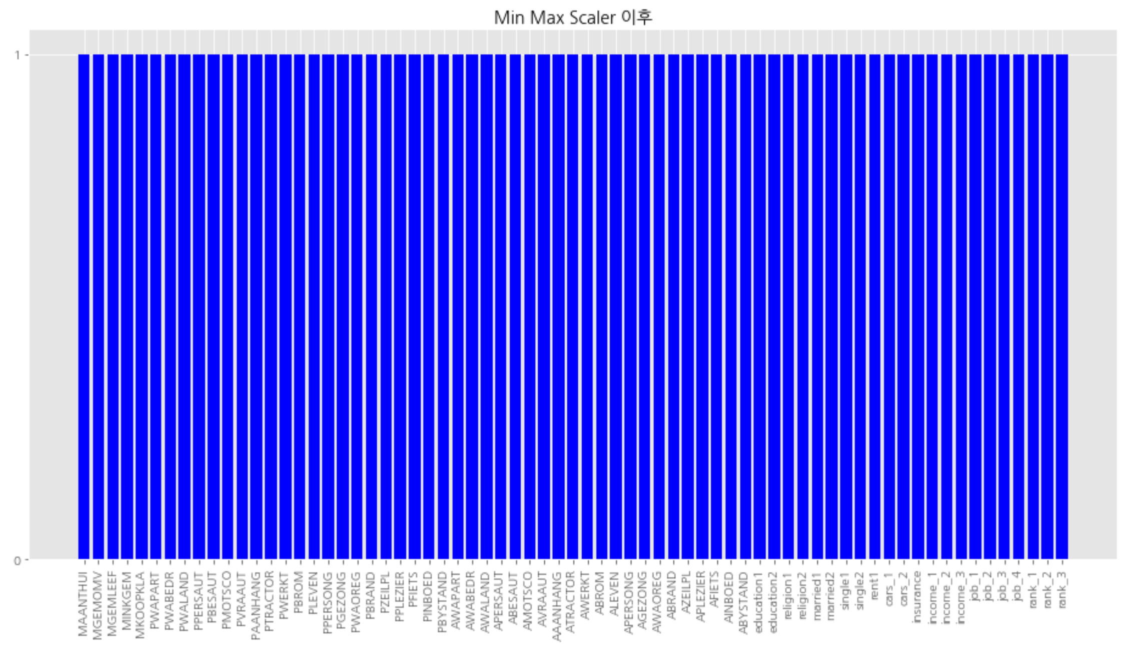
```
In [230]: fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = ticdata[a].columns
          x_pos = np.arange(len(objects))
          ax1 = plt.bar(x_pos, stat[a].loc['max'],color="blue" ,alpha=1)
          ax1 = plt.bar(x_pos, stat[a].loc['min'],color="green" ,alpha=1)
          plt.xticks(x_pos, objects)
          plt.xticks(rotation=90);
          plt.title('Min Max Scaler ', size=14)

Out[230]: Text(0.5, 1.0, 'Min Max Scaler ')
```



```
In [231]: scaler = preprocessing.MinMaxScaler()
          scaler.fit(ticdata_continuous)
          ticdata[a]=scaler.transform(ticdata_continuous)

C:\Users\jang\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:334: DataConversionWarn
  return self.partial_fit(X, y)


In [232]: stat=ticdata.describe()
          print(stat)

           MAANTHUI      MGEMOMV      MGEMLEEF      MINKGEM      MKOOPKLA  \
count   9822.000000  9822.000000  9822.000000  9822.000000  9822.000000
mean       0.012082     0.335512     0.399287     0.408756     0.440389
std        0.045789     0.156140     0.160932     0.168370     0.283173
min        0.000000     0.000000     0.000000     0.000000     0.000000
25%        0.000000     0.200000     0.200000     0.300000     0.166667
50%        0.000000     0.400000     0.400000     0.430000     0.500000
75%        0.000000     0.400000     0.400000     0.430000     0.666667
max        1.000000     1.000000     1.000000     1.000000     1.000000

            PWAPART      PWABEDR       PWALAND     PPERSAUT      PBESAUT  ...  \
count   9822.000000  9822.000000  9822.000000  9822.000000  9822.000000  ...
mean       0.188743     0.001271     0.015177     0.066515     0.003472  ...
std        0.240082     0.024248     0.111897     0.077350     0.039614  ...
min        0.000000     0.000000     0.000000     0.000000     0.000000  ...
25%        0.000000     0.000000     0.000000     0.000000     0.000000  ...
```

```
50%         0.000000     0.000000     0.000000     0.037500     0.000000  ...
75%         0.500000     0.000000     0.000000     0.150000     0.000000  ...
max         1.000000     1.000000     1.000000     1.000000     1.000000  ...

            income_1     income_2     income_3        job_1        job_2  \
count    9822.000000  9822.000000  9822.000000  9822.000000  9822.000000
mean        0.476555     0.480599     0.473689     0.409600     0.496526
std         0.176770     0.169593     0.188554     0.211505     0.176199
min         0.000000     0.000000     0.000000     0.000000     0.000000
25%         0.356838     0.364439     0.343874     0.229516     0.364866
50%         0.495988     0.494402     0.454487     0.405006     0.490940
75%         0.584669     0.595056     0.613392     0.563083     0.634613
max         1.000000     1.000000     1.000000     1.000000     1.000000

               job_3        job_4       rank_1       rank_2       rank_3
count    9822.000000  9822.000000  9822.000000  9822.000000  9822.000000
mean        0.448020     0.454225     0.424612     0.411628     0.446693
std         0.182963     0.150989     0.173632     0.212855     0.172016
min         0.000000     0.000000     0.000000     0.000000     0.000000
25%         0.325125     0.353344     0.305734     0.230964     0.309585
50%         0.429088     0.461955     0.431838     0.366591     0.449818
75%         0.558891     0.554732     0.551018     0.616758     0.571558
max         1.000000     1.000000     1.000000     1.000000     1.000000

[8 rows x 70 columns]
```

```
In [233]: fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = ticdata[a].columns
          x_pos = np.arange(len(objects))
          ax1 = plt.bar(x_pos, stat[a].loc['max'],color="blue" ,alpha=1)
          ax1 = plt.bar(x_pos, stat[a].loc['min'],color="green" ,alpha=1)
          plt.xticks(x_pos, objects)
          plt.xticks(rotation=90);
          plt.yticks(range(2))
          plt.title('Min Max Scaler ', size=14)

Out[233]: Text(0.5, 1.0, 'Min Max Scaler ')
```



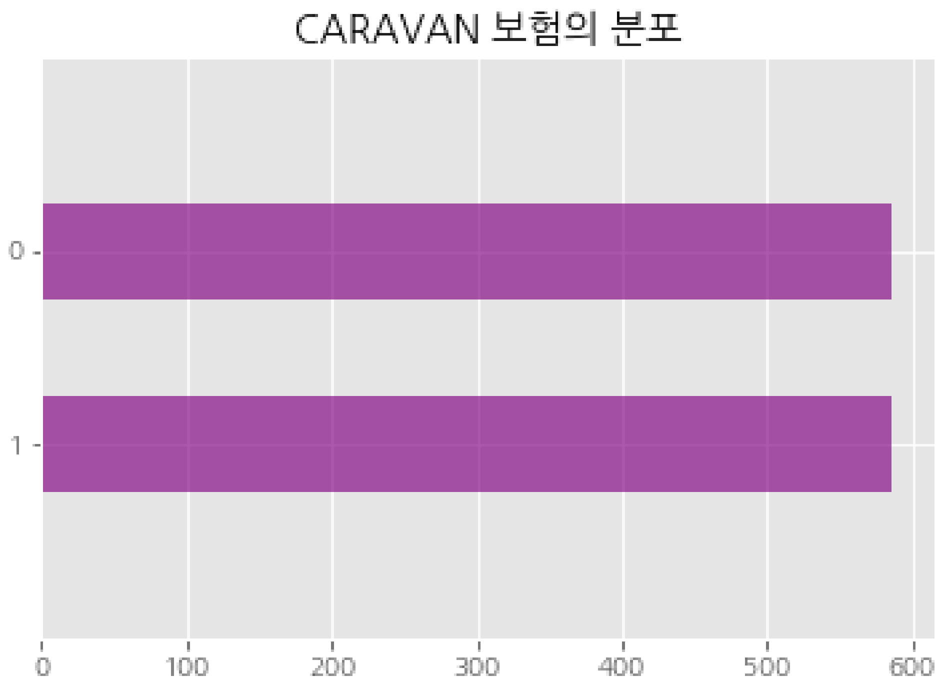
0.1

```
In [234]: ticdata['CARAVAN'].value_counts() #### 0 1          1

Out[234]: 0    9236
          1     586
          Name: CARAVAN, dtype: int64
```

0.1.1 & sample weight

```
In [235]: ticdata_1=ticdata.loc[ticdata['CARAVAN']==1]### 1
          ticdata_0=ticdata.loc[ticdata['CARAVAN']==0]### 0  .
          ticdata_0_sample=ticdata_0.sample(586,random_state=13) ####9236  0  586  .
          ticdata=pd.concat([ticdata_0_sample,ticdata_1],axis=0) ### 0:586 1:586
          ticdata=ticdata.sort_index() ### index
          ticdata.shape

Out[235]: (1172, 72)

In [236]: fig,ax= plt.subplots()
          ticdata.CARAVAN.value_counts().plot(kind='barh', color="purple", alpha=.65)
          ax.set_ylim(-1, len(ticdata.CARAVAN.value_counts()))
          plt.title("CARAVAN  ")
          print("CARAVAN  ")
          print(ticdata.CARAVAN.value_counts())
```

```
CARAVAN
1    586
0    586
Name: CARAVAN, dtype: int64
```



```
In [237]: sampleweight_0=allticdata['CARAVAN'].value_counts()[0]/sum(ticdata.CARAVAN==0)
          sampleweight_1=allticdata['CARAVAN'].value_counts()[1]/sum(ticdata.CARAVAN==1)
          print(sampleweight_0) ### 0 sample weight
          print(sampleweight_1) ### 1 sample weight
```

```
15.761092150170649
1.0
```

```
In [238]: weight_dict={0:sampleweight_0,1:sampleweight_1} ##sample weight dictionary
```

0.1.2 dummy variable

Mostype MOSHOOFD . MOSTYPE . MOSTYPE MOSHOOFD

```
In [239]: Dummies=True
          if Dummies:
              ##Dummy_MOSTYPE=pd.get_dummies(ticdata['MOSTYPE'],prefix="MOSTYPE")
              Dummy_MOSHOOFD=pd.get_dummies(ticdata['MOSHOOFD'],prefix="MOSHOOFD")
              ticdata=pd.concat([ticdata,Dummy_MOSHOOFD],1)
```

```
        ##ticdata=pd.concat([ticdata,Dummy_MOSTYPE],1)
        ticdata=ticdata.drop(['MOSTYPE'],1)
        ticdata=ticdata.drop(['MOSHOOFD'],1)
        print(ticdata.shape)
```

```
(1172, 80)
```

```
In [240]: ticdata.columns
```

```
Out[240]: Index(['MAANTHUI', 'MGEMOMV', 'MGEMLEEF', 'MINKGEM', 'MKOOPKLA', 'PWAPART',
                 'PWABEDR', 'PWALAND', 'PPERSAUT', 'PBESAUT', 'PMOTSCO', 'PVRAAUT',
                 'PAANHANG', 'PTRACTOR', 'PWERKT', 'PBROM', 'PLEVEN', 'PPERSONG',
                 'PGEZONG', 'PWAOREG', 'PBRAND', 'PZEILPL', 'PPLEZIER', 'PFIETS',
                 'PINBOED', 'PBYSTAND', 'AWAPART', 'AWABEDR', 'AWALAND', 'APERSAUT',
                 'ABESAUT', 'AMOTSCO', 'AVRAAUT', 'AAANHANG', 'ATRACTOR', 'AWERKT',
                 'ABROM', 'ALEVEN', 'APERSONG', 'AGEZONG', 'AWAOREG', 'ABRAND',
                 'AZEILPL', 'APLEZIER', 'AFIETS', 'AINBOED', 'ABYSTAND', 'CARAVAN',
                 'education1', 'education2', 'religion1', 'religion2', 'married1',
                 'married2', 'single1', 'single2', 'rent1', 'cars_1', 'cars_2',
                 'insurance', 'income_1', 'income_2', 'income_3', 'job_1', 'job_2',
                 'job_3', 'job_4', 'rank_1', 'rank_2', 'rank_3', 'MOSHOOFD_1',
                 'MOSHOOFD_2', 'MOSHOOFD_3', 'MOSHOOFD_4', 'MOSHOOFD_5', 'MOSHOOFD_6',
                 'MOSHOOFD_7', 'MOSHOOFD_8', 'MOSHOOFD_9', 'MOSHOOFD_10'],
                dtype='object')
```

