3? jupyter notebook(??)

June 8, 2019

0.0.1

```
In [480]: 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
          import seaborn as sns
          from sklearn import preprocessing
          from sklearn.utils.class_weight import compute_sample_weight
          %matplotlib inline
```

```
In [481]: data=pd.read_csv('_satisfaction.csv',index_col='id')
```

0.0.2 matplotlib

```
In [482]: plt.style.use('classic') ###CLASSIC style
```

```
In [483]: import matplotlib.font_manager as fm
          print ('  : ', mpl.matplotlib_fname())###matplotlib
```

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

```
In [484]: plt.rc('font', family='NanumGothic')###
```

10% . 120k-> 12k ( )

```
In [485]: data=data.sample(12000,random_state=1016)
```

```
In [486]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12000 entries, 55304 to 86560
Data columns (total 23 columns):
satisfaction_v2                    12000 non-null int64
Gender                             12000 non-null int64
```

```
Customer Type                          12000 non-null int64
Age                                    12000 non-null int64
Type of Travel                         12000 non-null int64
Class                                  12000 non-null int64
Flight Distance                        12000 non-null int64
Seat comfort                           12000 non-null int64
Departure/Arrival time convenient      12000 non-null int64
Food and drink                         12000 non-null int64
Gate location                          12000 non-null int64
Inflight wifi service                  12000 non-null int64
Inflight entertainment                 12000 non-null int64
Online support                         12000 non-null int64
Ease of Online booking                 12000 non-null int64
On-board service                       12000 non-null int64
Leg room service                       12000 non-null int64
Baggage handling                       12000 non-null int64
Checkin service                        12000 non-null int64
Cleanliness                            12000 non-null int64
Online boarding                        12000 non-null int64
Departure Delay in Minutes             12000 non-null int64
Arrival Delay in Minutes               11958 non-null float64
dtypes: float64(1), int64(22)
memory usage: 2.2 MB
```

```
In [487]: data.describe()
```

Out[487]:

	satisfaction_v2	Gender	Customer Type	Age
count	12000.000000	12000.000000	12000.000000	12000.000000
mean	0.543750	0.510333	0.810417	39.205500
std	0.498103	0.499914	0.391988	15.197846
min	0.000000	0.000000	0.000000	7.000000
25%	0.000000	0.000000	1.000000	27.000000
50%	1.000000	1.000000	1.000000	39.000000
75%	1.000000	1.000000	1.000000	51.000000
max	1.000000	1.000000	1.000000	85.000000

	Type of Travel	Class	Flight Distance	Seat comfort
count	12000.000000	12000.000000	12000.000000	12000.000000
mean	0.688333	0.592833	1989.940833	2.861333
std	0.463194	0.621756	1025.567978	1.388500
min	0.000000	0.000000	50.000000	0.000000
25%	0.000000	0.000000	1380.000000	2.000000
50%	1.000000	1.000000	1936.000000	3.000000
75%	1.000000	1.000000	2536.000000	4.000000
max	1.000000	2.000000	6837.000000	5.000000

	Departure/Arrival time convenient	Food and drink	...	Online support

```
              count                      12000.000000    12000.000000  ...    12000.000000
              mean                           3.012083        2.863500  ...        3.511500
              std                            1.524565        1.443101  ...        1.308757
              min                            0.000000        0.000000  ...        1.000000
              25%                            2.000000        2.000000  ...        3.000000
              50%                            3.000000        3.000000  ...        4.000000
              75%                            4.000000        4.000000  ...        5.000000
              max                            5.000000        5.000000  ...        5.000000

                     Ease of Online booking  On-board service  Leg room service  \
              count             12000.000000      12000.000000      12000.000000
              mean                  3.461000          3.470750          3.471917
              std                   1.306887          1.261458          1.297633
              min                   1.000000          1.000000          0.000000
              25%                   2.000000          3.000000          2.000000
              50%                   4.000000          4.000000          4.000000
              75%                   5.000000          4.000000          5.000000
              max                   5.000000          5.000000          5.000000

                     Baggage handling  Checkin service   Cleanliness  Online boarding  \
              count      12000.000000     12000.000000  12000.000000     12000.000000
              mean           3.696833         3.357083      3.706750         3.354833
              std            1.163774         1.267626      1.156734         1.304186
              min            1.000000         1.000000      1.000000         1.000000
              25%            3.000000         3.000000      3.000000         2.000000
              50%            4.000000         3.000000      4.000000         4.000000
              75%            5.000000         4.000000      5.000000         4.000000
              max            5.000000         5.000000      5.000000         5.000000

                     Departure Delay in Minutes  Arrival Delay in Minutes
              count                12000.000000              11958.000000
              mean                    14.845583                 15.078943
              std                     40.105745                 40.136804
              min                      0.000000                  0.000000
              25%                      0.000000                  0.000000
              50%                      0.000000                  0.000000
              75%                     13.000000                 13.000000
              max                   1592.000000               1584.000000

              [8 rows x 23 columns]

In [488]: data.head()

Out[488]:          satisfaction_v2  Gender  Customer Type  Age  Type of Travel  Class  \
              id
              55304                1       0              0   23               1      1
              98174                1       1              1   41               1      0
              114636               1       0              1   45               1      0
```

```
102451                1       1              1   11                0       1
48872                 1       0              1   55                1       0

        Flight Distance  Seat comfort  Departure/Arrival time convenient  \
id
55304              2695             5                                  4
98174               340             1                                  1
114636             2702             1                                  1
102451             1205             4                                  5
48872                83             1                                  1

        Food and drink  ...  Online support  Ease of Online booking  \
id                      ...
55304                4  ...              5                       4
98174                1  ...              5                       5
114636               1  ...              4                       5
102451               4  ...              1                       1
48872                1  ...              3                       3

        On-board service  Leg room service  Baggage handling  Checkin service  \
id
55304                  4                 5                 3                5
98174                  5                 5                 5                5
114636                 5                 5                 5                3
102451                 4                 5                 4                4
48872                  3                 4                 3                3

        Cleanliness  Online boarding  Departure Delay in Minutes  \
id
55304             4                4                          37
98174             5                3                           0
114636            5                4                           0
102451            4                1                           0
48872             3                2                          49

        Arrival Delay in Minutes
id
55304                       39.0
98174                        0.0
114636                       0.0
102451                       0.0
48872                       59.0

[5 rows x 23 columns]
```