1 1.

```
In [241]: numerical_columns=[name for name in list(ticdata.columns) if name not in ['CARAVAN']]

          correlation_df=ticdata[numerical_columns].corr()
          correlated_pairs=list(correlation_df[abs(correlation_df)>0.7].stack().index)
          correlated_pairs=[ pair for pair in correlated_pairs if (pair[0]!=pair[1])]
          print(" 0.7  ",np.int(len(set(correlated_pairs))/2)," ")
```

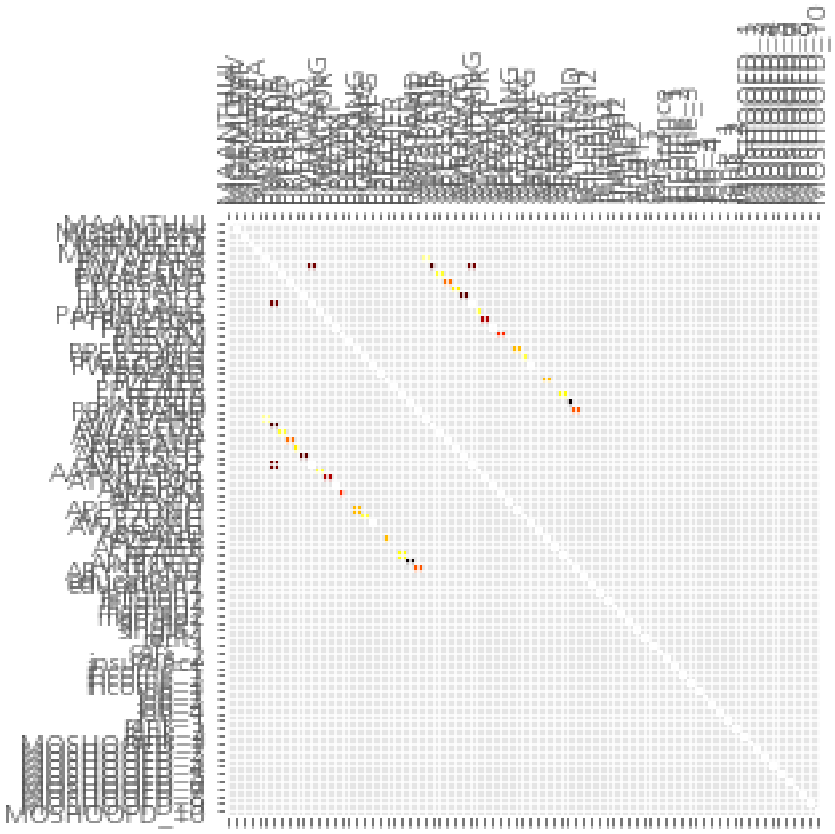
```
 0.7   21
```

```
In [242]: correlated_pairs
```

```
Out[242]: [('MGEMOMV', 'single1'),
           ('PWAPART', 'AWAPART'),
           ('PWABEDR', 'PVRAAUT'),
           ('PWABEDR', 'AWABEDR'),
           ('PWABEDR', 'AVRAAUT'),
           ('PWALAND', 'AWALAND'),
           ('PPERSAUT', 'APERSAUT'),
           ('PBESAUT', 'ABESAUT'),
           ('PMOTSCO', 'AMOTSCO'),
           ('PVRAAUT', 'PWABEDR'),
           ('PVRAAUT', 'AVRAAUT'),
           ('PAANHANG', 'AAANHANG'),
           ('PTRACTOR', 'ATRACTOR'),
           ('PWERKT', 'AWERKT'),
           ('PBROM', 'ABROM'),
           ('PPERSONG', 'APERSONG'),
           ('PGEZONG', 'AGEZONG'),
           ('PWAOREG', 'AWAOREG'),
           ('PZEILPL', 'AZEILPL'),
           ('PFIETS', 'AFIETS'),
           ('PINBOED', 'AINBOED'),
           ('PBYSTAND', 'ABYSTAND'),
           ('AWAPART', 'PWAPART'),
           ('AWABEDR', 'PWABEDR'),
           ('AWALAND', 'PWALAND'),
           ('APERSAUT', 'PPERSAUT'),
           ('ABESAUT', 'PBESAUT'),
           ('AMOTSCO', 'PMOTSCO'),
           ('AVRAAUT', 'PWABEDR'),
           ('AVRAAUT', 'PVRAAUT'),
           ('AAANHANG', 'PAANHANG'),
           ('ATRACTOR', 'PTRACTOR'),
           ('AWERKT', 'PWERKT'),
           ('ABROM', 'PBROM'),
           ('APERSONG', 'PPERSONG'),
           ('AGEZONG', 'PGEZONG'),
           ('AWAOREG', 'PWAOREG'),
           ('AZEILPL', 'PZEILPL'),
           ('AFIETS', 'PFIETS'),
           ('AINBOED', 'PINBOED'),
           ('ABYSTAND', 'PBYSTAND'),
           ('single1', 'MGEMOMV')]
```

```
In [243]: plt.figure(figsize=(100,100))
          plt.matshow(ticdata[numerical_columns].corr()[ticdata[numerical_columns].corr()>0.7]
          plt.xticks(range(len(ticdata[numerical_columns].columns)),ticdata[numerical_columns]
          plt.subplots_adjust(bottom=0.15)
          plt.yticks(range(len(ticdata[numerical_columns].columns)),ticdata[numerical_columns]
```

```
Out[243]: ([<matplotlib.axis.YTick at 0x1d8675f3898>,
            <matplotlib.axis.YTick at 0x1d8675f31d0>,
            <matplotlib.axis.YTick at 0x1d8675ff320>,
            <matplotlib.axis.YTick at 0x1d8685b4a20>,
            <matplotlib.axis.YTick at 0x1d8676e86a0>,
            <matplotlib.axis.YTick at 0x1d8685c3a90>,
            <matplotlib.axis.YTick at 0x1d8685ca128>,
            <matplotlib.axis.YTick at 0x1d8685ca5c0>,
            <matplotlib.axis.YTick at 0x1d8685cab00>,
            <matplotlib.axis.YTick at 0x1d8685d30f0>,
            <matplotlib.axis.YTick at 0x1d8685d35c0>,
            <matplotlib.axis.YTick at 0x1d8685d3b00>,
            <matplotlib.axis.YTick at 0x1d8685ca908>,
            <matplotlib.axis.YTick at 0x1d8685b46a0>,
            <matplotlib.axis.YTick at 0x1d8685d3ef0>,
            <matplotlib.axis.YTick at 0x1d8685dc470>,
            <matplotlib.axis.YTick at 0x1d8685dc9b0>,
            <matplotlib.axis.YTick at 0x1d8685dcef0>,
            <matplotlib.axis.YTick at 0x1d8685e3470>,
            <matplotlib.axis.YTick at 0x1d8685e39b0>,
            <matplotlib.axis.YTick at 0x1d8685e3ef0>,
            <matplotlib.axis.YTick at 0x1d8685e3828>,
            <matplotlib.axis.YTick at 0x1d8685dc908>,
            <matplotlib.axis.YTick at 0x1d8685b44a8>,
            <matplotlib.axis.YTick at 0x1d8685eb8d0>,
            <matplotlib.axis.YTick at 0x1d8685ebe10>,
            <matplotlib.axis.YTick at 0x1d8685f3390>,
            <matplotlib.axis.YTick at 0x1d8685f38d0>,
            <matplotlib.axis.YTick at 0x1d8685f3e10>,
            <matplotlib.axis.YTick at 0x1d8685fc390>,
            <matplotlib.axis.YTick at 0x1d8685fc8d0>,
            <matplotlib.axis.YTick at 0x1d8685f3828>,
            <matplotlib.axis.YTick at 0x1d8685adba8>,
            <matplotlib.axis.YTick at 0x1d8685fccc0>,
            <matplotlib.axis.YTick at 0x1d868603240>,
            <matplotlib.axis.YTick at 0x1d868603780>,
            <matplotlib.axis.YTick at 0x1d868603cc0>,
            <matplotlib.axis.YTick at 0x1d86860b240>,
            <matplotlib.axis.YTick at 0x1d86860b780>,
            <matplotlib.axis.YTick at 0x1d86860bcc0>,
            <matplotlib.axis.YTick at 0x1d86860b6d8>,
            <matplotlib.axis.YTick at 0x1d868603828>,
            <matplotlib.axis.YTick at 0x1d8685eb6d8>,
            <matplotlib.axis.YTick at 0x1d8686136a0>,
            <matplotlib.axis.YTick at 0x1d868613be0>,
            <matplotlib.axis.YTick at 0x1d86861c198>,
            <matplotlib.axis.YTick at 0x1d86861c6a0>,
            <matplotlib.axis.YTick at 0x1d86861cbe0>,
            <matplotlib.axis.YTick at 0x1d868622198>,
            <matplotlib.axis.YTick at 0x1d8686226a0>,
            <matplotlib.axis.YTick at 0x1d86861c748>,
            <matplotlib.axis.YTick at 0x1d8685eb828>,
            <matplotlib.axis.YTick at 0x1d868622a90>,
            <matplotlib.axis.YTick at 0x1d868622be0>,
            <matplotlib.axis.YTick at 0x1d86862c550>,
            <matplotlib.axis.YTick at 0x1d86862ca90>,
            <matplotlib.axis.YTick at 0x1d86862cbe0>,
            <matplotlib.axis.YTick at 0x1d868632550>,
            <matplotlib.axis.YTick at 0x1d868632a90>,
            <matplotlib.axis.YTick at 0x1d868632be0>,
            <matplotlib.axis.YTick at 0x1d86862ca58>,
            <matplotlib.axis.YTick at 0x1d867631358>,
            <matplotlib.axis.YTick at 0x1d86863b470>,
            <matplotlib.axis.YTick at 0x1d86863b9b0>,
            <matplotlib.axis.YTick at 0x1d86863bef0>,
            <matplotlib.axis.YTick at 0x1d868643470>,
            <matplotlib.axis.YTick at 0x1d8686439b0>,
            <matplotlib.axis.YTick at 0x1d868643ef0>,
            <matplotlib.axis.YTick at 0x1d8686437b8>,
            <matplotlib.axis.YTick at 0x1d86863ba90>,
            <matplotlib.axis.YTick at 0x1d86864b390>,
            <matplotlib.axis.YTick at 0x1d86864b9b0>,
            <matplotlib.axis.YTick at 0x1d86864bef0>,
            <matplotlib.axis.YTick at 0x1d868652470>,
            <matplotlib.axis.YTick at 0x1d8686529b0>,
            <matplotlib.axis.YTick at 0x1d868652ef0>,
            <matplotlib.axis.YTick at 0x1d86865a470>,
            <matplotlib.axis.YTick at 0x1d86865a9b0>,
            <matplotlib.axis.YTick at 0x1d868652908>],
           <a list of 79 Text yticklabel objects>)

<Figure size 7200x7200 with 0 Axes>
```



Random Forest

0.005%   max_depth=5   (   )

```
In [244]: from sklearn.ensemble import RandomForestClassifier
          clf= RandomForestClassifier(max_depth=1,random_state=0,oob_score=True,n_estimators=1
          clf.fit(ticdata.drop('CARAVAN',1),ticdata['CARAVAN'])
          print(clf.oob_score_)
          variable_importances=list(zip(ticdata.drop('CARAVAN',1).columns,np.round(clf.feature_
          feature_importance_matrix=pd.DataFrame(variable_importances).sort_values(by=1)
          print(feature_importance_matrix)
```

```
0.7064846416382252
              0     1
0      MAANTHUI  0.00
32      AVRAAUT  0.00
33     AAANHANG  0.00
34     ATRACTOR  0.00
35       AWERKT  0.00
37       ALEVEN  0.00
```

```
38     APERSONG  0.00
77   MOSHOOFD_9  0.00
40       AWAOREG  0.00
42       AZEILPL  0.00
43     APLEZIER  0.00
44        AFIETS  0.00
45       AINBOED  0.00
46     ABYSTAND  0.00
48   education2  0.00
49    religion1  0.00
50    religion2  0.00
54       single2  0.00
61      income_3  0.00
62         job_1  0.00
65         job_4  0.00
68        rank_3  0.00
69   MOSHOOFD_1  0.00
71   MOSHOOFD_3  0.00
72   MOSHOOFD_4  0.00
74   MOSHOOFD_6  0.00
75   MOSHOOFD_7  0.00
76   MOSHOOFD_8  0.00
31       AMOTSCO  0.00
30       ABESAUT  0.00
..           ...   ...
27       AWABEDR  0.00
21       PZEILPL  0.00
15         PBROM  0.01
1        MGEMOMV  0.01
66        rank_1  0.01
64         job_3  0.01
57        cars_2  0.01
58     insurance  0.01
60      income_2  0.01
36         ABROM  0.01
73   MOSHOOFD_5  0.02
70   MOSHOOFD_2  0.02
67        rank_2  0.02
63         job_2  0.02
41        ABRAND  0.02
53       single1  0.02
52      married2  0.02
78  MOSHOOFD_10  0.02
55         rent1  0.03
51      married1  0.03
3        MINKGEM  0.04
47   education1  0.04
59      income_1  0.04
```

```
56       cars_1  0.05
26      AWAPART  0.06
4       MKOOPKLA  0.06
29      APERSAUT  0.08
5        PWAPART  0.08
20        PBRAND  0.09
8       PPERSAUT  0.12

[79 rows x 2 columns]
```

1.0.1 0.01

```
In [245]: selected_variables=feature_importance_matrix[feature_importance_matrix.iloc[:,1]>=0.0
```

```
In [246]: selected_variables.shape ####35
```

```
Out[246]: (28,)
```

```
In [247]: selected_variables
```

```
Out[247]: 15           PBROM
          1          MGEMOMV
          66          rank_1
          64           job_3
          57          cars_2
          58       insurance
          60        income_2
          36           ABROM
          73      MOSHOOFD_5
          70      MOSHOOFD_2
          67          rank_2
          63           job_2
          41          ABRAND
          53         single1
          52        married2
          78     MOSHOOFD_10
          55           rent1
          51        married1
          3          MINKGEM
          47      education1
          59        income_1
          56          cars_1
          26         AWAPART
          4         MKOOPKLA
          29        APERSAUT
          5          PWAPART
          20          PBRAND
          8         PPERSAUT
          Name: 0, dtype: object
```

1.0.2

```
In [248]: from statsmodels.stats.outliers_influence import variance_inflation_factor
          vifdata=ticdata[selected_variables].copy()

In [249]: vif = pd.DataFrame()
          vif["VIF Factor"] = [variance_inflation_factor(vifdata.values, i) for i in range(vifd
          vif["features"] = vifdata.columns

In [250]: vif

Out[250]:     VIF Factor     features
          0     3.434133        PBROM
          1     9.934478      MGEMOMV
          2    13.440081       rank_1
          3     7.497014        job_3
          4    10.679463       cars_2
          5     5.561930    insurance
          6    11.334685     income_2
          7     3.464604        ABROM
          8     1.439118   MOSHOOFD_5
          9     1.655879   MOSHOOFD_2
          10    6.159043       rank_2
          11    9.860732        job_2
          12    4.513544       ABRAND
          13   12.812552      single1
          14    5.304505     married2
          15    1.196319  MOSHOOFD_10
          16    4.043941        rent1
          17    6.268416     married1
          18   15.395973      MINKGEM
          19   13.494484   education1
          20   17.019067     income_1
          21    6.019436       cars_1
          22   33.800844      AWAPART
          23    8.294747     MKOOPKLA
          24    8.776984     APERSAUT
          25   32.219449      PWAPART
          26    1.651168       PBRAND
          27    8.831165     PPERSAUT

In [251]: delcolumns={}
          while True:
              vif = pd.DataFrame()
              vif["VIF Factor"] = [variance_inflation_factor(vifdata.values, i) for i in range
              vif["features"] = vifdata.columns
              if vif.max()[0]>10:
                  vifdata=vifdata.drop(vif[vif['VIF Factor']==vif['VIF Factor'].max()].features
                  delcolumns[vif[vif['VIF Factor']==vif['VIF Factor'].max()].features.values[0]
```

```
            else:
                break
```

```
In [252]: delcolumns ###
```

```
Out[252]: {'AWAPART': 33.80084356848073,
           'income_1': 16.953913437498223,
           'education1': 13.334357596924699,
           'single1': 12.44758382731473,
           'MINKGEM': 12.254281825492022,
           'income_2': 10.800775830087982,
           'cars_2': 10.472816215970791}
```

```
In [253]: vif ### vif
```

```
Out[253]:     VIF Factor     features
          0     3.411410        PBROM
          1     6.671478      MGEMOMV
          2     8.909051       rank_1
          3     6.247957        job_3
          4     4.847936    insurance
          5     3.454182        ABROM
          6     1.384133   MOSHOOFD_5
          7     1.613451   MOSHOOFD_2
          8     5.545659       rank_2
          9     8.751763        job_2
          10    4.335823       ABRAND
          11    4.452644     married2
          12    1.170118  MOSHOOFD_10
          13    3.597038        rent1
          14    5.505323     married1
          15    5.764081       cars_1
          16    7.352016     MKOOPKLA
          17    8.726838     APERSAUT
          18    2.845487      PWAPART
          19    1.635778       PBRAND
          20    8.770715     PPERSAUT
```

```
In [254]: vifdata.columns
```

```
Out[254]: Index(['PBROM', 'MGEMOMV', 'rank_1', 'job_3', 'insurance', 'ABROM',
                 'MOSHOOFD_5', 'MOSHOOFD_2', 'rank_2', 'job_2', 'ABRAND', 'married2',
                 'MOSHOOFD_10', 'rent1', 'married1', 'cars_1', 'MKOOPKLA', 'APERSAUT',
                 'PWAPART', 'PBRAND', 'PPERSAUT'],
                dtype='object')
```

```
In [255]: clf= RandomForestClassifier(max_depth=1,random_state=0,oob_score=True,n_estimators=1
          clf.fit(vifdata,ticdata['CARAVAN'])
          print(clf.oob_score_)
```

```
         variable_importances=list(zip(vifdata.columns,np.round(clf.feature_importances_,2)))
         feature_importance_matrix=pd.DataFrame(variable_importances).sort_values(by=1)
         print(feature_importance_matrix)
```

```
0.6962457337883959
              0     1
0         PBROM  0.00
2        rank_1  0.00
3         job_3  0.00
5         ABROM  0.00
12  MOSHOOFD_10  0.00
1       MGEMOMV  0.01
9         job_2  0.01
11     married2  0.01
4     insurance  0.02
6    MOSHOOFD_5  0.02
7    MOSHOOFD_2  0.02
8        rank_2  0.02
13        rent1  0.03
10       ABRAND  0.03
14     married1  0.05
15       cars_1  0.05
18      PWAPART  0.10
16     MKOOPKLA  0.11
19       PBRAND  0.14
17     APERSAUT  0.17
20     PPERSAUT  0.18
```

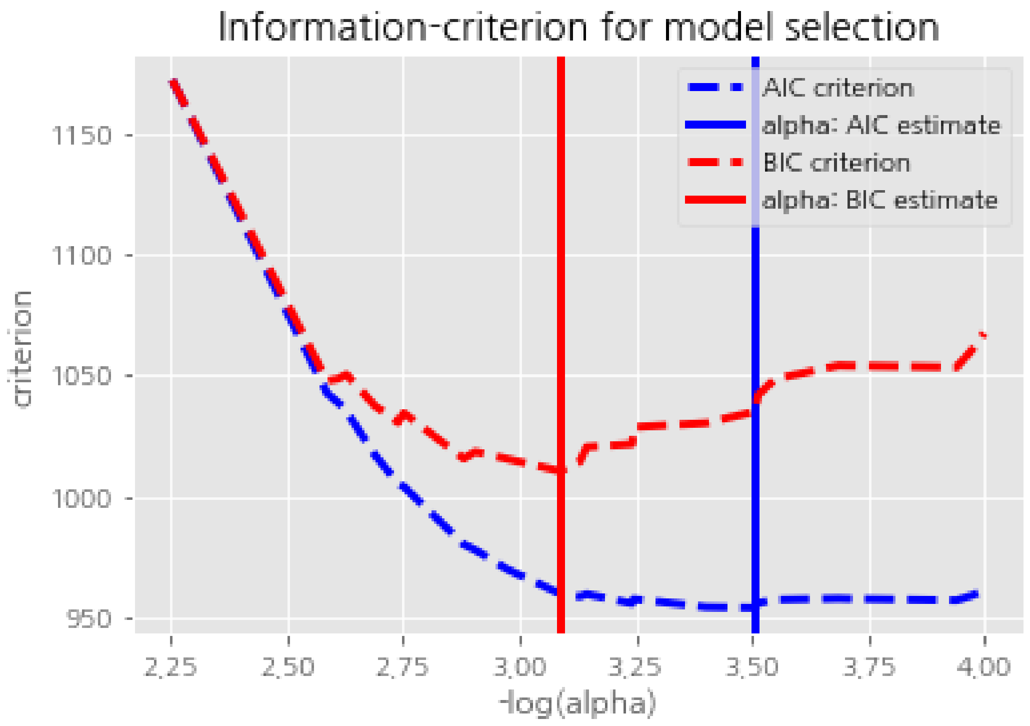
```
In [256]: rf_var=feature_importance_matrix.iloc[:,0]
```

1.0.3 AIC,BIC

```
In [257]: from sklearn.preprocessing import StandardScaler
          from sklearn.linear_model import LassoCV, LassoLarsCV, LassoLarsIC
          from sklearn import datasets

          EPSILON = 1e-4
          X = vifdata[rf_var].copy()
          y = ticdata['CARAVAN']

          rng = np.random.RandomState(42)

          stscaler=StandardScaler() ####
          stscaler.fit(X)
          X=stscaler.transform(X)

          # #############################################################################
          # LassoLarsIC: least angle regression with BIC/AIC criterion

          model_bic = LassoLarsIC(criterion='bic')
          model_bic.fit(X, y)
          alpha_bic_ = model_bic.alpha_

          model_aic = LassoLarsIC(criterion='aic')
          model_aic.fit(X, y)
          alpha_aic_ = model_aic.alpha_


          def plot_ic_criterion(model, name, color):
              alpha_ = model.alpha_ + EPSILON
              alphas_ = model.alphas_ + EPSILON
              criterion_ = model.criterion_
              plt.plot(-np.log10(alphas_), criterion_, '--', color=color,
                       linewidth=3, label='%s criterion' % name)
              plt.axvline(-np.log10(alpha_), color=color, linewidth=3,
                          label='alpha: %s estimate' % name)
              plt.xlabel('-log(alpha)')
              plt.ylabel('criterion')

          plt.figure()
          plot_ic_criterion(model_aic, 'AIC', 'b')
          plot_ic_criterion(model_bic, 'BIC', 'r')
          plt.legend()
          plt.title('Information-criterion for model selection')
```

```
C:\Users\jang\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:645: DataConversionWarr
  return self.partial_fit(X, y)
C:\Users\jang\Anaconda3\lib\site-packages\ipykernel_launcher.py:13: DataConversionWarning: Data
  del sys.path[0]
```

```
Out[257]: Text(0.5, 1.0, 'Information-criterion for model selection')
```



1.0.4 aic aic

```
In [258]: aic_var=vifdata[rf_var].columns[model_aic.coef_!=0]
```

```
In [259]: print(pd.Series(dict(zip(vifdata[rf_var].columns,abs(model_aic.coef_)))).sort_values
```

```
PPERSAUT       0.150990
PWAPART        0.053764
MOSHOOFD_5     0.039220
cars_1         0.035388
MOSHOOFD_10    0.033724
rent1          0.027027
ABROM          0.019995
married1       0.017679
MOSHOOFD_2     0.014854
married2       0.012536
job_2          0.010457
ABRAND         0.007201
MGEMOMV        0.006995
insurance      0.006193
PBRAND         0.005105
MKOOPKLA       0.002960
rank_2         0.000000
APERSAUT       0.000000
```

```
job_3          0.000000
rank_1         0.000000
PBROM          0.000000
dtype: float64
```

1.0.5 bic aic

```
In [260]: bic_var=vifdata[rf_var].columns[model_bic.coef_!=0]
```

```
In [261]: print(pd.Series(dict(zip(vifdata[rf_var].columns,abs(model_bic.coef_)))).sort_values
```

```
PPERSAUT       0.141821
PWAPART        0.044831
cars_1         0.028638
MOSHOOFD_5     0.024242
rent1          0.021253
MOSHOOFD_10    0.016569
MKOOPKLA       0.014054
married1       0.012507
MOSHOOFD_2     0.007839
ABROM          0.006663
rank_2         0.000000
APERSAUT       0.000000
ABRAND         0.000000
insurance      0.000000
married2       0.000000
job_2          0.000000
MGEMOMV        0.000000
PBRAND         0.000000
job_3          0.000000
rank_1         0.000000
PBROM          0.000000
dtype: float64
```

```
In [262]: aic_var
```

```
Out[262]: Index(['ABROM', 'MOSHOOFD_10', 'MGEMOMV', 'job_2', 'married2', 'insurance',
                 'MOSHOOFD_5', 'MOSHOOFD_2', 'rent1', 'ABRAND', 'married1', 'cars_1',
                 'PWAPART', 'MKOOPKLA', 'PBRAND', 'PPERSAUT'],
                dtype='object')
```

```
In [263]: bic_var
```

```
Out[263]: Index(['ABROM', 'MOSHOOFD_10', 'MOSHOOFD_5', 'MOSHOOFD_2', 'rent1', 'married1',
                 'cars_1', 'PWAPART', 'MKOOPKLA', 'PPERSAUT'],
                dtype='object')
```