In [489]:
```
data.tail()
```

Out[489]:
```
        satisfaction_v2  Gender  Customer Type  Age  Type of Travel  Class  \
id
```

```
67706                1       0              1   45                1       0
97493                0       1              1    9                0       1
50072                0       1              1   38                1       0
129204               0       1              1   39                1       1
86560                0       0              1   33                1       2

        Flight Distance  Seat comfort  Departure/Arrival time convenient  \
id
67706              1891             5                                  5
97493              1809             4                                  5
50072              3703             1                                  5
129204             3934             2                                  2
86560              2220             4                                  5

        Food and drink  ...  Online support  Ease of Online booking  \
id                      ...
67706                5  ...              5                       4
97493                4  ...              1                       1
50072                4  ...              3                       2
129204               2  ...              2                       2
86560                5  ...              4                       4

        On-board service  Leg room service  Baggage handling  Checkin service  \
id
67706                  4                 4                 4                4
97493                  5                 3                 5                5
50072                  2                 2                 1                3
129204                 2                 4                 2                2
86560                  4                 4                 4                2

        Cleanliness  Online boarding  Departure Delay in Minutes  \
id
67706             4                4                           0
97493             5                1                           4
50072             2                3                           0
129204            3                2                           4
86560             4                4                          57

        Arrival Delay in Minutes
id
67706                       0.0
97493                       9.0
50072                       0.0
129204                      0.0
86560                      44.0

[5 rows x 23 columns]
```

```
In [490]: print('Missing values: %i' % data.isnull().sum().sum())

Missing values: 42
```

0.0.3 missing value

```
In [491]: data[data.isnull().sum(1)==1]['satisfaction_v2'].value_counts() ## missing value

Out[491]: 0    22
          1    20
          Name: satisfaction_v2, dtype: int64

In [492]: data['satisfaction_v2'].value_counts()

Out[492]: 1    6525
          0    5475
          Name: satisfaction_v2, dtype: int64

In [493]: data.isnull().sum()

Out[493]: satisfaction_v2                      0
          Gender                               0
          Customer Type                        0
          Age                                  0
          Type of Travel                       0
          Class                                0
          Flight Distance                      0
          Seat comfort                         0
          Departure/Arrival time convenient    0
          Food and drink                       0
          Gate location                        0
          Inflight wifi service                0
          Inflight entertainment               0
          Online support                       0
          Ease of Online booking               0
          On-board service                     0
          Leg room service                     0
          Baggage handling                     0
          Checkin service                      0
          Cleanliness                          0
          Online boarding                      0
          Departure Delay in Minutes           0
          Arrival Delay in Minutes            42
          dtype: int64
```

Arrival Delay in Minutes . Arrival Delay in Minutes=Departure Delay in Minutes

```
In [494]: data.describe().loc[:,['Departure Delay in Minutes','Arrival Delay in Minutes']]
```