2

2.0.1 imort

```
In [264]: from sklearn.model_selection import train_test_split
          import sklearn
          from sklearn.naive_bayes import GaussianNB, BernoulliNB
          from sklearn.linear_model import LogisticRegression
          from sklearn.tree import DecisionTreeClassifier
          from sklearn.neural_network import *
          from sklearn.ensemble import RandomForestClassifier
          from sklearn.ensemble import AdaBoostClassifier
          from sklearn.ensemble import GradientBoostingClassifier
          from sklearn.svm import SVC
          import seaborn as sns
          from sklearn.metrics import roc_curve
          from sklearn.metrics import precision_recall_curve
          from sklearn.metrics import roc_auc_score
          from sklearn.metrics import average_precision_score
          from sklearn.model_selection import GridSearchCV
```

2.0.2

```
In [265]: variable=aic_var
          print(" ")
          print(variable)
          print(" :",len(variable))
```

```
Index(['ABROM', 'MOSHOOFD_10', 'MGEMOMV', 'job_2', 'married2', 'insurance',
       'MOSHOOFD_5', 'MOSHOOFD_2', 'rent1', 'ABRAND', 'married1', 'cars_1',
       'PWAPART', 'MKOOPKLA', 'PBRAND', 'PPERSAUT'],
      dtype='object')
 : 16
```

2.0.3 Undersampling

```
In [266]: ticdata['CARAVAN'].value_counts()

Out[266]: 1    586
          0    586
          Name: CARAVAN, dtype: int64
```

Train Set: VAlidation set: Test set 0.7: 0,15:0.15

```
In [267]: ###dataset=allticdata
          dataset=ticdata
          ###var=important_features
```

```
        var=variable


        X = (dataset[dataset[var].columns.values]) ####    X
        y = np.array(dataset['CARAVAN']) #### "CARAVAN" y

        print('\n')
        print('X and y Input Data:   ', X.shape, y.shape)


        X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=

        print('Training Set Shape:   ', X_train.shape, y_train.shape)

        print('Validation Set Shape: ', X_val.shape,y_val.shape)
```

```
X and y Input Data:    (1172, 16) (1172,)
Training Set Shape:    (820, 16) (820,)
Validation Set Shape:  (352, 16) (352,)
```

```
In [268]: print(sum(y_train==0),sum(y_train==1))
```

```
403 417
```

Train Set: VAlidation set: Test set sample_weight   array

```
In [269]: def weight_array(y):
              a=[]
              for i in range(len(y)):
                  if np.hstack(y)[i]==0:
                      a.append(sampleweight_0)
                  else:
                      a.append(sampleweight_1)
              return a
```

```
In [270]: train_sampleweight=weight_array(y_train)
          val_sampleweight=weight_array(y_val)
```

2.0.4

```
In [271]: def calc_lift(x,y,clf,bins=10):
               #Actual Value of y
              y_actual = np.hstack(y)
              #Predicted Probability that y = 1
              y_prob = clf.predict_proba(x)
```

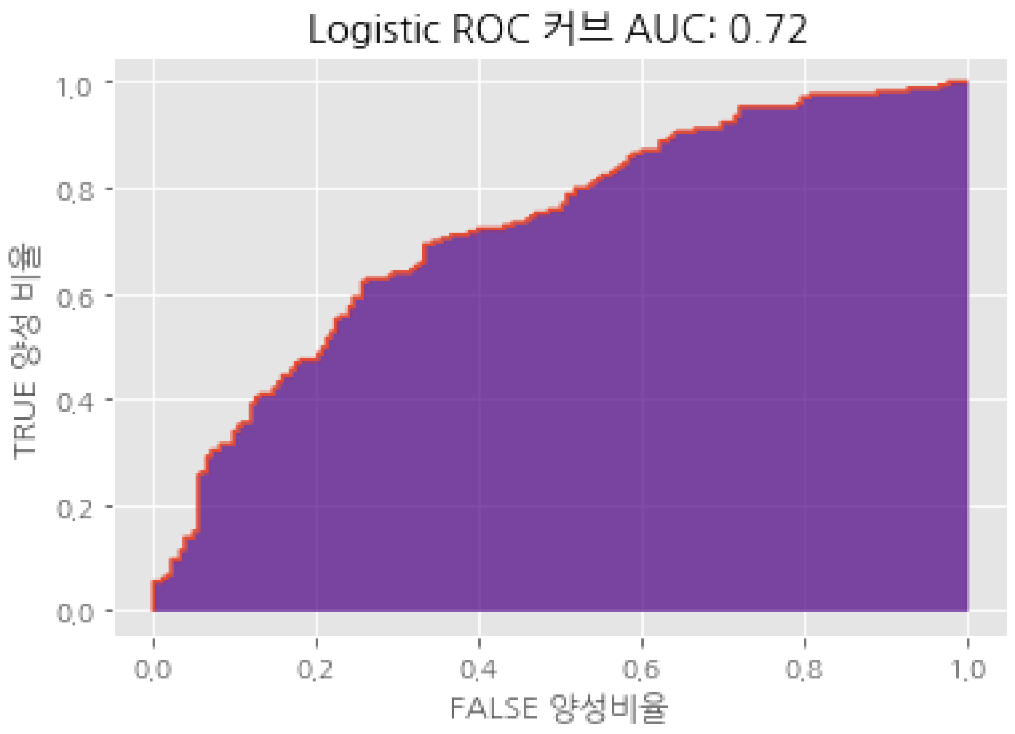
```
        #Predicted Value of Y
        y_pred = clf.predict(x)
        cols = ['ACTUAL','PROB_POSITIVE','PREDICTED']
        data = [y_actual,y_prob[:,1],y_pred]
        df = pd.DataFrame(dict(zip(cols,data)))
        #Observations where y=1
        total_positive_n = df['ACTUAL'].sum()
        #Total Observations
        total_n = df.index.size
        natural_positive_prob = total_positive_n/float(total_n)
        df[''] = pd.qcut(df['PROB_POSITIVE'],bins,labels=False, duplicates='drop')
        pos_group_df = df.groupby('')
        #Percentage of Observations in each Bin where y = 1
        actual=pos_group_df['ACTUAL'].sum().sort_index(ascending=False)
        bin_count=pos_group_df['ACTUAL'].count().sort_index(ascending=False)
        cumsum=np.cumsum(bin_count)
        cumsum_percentage=np.cumsum(bin_count)/np.sum(bin_count)
        lift_positive = pos_group_df['ACTUAL'].sum().sort_index(ascending=False)/pos_grou
        cum_active=np.cumsum(pos_group_df['ACTUAL'].sum().sort_index(ascending=False))/np
        cum_lift_positive=np.cumsum(pos_group_df['ACTUAL'].sum().sort_index(ascending=Fal
        cum_lift_positive=cum_lift_positive/natural_positive_prob
        lift_index_positive = (lift_positive/natural_positive_prob)

        #Consolidate Results into Output Dataframe
        lift_df = pd.DataFrame({ '   ':bin_count
                                   ,'   (%)':lift_positive*100 ,
                                     ' LIFT':lift_index_positive,
                                    '     ':actual,
                                    '   ':np.round(cumsum,1),
                                   '    ':np.round(cumsum_percentage,1),
                                   '    %':np.round(cum_active,2),
                               '    ':np.cumsum(actual),
                               ' lift (%)':cum_lift_positive})
        lift_df.index=[1,2,3,4,5,6,7,8,9,10]
        lift_df.index.name=''
        return lift_df
```

2.1 Logistic Regression

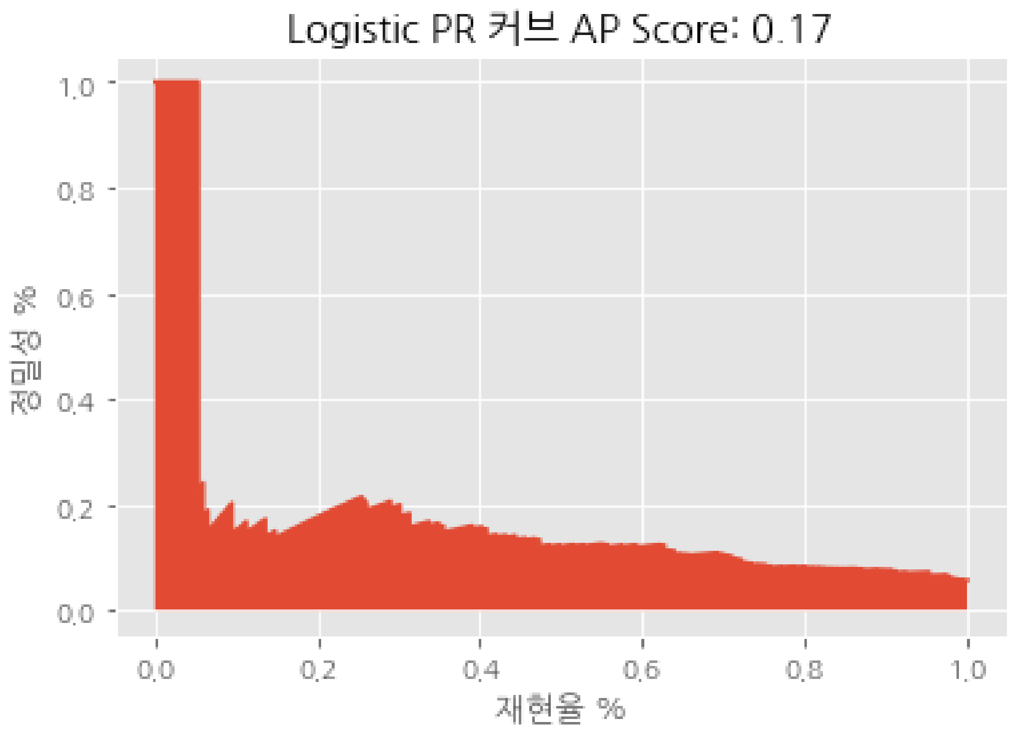
```
In [312]: clf_Log = LogisticRegression(max_iter=100,random_state=10)
          parameters={'solver':['liblinear']}
          clf_Log=GridSearchCV(clf_Log,parameters,cv=10)
          clf_Log=clf_Log.fit(X_train, y_train,sample_weight=train_sampleweight).best_estimato
          y_score = clf_Log.decision_function(X_val)
          y_pred_Log = clf_Log.predict(X_val)
```

```
In [276]: fpr, tpr, thresholds = roc_curve(y_val,y_score,sample_weight=val_sampleweight)
          auc=np.round(roc_auc_score(y_val,y_score,sample_weight=val_sampleweight),2)
```

```
        plt.plot(fpr,tpr)
        plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
        plt.title("Logistic ROC  AUC: %s" %auc)
        plt.xlabel("FALSE ")
        plt.ylabel("TRUE  ")

Out[276]: Text(0, 0.5, 'TRUE  ')
```



```
In [277]: precision, recall, thresholds = precision_recall_curve(y_val, y_score,sample_weight=v
          ap_score_Log=np.round(average_precision_score(y_val, y_score,sample_weight=val_sample
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title("Logistic PR  AP Score: %s"%ap_score_Log)
          plt.xlabel(" %")
          plt.ylabel(" %")

Out[277]: Text(0, 0.5, ' %')
```



```
In [278]: calc_lift(X_train,y_train,clf_Log,bins=10)

Out[278]:           (%)    LIFT                 \

          1          82     84.146341  1.654676                69         82           0.1
          2          81     75.308642  1.480889                61        163           0.2
          3          83     71.084337  1.397822                59        246           0.3
          4          82     60.975610  1.199041                50        328           0.4
          5          82     63.414634  1.247002                52        410           0.5
          6          82     48.780488  0.959233                40        492           0.6
          7          82     35.365854  0.695444                29        574           0.7
          8          82     36.585366  0.719424                30        656           0.8
          9          82     26.829268  0.527578                22        738           0.9
          10         82      6.097561  0.119904                 5        820           1.0

                 %        lift (%)

          1          0.84             69      1.654676
          2          0.80            130      1.568316
          3          0.77            189      1.510791
          4          0.73            239      1.432854
          5          0.71            291      1.395683
          6          0.67            331      1.322942
          7          0.63            360      1.233299
```

```
8          0.59           390     1.169065
9          0.56           412     1.097788
10         0.51           417     1.000000
```

```
In [279]: calc_lift(X_val,y_val,clf_Log,bins=10)
```

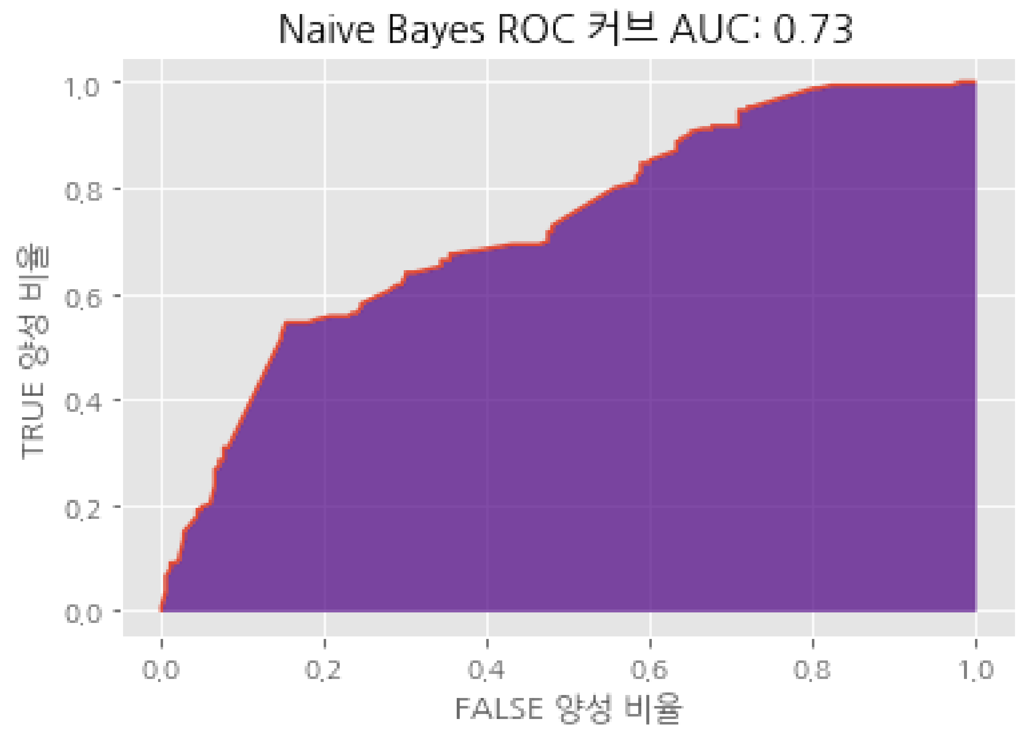
```
Out[279]:          (%)    LIFT                 \

          1          36     72.222222  1.504274             26         36            0.1
          2          35     77.142857  1.606762             27         71            0.2
          3          35     62.857143  1.309214             22        106            0.3
          4          35     62.857143  1.309214             22        141            0.4
          5          35     51.428571  1.071175             18        176            0.5
          6          35     31.428571  0.654607             11        211            0.6
          7          35     42.857143  0.892646             15        246            0.7
          8          35     37.142857  0.773626             13        281            0.8
          9          35     31.428571  0.654607             11        316            0.9
          10         36     11.111111  0.231427              4        352            1.0

                 %        lift (%)

          1          0.72            26     1.504274
          2          0.75            53     1.554796
          3          0.71            75     1.473708
          4          0.69            97     1.432876
          5          0.65           115     1.360947
          6          0.60           126     1.243781
          7          0.57           141     1.193823
          8          0.55           154     1.141485
          9          0.52           165     1.087559
          10         0.48           169     1.000000
```

2.1.1 NAIVE BAYES CLASSIFIER

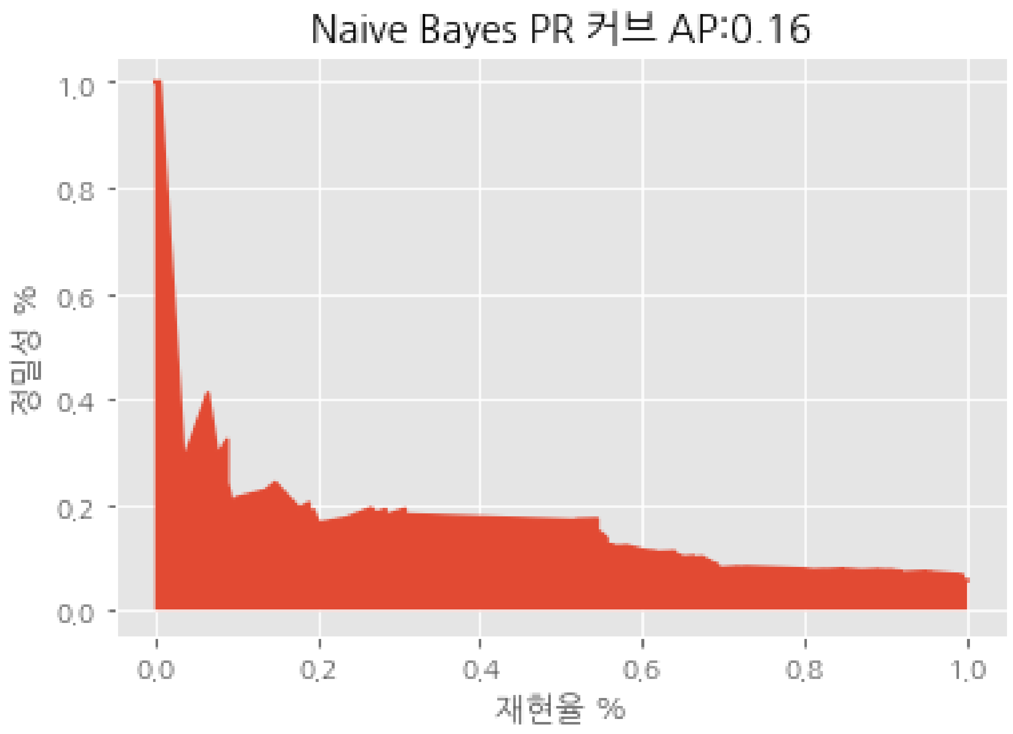
```
In [280]: clf_NB = BernoulliNB()
          clf_NB.fit(X_train, y_train,sample_weight=train_sampleweight)
          y_pred_NB = clf_NB.predict(X_val)
```

```
In [281]: y_score = clf_NB.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          auc=np.round(roc_auc_score(y_val,y_score,sample_weight=val_sampleweight),2)
          fpr, tpr, thresholds = roc_curve(y_val,y_score)
          plt.plot(fpr,tpr)
          plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
          plt.title("Naive Bayes ROC  AUC: %s"%auc)
          plt.xlabel("FALSE  ")
          plt.ylabel("TRUE  ")
```

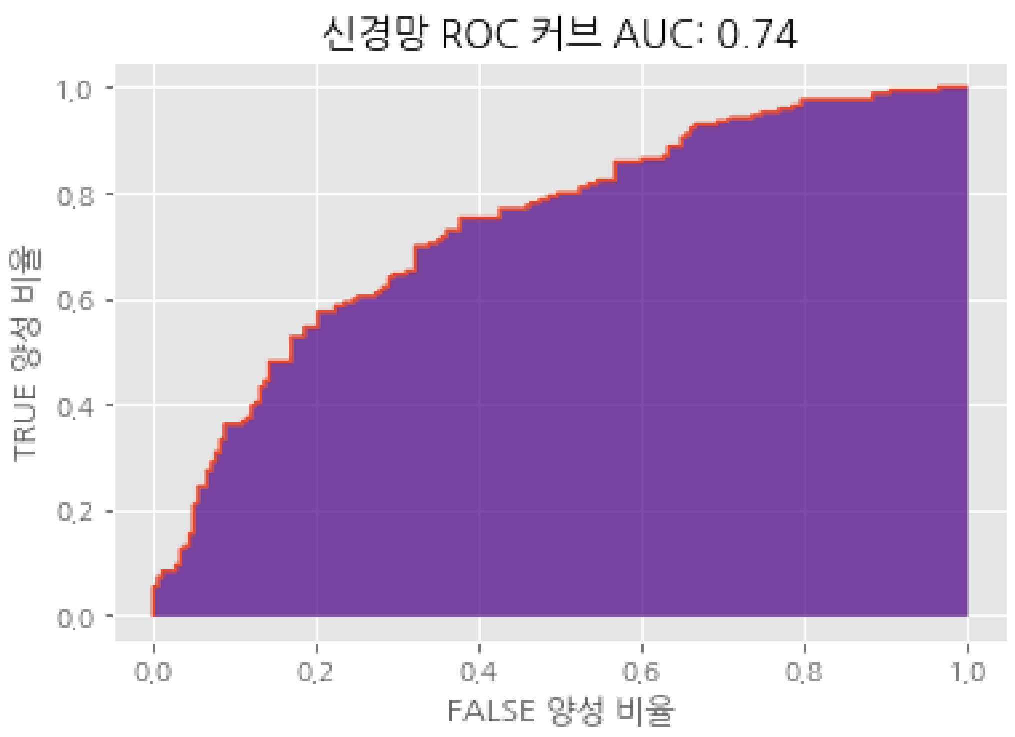
```
Out[281]: Text(0, 0.5, 'TRUE  ')
```



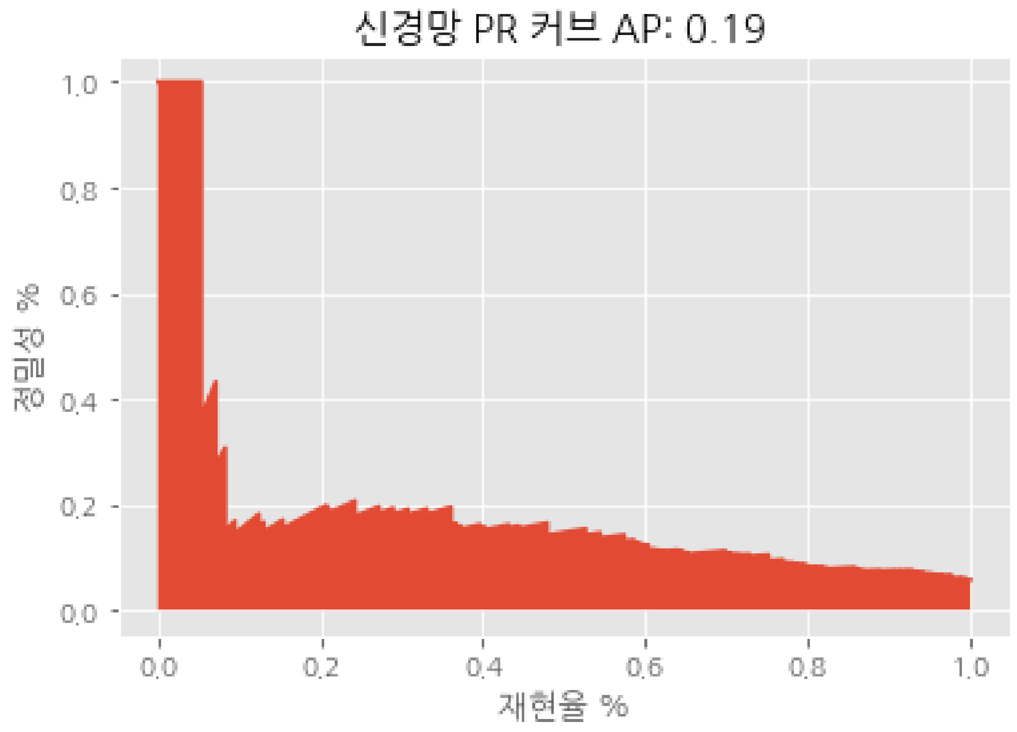
```
In [282]: y_score = clf_NB.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          precision, recall, thresholds = precision_recall_curve(y_val, y_score,sample_weight=v
          ap_score_NB=np.round(average_precision_score(y_val, y_score,sample_weight=val_samplew
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title("Naive Bayes PR  AP:%s"%ap_score_NB)
          plt.xlabel(" %")
          plt.ylabel(" %")

Out[282]: Text(0, 0.5, ' %')
```



```
In [283]: clf_MLP = MLPClassifier(alpha=1e-05, hidden_layer_sizes=(100))

          clf_MLP.fit(X_train, y_train)
          y_pred_MLP = clf_MLP.predict(X_val)

C:\Users\jang\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:562:
  % self.max_iter, ConvergenceWarning)


In [284]: y_score = clf_MLP.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          fpr, tpr, thresholds = roc_curve(y_val,y_score)
          auc=np.round(roc_auc_score(y_val,y_score,sample_weight=val_sampleweight),2)
          plt.plot(fpr,tpr)
          plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
          plt.title(" ROC  AUC: %s"%auc)
          plt.xlabel("FALSE  ")
          plt.ylabel("TRUE  ")

Out[284]: Text(0, 0.5, 'TRUE  ')
```



```
In [285]: y_score = clf_MLP.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          precision, recall, thresholds = precision_recall_curve(y_val, y_score,sample_weight=v
          ap_score_MLP=np.round(average_precision_score(y_val, y_score,sample_weight=val_sample
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title(" PR  AP: %s "%ap_score_MLP)
          plt.xlabel(" %")
          plt.ylabel(" %")

Out[285]: Text(0, 0.5, ' %')
```



```
In [286]: calc_lift(X_train,y_train,clf_MLP)

Out[286]:          (%)     LIFT                 \

        1          82      91.463415   1.798561              75          82           0.1
        2          82      79.268293   1.558753              65         164           0.2
        3          82      64.634146   1.270983              53         246           0.3
        4          82      69.512195   1.366906              57         328           0.4
        5          82      57.317073   1.127098              47         410           0.5
        6          82      54.878049   1.079137              45         492           0.6
        7          82      37.804878   0.743405              31         574           0.7
        8          82      31.707317   0.623501              26         656           0.8
        9          82      18.292683   0.359712              15         738           0.9
        10         82       3.658537   0.071942               3         820           1.0

                 %        lift (%)

        1          0.91             75      1.798561
        2          0.85            140      1.678657
        3          0.78            193      1.542766
        4          0.76            250      1.498801
        5          0.72            297      1.424460
        6          0.70            342      1.366906
        7          0.65            373      1.277835
```

```
8          0.61             399     1.196043
9          0.56             414     1.103118
10         0.51             417     1.000000
```

```
In [287]: calc_lift(X_val,y_val,clf_MLP)
```

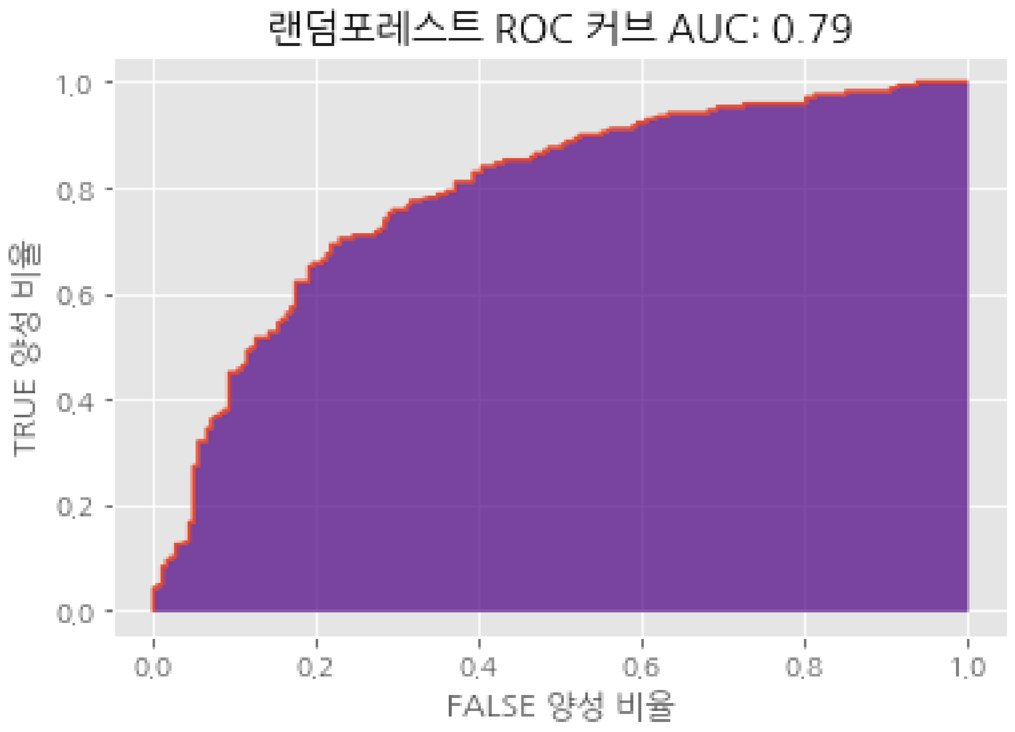
```
Out[287]:          (%)    LIFT                 \

          1          36     75.000000  1.562130             27          36              0.1
          2          35     82.857143  1.725782             29          71              0.2
          3          35     68.571429  1.428233             24         106              0.3
          4          35     54.285714  1.130685             19         141              0.4
          5          35     51.428571  1.071175             18         176              0.5
          6          35     37.142857  0.773626             13         211              0.6
          7          35     34.285714  0.714117             12         246              0.7
          8          35     42.857143  0.892646             15         281              0.8
          9          35     22.857143  0.476078              8         316              0.9
          10         36     11.111111  0.231427              4         352              1.0

                 %        lift (%)

          1          0.75              27     1.562130
          2          0.79              56     1.642804
          3          0.75              80     1.571955
          4          0.70              99     1.462420
          5          0.66             117     1.384615
          6          0.62             130     1.283266
          7          0.58             142     1.202290
          8          0.56             157     1.163722
          9          0.52             165     1.087559
          10         0.48             169     1.000000
```

2.2

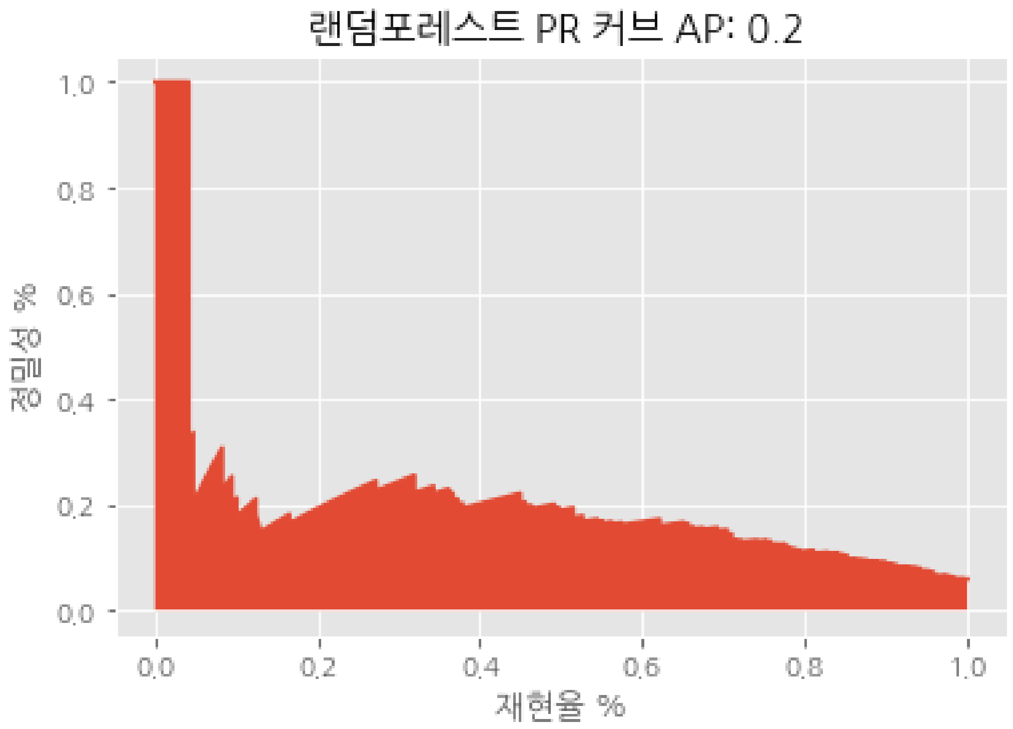
```
In [288]: clf_RF = RandomForestClassifier(n_estimators=500, criterion='gini', max_depth=1)
          clf_RF.fit(X_train, y_train,sample_weight=train_sampleweight)
          y_pred_RF = clf_RF.predict(X_val)
```

```
In [289]: y_score = clf_RF.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          fpr, tpr, thresholds = roc_curve(y_val,y_score)
          auc=np.round(roc_auc_score(y_val,y_score,sample_weight=val_sampleweight),2)
          plt.plot(fpr,tpr)
          plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
          plt.title(" ROC  AUC: %s"%auc)
          plt.xlabel("FALSE  ")
          plt.ylabel("TRUE  ")
```

```
Out[289]: Text(0, 0.5, 'TRUE  ')
```



```
In [290]: y_score = clf_RF.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          precision, recall, thresholds = precision_recall_curve(y_val, y_score,sample_weight=
          ap_score_RF=np.round(average_precision_score(y_val, y_score,sample_weight=val_sample
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title(" PR  AP: %s "%ap_score_RF)
          plt.xlabel(" %")
          plt.ylabel(" %")

Out[290]: Text(0, 0.5, ' %')
```



```
In [291]: calc_lift(X_train,y_train,clf_RF)
```

```
Out[291]:          (%)    LIFT                \

        1          82      89.024390   1.750600               73          82           0.1
        2          82      70.731707   1.390887               58         164           0.2
        3          82      71.951220   1.414868               59         246           0.3
        4          82      62.195122   1.223022               51         328           0.4
        5          82      53.658537   1.055156               44         410           0.5
        6          82      56.097561   1.103118               46         492           0.6
        7          82      40.243902   0.791367               33         574           0.7
        8          82      19.512195   0.383693               16         656           0.8
        9          82      21.951220   0.431655               18         738           0.9
        10         82      23.170732   0.455635               19         820           1.0

                %        lift (%)

        1          0.89            73      1.750600
        2          0.80           131      1.570743
        3          0.77           190      1.518785
        4          0.73           241      1.444844
        5          0.70           285      1.366906
        6          0.67           331      1.322942
        7          0.63           364      1.247002
```

```
8          0.58            380     1.139089
9          0.54            398     1.060485
10         0.51            417     1.000000
```

```
In [292]: calc_lift(X_val,y_val,clf_RF)
```

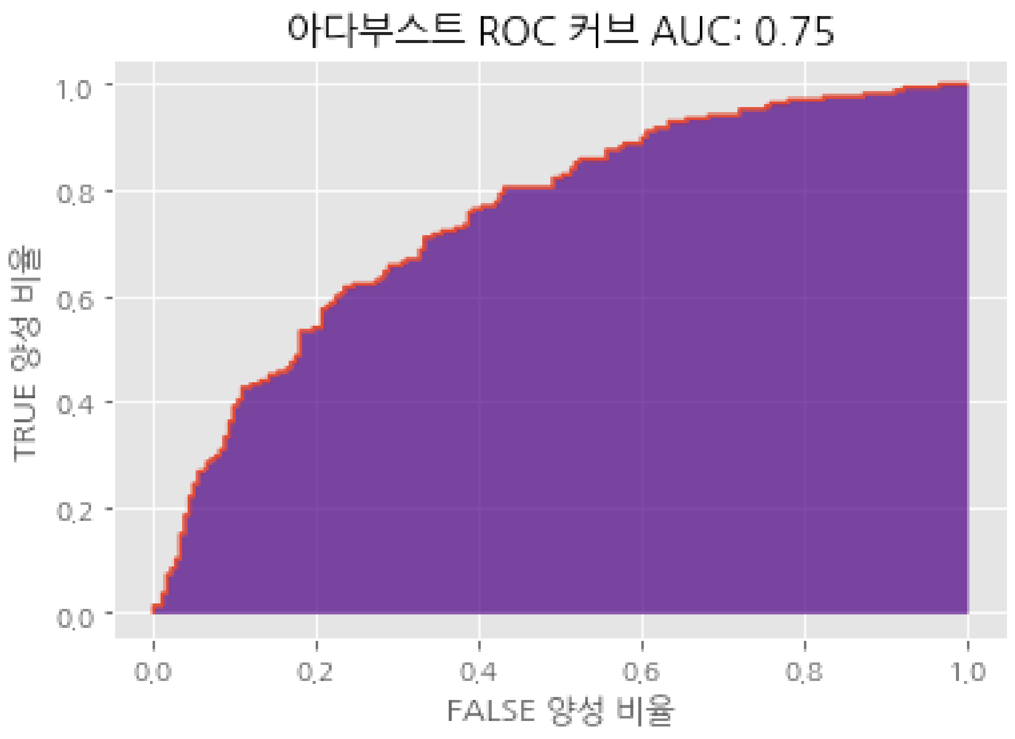
```
Out[292]:          (%)    LIFT                 \

          1          36     77.777778  1.619987             28          36            0.1
          2          35     85.714286  1.785292             30          71            0.2
          3          35     74.285714  1.547253             26         106            0.3
          4          35     62.857143  1.309214             22         141            0.4
          5          35     51.428571  1.071175             18         176            0.5
          6          35     42.857143  0.892646             15         211            0.6
          7          35     34.285714  0.714117             12         246            0.7
          8          35     22.857143  0.476078              8         281            0.8
          9          35     17.142857  0.357058              6         316            0.9
          10         36     11.111111  0.231427              4         352            1.0

                %         lift (%)

          1          0.78             28     1.619987
          2          0.82             58     1.701475
          3          0.79             84     1.650553
          4          0.75            106     1.565823
          5          0.70            124     1.467456
          6          0.66            139     1.372108
          7          0.61            151     1.278491
          8          0.57            159     1.178547
          9          0.52            165     1.087559
          10         0.48            169     1.000000
```

2.3

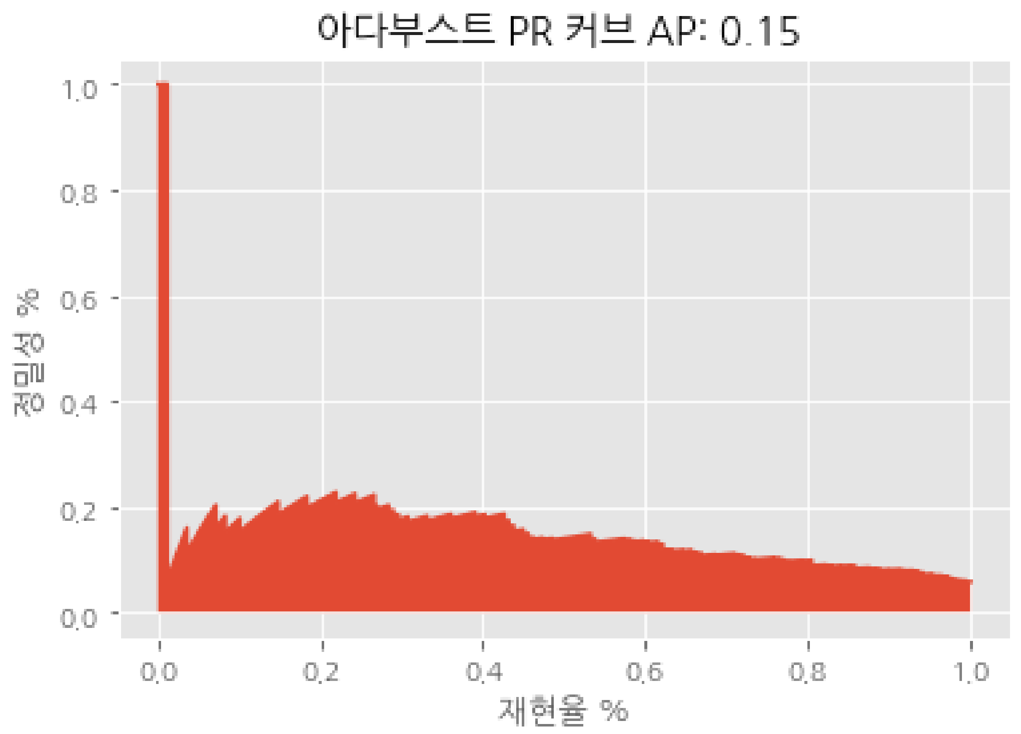
```
In [293]: clf_AdaB = AdaBoostClassifier(n_estimators=100)
          clf_AdaB.fit(X_train, y_train,sample_weight=train_sampleweight)
          y_pred_AdaB = clf_AdaB.predict(X_val)
```

```
In [294]: y_score = clf_AdaB.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          fpr, tpr, thresholds = roc_curve(y_val,y_score)
          auc=np.round(roc_auc_score(y_val,y_score,sample_weight=val_sampleweight),2)
          plt.plot(fpr,tpr)
          plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
          plt.title(" ROC  AUC: %s"%auc)
          plt.xlabel("FALSE  ")
          plt.ylabel("TRUE  ")
```

```
Out[294]: Text(0, 0.5, 'TRUE  ')
```



```
In [295]: y_score = clf_AdaB.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          precision, recall, thresholds = precision_recall_curve(y_val, y_score,sample_weight=v
          ap_score_ADA=np.round(average_precision_score(y_val, y_score,sample_weight=val_sample
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title(" PR  AP: %s "%ap_score_ADA)
          plt.xlabel(" %")
          plt.ylabel(" %")

Out[295]: Text(0, 0.5, ' %')
```



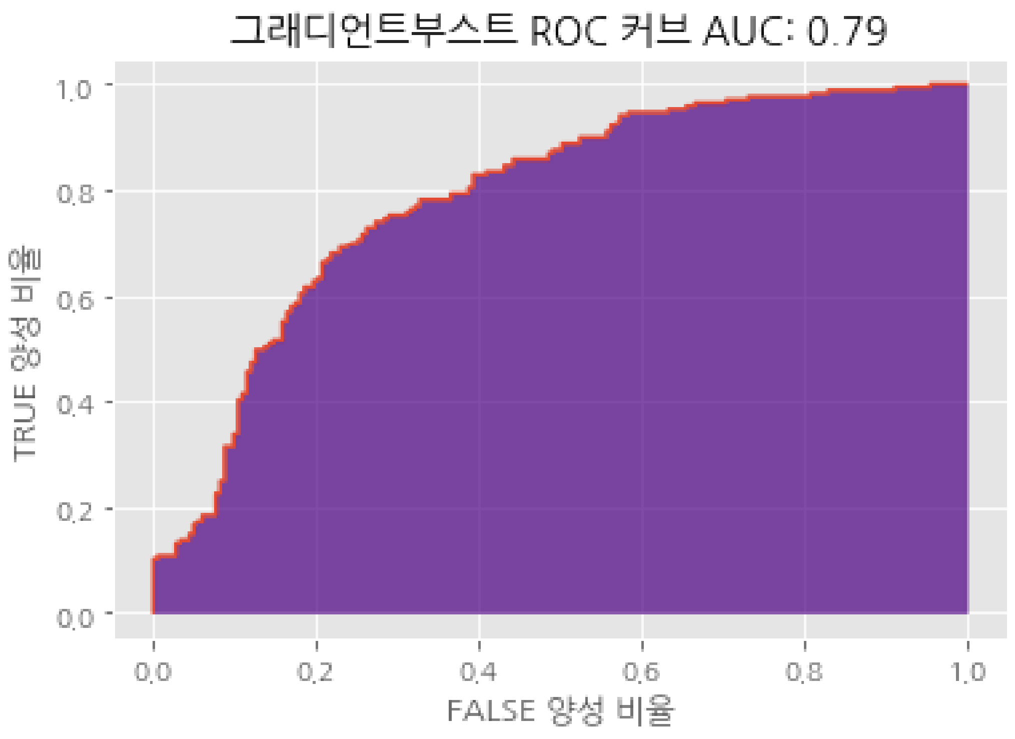
```
In [296]: calc_lift(X_train,y_train,clf_AdaB)

Out[296]:          (%)    LIFT                \

          1          82     92.682927   1.822542            76          82           0.1
          2          82     82.926829   1.630695            68         164           0.2
          3          82     71.951220   1.414868            59         246           0.3
          4          82     70.731707   1.390887            58         328           0.4
          5          82     58.536585   1.151079            48         410           0.5
          6          82     53.658537   1.055156            44         492           0.6
          7          82     40.243902   0.791367            33         574           0.7
          8          82     23.170732   0.455635            19         656           0.8
          9          82     14.634146   0.287770            12         738           0.9
          10         82      0.000000   0.000000             0         820           1.0

                %        lift (%)

          1          0.93             76      1.822542
          2          0.88            144      1.726619
          3          0.83            203      1.622702
          4          0.80            261      1.564748
          5          0.75            309      1.482014
          6          0.72            353      1.410871
          7          0.67            386      1.322371
```

```
8           0.62             405     1.214029
9           0.57             417     1.111111
10          0.51             417     1.000000
```

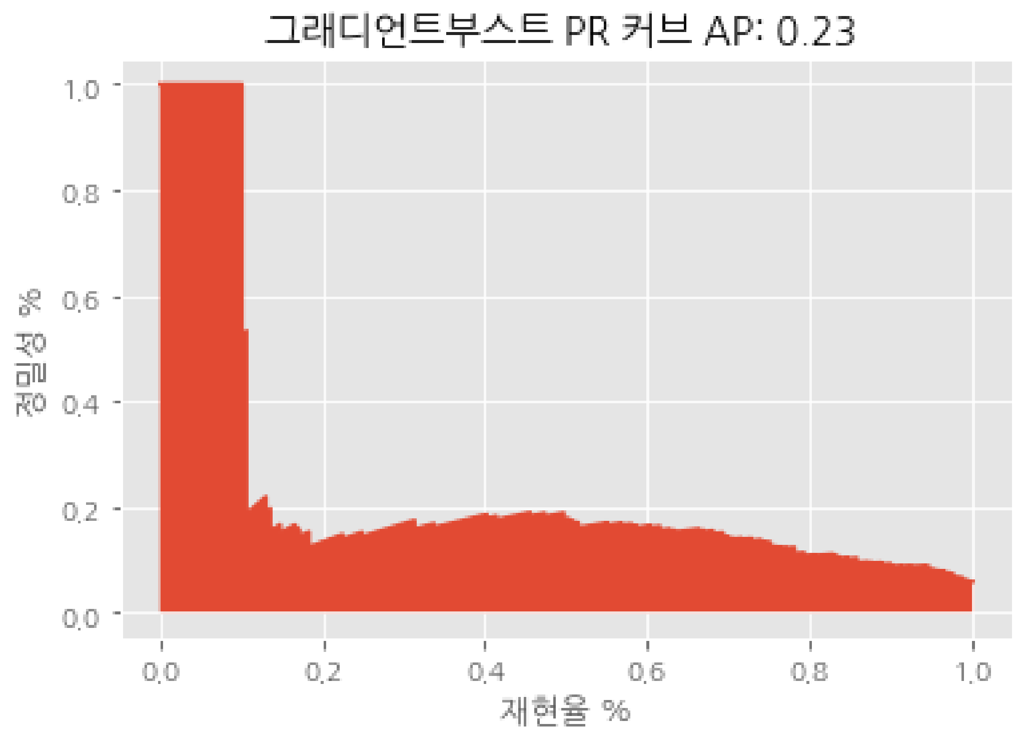
```
In [297]: calc_lift(X_val,y_val,clf_AdaB)

Out[297]:          (%)    LIFT                 \

          1          36     80.555556  1.677844              29          36            0.1
          2          35     74.285714  1.547253              26          71            0.2
          3          35     62.857143  1.309214              22         106            0.3
          4          35     65.714286  1.368724              23         141            0.4
          5          35     45.714286  0.952156              16         176            0.5
          6          35     48.571429  1.011665              17         211            0.6
          7          35     34.285714  0.714117              12         246            0.7
          8          35     37.142857  0.773626              13         281            0.8
          9          35     20.000000  0.416568               7         316            0.9
          10         36     11.111111  0.231427               4         352            1.0

                 %        lift (%)

          1          0.81              29     1.677844
          2          0.77              55     1.613468
          3          0.73              77     1.513007
          4          0.71             100     1.477192
          5          0.66             116     1.372781
          6          0.63             133     1.312880
          7          0.59             145     1.227690
          8          0.56             158     1.171134
          9          0.52             165     1.087559
          10         0.48             169     1.000000
```

2.4

```
In [298]: clf_GB = GradientBoostingClassifier(n_estimators=150, learning_rate=0.05, random_sta
          clf_GB.fit(X_train, y_train,sample_weight=train_sampleweight)
          y_pred_GB = clf_GB.predict(X_val)

In [299]: y_score = clf_GB.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          fpr, tpr, thresholds = roc_curve(y_val,y_score)
          auc=np.round(roc_auc_score(y_val,y_score,sample_weight=val_sampleweight),2)
          plt.plot(fpr,tpr)
          plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
          plt.title(" ROC  AUC: %s"%auc)
          plt.xlabel("FALSE  ")
          plt.ylabel("TRUE  ")

Out[299]: Text(0, 0.5, 'TRUE  ')
```



```
In [300]: y_score = clf_GB.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          precision, recall, thresholds = precision_recall_curve(y_val, y_score,sample_weight=v
          ap_score_GB=np.round(average_precision_score(y_val, y_score,sample_weight=val_samplew
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title(" PR  AP: %s "%ap_score_GB)
          plt.xlabel(" %")
          plt.ylabel(" %")

Out[300]: Text(0, 0.5, ' %')
```



```
In [301]: calc_lift(X_train,y_train,clf_GB)

Out[301]:          (%)     LIFT                 \

          1          82    100.000000  1.966427             82          82           0.1
          2          81     93.827160  1.845042             76         163           0.2
          3          83     81.927711  1.611049             68         246           0.3
          4          82     70.731707  1.390887             58         328           0.4
          5          82     56.097561  1.103118             46         410           0.5
          6          82     34.146341  0.671463             28         492           0.6
          7          82     30.487805  0.599520             25         574           0.7
          8          82     30.487805  0.599520             25         656           0.8
          9          82      9.756098  0.191847              8         738           0.9
          10         82      1.219512  0.023981              1         820           1.0

                 %         lift (%)

          1          1.00              82      1.966427
          2          0.97             158      1.906107
          3          0.92             226      1.806555
          4          0.87             284      1.702638
          5          0.80             330      1.582734
          6          0.73             358      1.430855
          7          0.67             383      1.312093
```

```
        8           0.62            408     1.223022
        9           0.56            416     1.108447
        10          0.51            417     1.000000
```

```
In [302]: calc_lift(X_val,y_val,clf_GB)
```

```
Out[302]:           (%)    LIFT                 \

        1           36     75.000000  1.562130            27          36            0.1
        2           35     74.285714  1.547253            26          71            0.2
        3           35     85.714286  1.785292            30         106            0.3
        4           35     62.857143  1.309214            22         141            0.4
        5           34     58.823529  1.225200            20         175            0.5
        6           36     38.888889  0.809993            14         211            0.6
        7           35     31.428571  0.654607            11         246            0.7
        8           35     31.428571  0.654607            11         281            0.8
        9           35     14.285714  0.297549             5         316            0.9
        10          36      8.333333  0.173570             3         352            1.0

              %        lift (%)

        1           0.75             27     1.562130
        2           0.75             53     1.554796
        3           0.78             83     1.630903
        4           0.74            105     1.551051
        5           0.71            125     1.487743
        6           0.66            139     1.372108
        7           0.61            150     1.270025
        8           0.57            161     1.193371
        9           0.53            166     1.094150
        10          0.48            169     1.000000
```

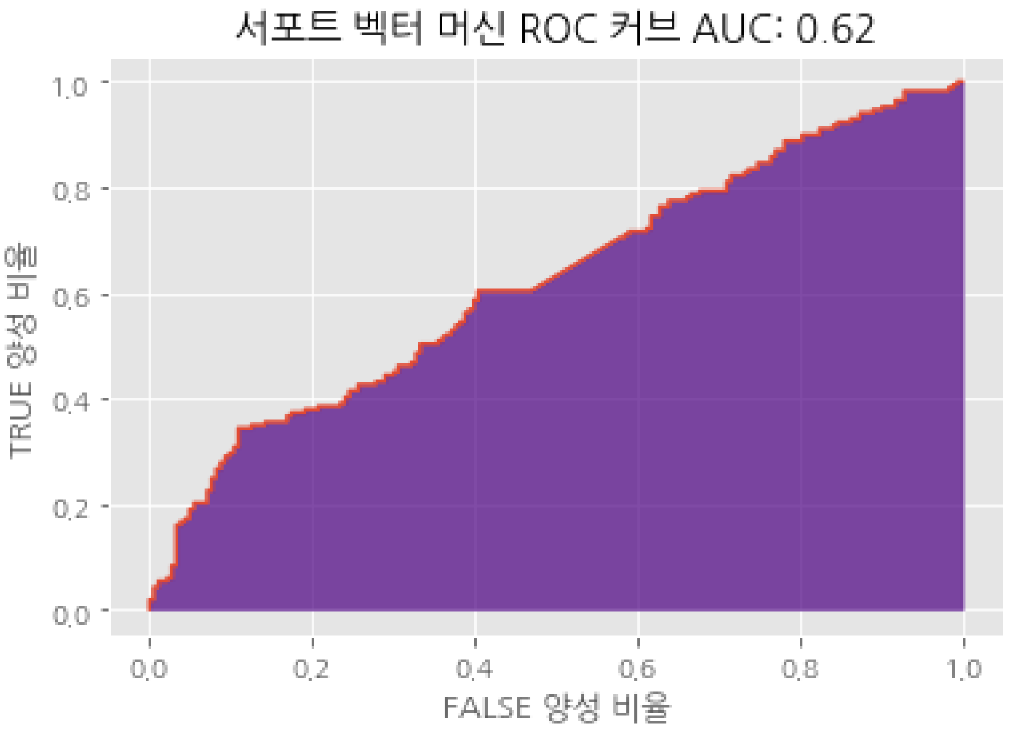
2.4.1 SVM ( )

```
In [303]: clf_SVM = SVC(C=0.1,probability=True)
          clf_SVM.fit(X_train, y_train,sample_weight=train_sampleweight)
          y_pred_SVM = clf_SVM.predict(X_val)
```

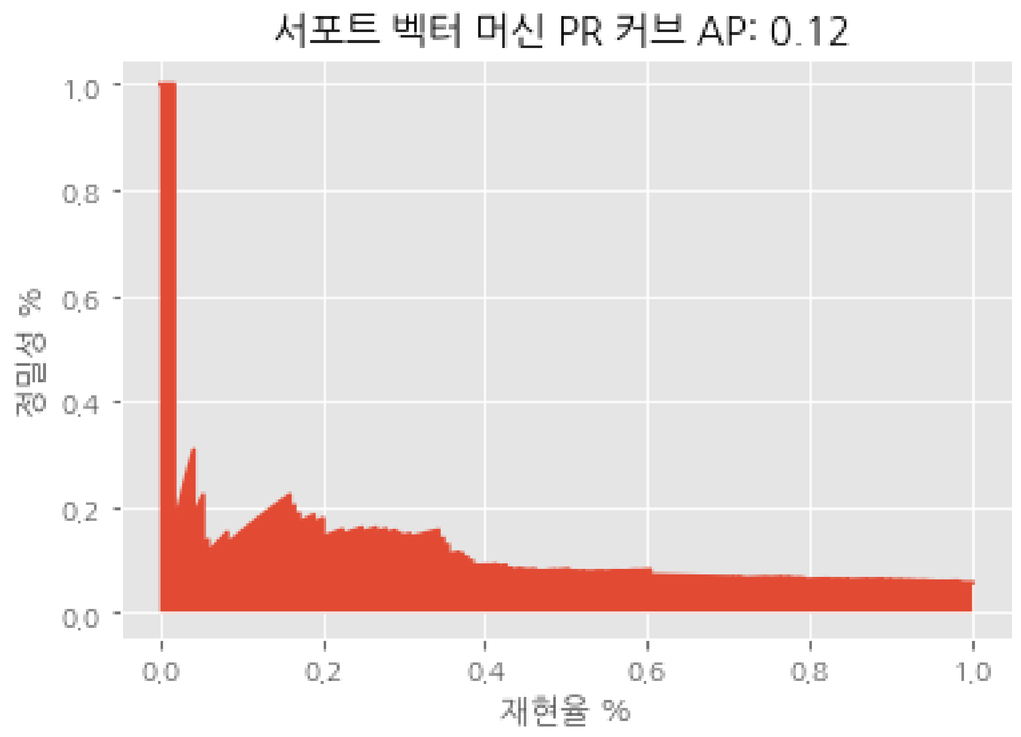
```
C:\Users\jang\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default v
  "avoid this warning.", FutureWarning)
```

```
In [304]: y_score = clf_SVM.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          fpr, tpr, thresholds = roc_curve(y_val,y_score)
          auc=np.round(roc_auc_score(y_val,y_score,sample_weight=val_sampleweight),2)
          plt.plot(fpr,tpr)
          plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
          plt.title("   ROC  AUC: %s"%auc)
```

```
        plt.xlabel("FALSE  ")
        plt.ylabel("TRUE  ")

Out[304]: Text(0, 0.5, 'TRUE  ')
```



```
In [305]: y_score = clf_SVM.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          precision, recall, thresholds = precision_recall_curve(y_val, y_score,sample_weight=v
          ap_score_SVM=np.round(average_precision_score(y_val, y_score,sample_weight=val_sampl
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title("   PR  AP: %s "%ap_score_SVM)
          plt.xlabel(" %")
          plt.ylabel(" %")

Out[305]: Text(0, 0.5, ' %')
```



```
In [306]: calc_lift(X_train,y_train,clf_SVM)

Out[306]:           (%)    LIFT                 \

          1          81      97.530864  1.917873                  79          81            0.1
          2          83      77.108434  1.516281                  64         164            0.2
          3          82      50.000000  0.983213                  41         246            0.3
          4          82      43.902439  0.863309                  36         328            0.4
          5          82      51.219512  1.007194                  42         410            0.5
          6          21      42.857143  0.842754                   9         431            0.5
          7         143      44.755245  0.880079                  64         574            0.7
          8          82      37.804878  0.743405                  31         656            0.8
          9          82      31.707317  0.623501                  26         738            0.9
          10         82      30.487805  0.599520                  25         820            1.0

                %        lift (%)

          1          0.98             79      1.917873
          2          0.87            143      1.714628
          3          0.75            184      1.470823
          4          0.67            220      1.318945
          5          0.64            262      1.256595
          6          0.63            271      1.236431
          7          0.58            335      1.147653
```

```
8           0.56             366     1.097122
9           0.53             392     1.044498
10          0.51             417     1.000000
```

```
In [307]: calc_lift(X_val,y_val,clf_SVM)
```

```
Out[307]:           (%)    LIFT                 \

          1         36     77.777778  1.619987              28          36            0.1
          2         35     68.571429  1.428233              24          71            0.2
          3         35     37.142857  0.773626              13         106            0.3
          4         35     45.714286  0.952156              16         141            0.4
          5         35     60.000000  1.249704              21         176            0.5
          6         12      0.000000  0.000000               0         188            0.5
          7         58     46.551724  0.969598              27         246            0.7
          8         35     40.000000  0.833136              14         281            0.8
          9         35     40.000000  0.833136              14         316            0.9
          10        36     33.333333  0.694280              12         352            1.0

                 %        lift (%)

          1         0.78              28     1.619987
          2         0.73              52     1.525460
          3         0.61              65     1.277213
          4         0.57              81     1.196525
          5         0.58             102     1.207101
          6         0.54             102     1.130052
          7         0.52             129     1.092221
          8         0.51             143     1.059951
          9         0.50             157     1.034829
          10        0.48             169     1.000000
```

2.4.2 ROC curves should be used when there are roughly equal numbers of observations for each class.

Precision-Recall curves should be used when there is a moderate to large class imbalance.

However, ROC curves can present an overly optimistic view of an algorithm's performance if there is a large skew in the class distribution. [...] Precision-Recall (PR) curves, often used in Information Retrieval , have been cited as an alternative to ROC curves for tasks with a large skew in the class distribution.

2.4.3 ROC curve PR curve .

```
In [308]: print('      Validation AP Score     ')
          print('--------------------------------')
          print('Naive Bayes                      AP  Score: %s '%ap_score_NB )
          print('Neural Network                   AP  Score: %s '%ap_score_MLP )
```

```
        print('Logistic Regression           AP  Score: %s '%ap_score_Log )
        print('Random Forest                 AP  Score: %s '%ap_score_RF )
        print('AdaBoost                      AP  Score: %s '%ap_score_ADA )
        print('GradientBoost                 AP  Score: %s '%ap_score_GB )
        print('Support Vector Machine        AP  Score: %s '%ap_score_SVM )
```

```
      Validation AP Score
--------------------------------
Naive Bayes                     AP  Score: 0.16
Neural Network                  AP  Score: 0.19
Logistic Regression             AP  Score: 0.17
Random Forest                   AP  Score: 0.2
AdaBoost                        AP  Score: 0.15
GradientBoost                   AP  Score: 0.23
Support Vector Machine          AP  Score: 0.12
```

2.4.4 RandomForest , Logistic

```
In [309]: print("RandomForest  ")
          variable_importances=list(zip(X_train.columns,np.round(clf_RF.feature_importances_,2
          feature_importance_matrix=pd.DataFrame(variable_importances).sort_values(by=1)
          print(feature_importance_matrix)
```

```
RandomForest
              0     1
0         ABROM  0.00
1   MOSHOOFD_10  0.00
6    MOSHOOFD_5  0.00
9        ABRAND  0.00
2       MGEMOMV  0.01
4      married2  0.02
5     insurance  0.02
3         job_2  0.04
8         rent1  0.05
7    MOSHOOFD_2  0.06
10     married1  0.06
11       cars_1  0.09
14       PBRAND  0.11
12      PWAPART  0.12
13     MKOOPKLA  0.16
15     PPERSAUT  0.26
```

```
In [310]: print("Logistic  ")
          print(pd.DataFrame(zip(X_train.columns,np.hstack(clf_Log.coef_))).sort_values([1]))
```

```
Logistic
              0          1
```

```
0          ABROM -1.546908
1    MOSHOOFD_10 -1.453082
11        cars_1 -1.148155
10      married1 -1.020027
6     MOSHOOFD_5 -0.896549
4       married2 -0.833053
3          job_2 -0.751103
8          rent1 -0.520202
14        PBRAND -0.516569
2        MGEMOMV -0.119983
13      MKOOPKLA  0.071469
9         ABRAND  0.085347
5      insurance  0.135476
7     MOSHOOFD_2  0.317492
12       PWAPART  1.130559
15      PPERSAUT  5.159477
```

```
In [ ]:
```