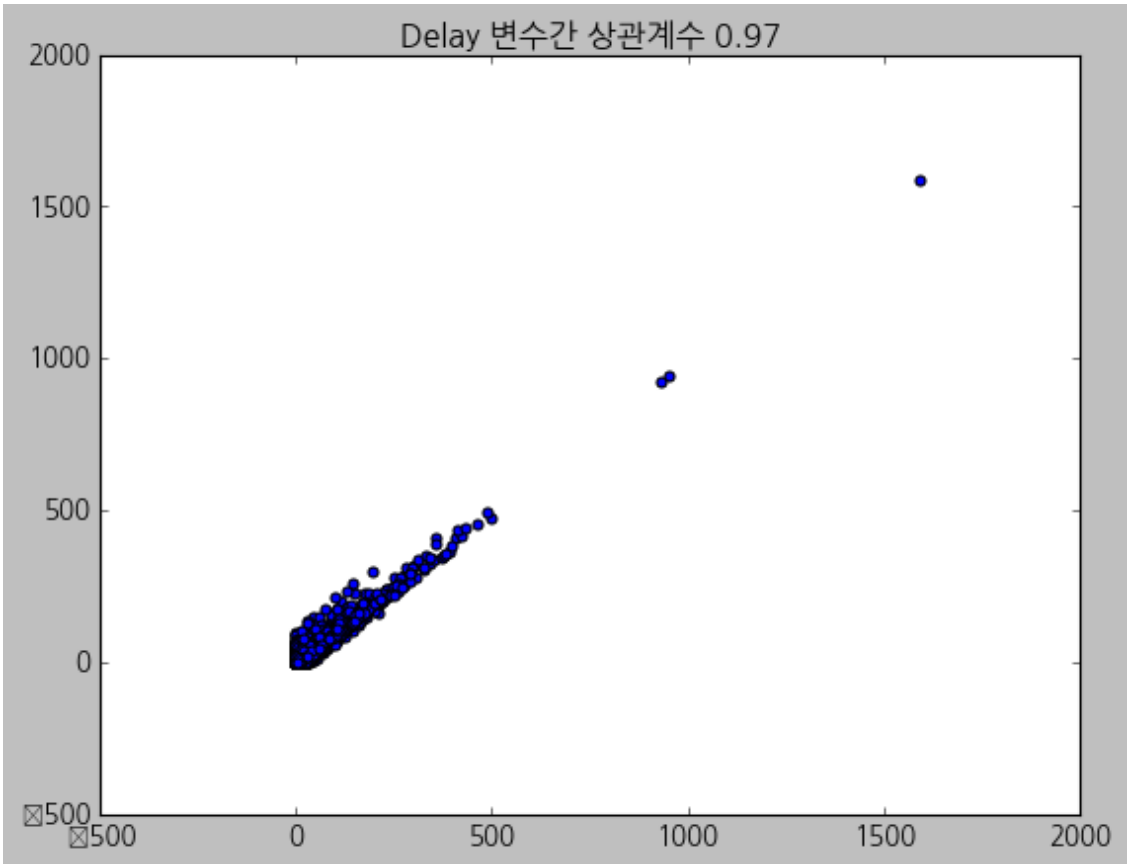
Out[494]:

	Departure Delay in Minutes	Arrival Delay in Minutes
count	12000.000000	11958.000000
mean	14.845583	15.078943
std	40.105745	40.136804
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	13.000000	13.000000
max	1592.000000	1584.000000

In [495]:
```
corr_delay=data.corr().stack()['Departure Delay in Minutes']['Arrival Delay in Minute
```

In [496]:
```
plt.scatter(data['Departure Delay in Minutes'],data['Arrival Delay in Minutes'])
plt.title('Delay   %s'%np.round(corr_delay,2) )
```

Out[496]: Text(0.5, 1.0, 'Delay   0.97')



In [497]:
```
missing_value_index=data[data.isnull().apply(sum,1)!=0].index
```

In [498]:
```
data.loc[missing_value_index,'Arrival Delay in Minutes']=list(data.loc[missing_value
```

In [499]:
```
data.loc[missing_value_index,'Arrival Delay in Minutes']
```

```
Out[499]: id
          40000       0.0
          45562      28.0
          13777       4.0
          60033      40.0
          119008      2.0
          72940      41.0
          100925     30.0
          8522        0.0
          104169    181.0
          43806      15.0
          21763     174.0
          64934      19.0
          7722        0.0
          48612      15.0
          71751      19.0
          70534       0.0
          71265     217.0
          21780      26.0
          121173      3.0
          67589      81.0
          8523        0.0
          112231      4.0
          103297     46.0
          36729     230.0
          114031      1.0
          17846       0.0
          19506      60.0
          27120       0.0
          34036      55.0
          39149       1.0
          41378       0.0
          6694        0.0
          67984       0.0
          98862       0.0
          27638      14.0
          71867     153.0
          96604       2.0
          78260       0.0
          3449        0.0
          2408       58.0
          17767     110.0
          74770       0.0
          Name: Arrival Delay in Minutes, dtype: float64

In [500]: print('Missing values: %i' % data.isnull().sum().sum()) ###missing value

Missing values: 0
```

0.0.4 Class dummy variable

```
In [501]: data['Class'].unique() ###

Out[501]: array([1, 0, 2], dtype=int64)

In [502]: data['Class']=data['Class'].astype('category')

In [503]: Dummies=True
          if Dummies:
              Dummy_Class=pd.get_dummies(data['Class'],prefix="Class").iloc[:,1:3]
              Dummy_Class.columns=['Class_Eco','Class_Eco_plus']###default business
              data=pd.concat([data,Dummy_Class],1)
              data=data.drop('Class',1)
```

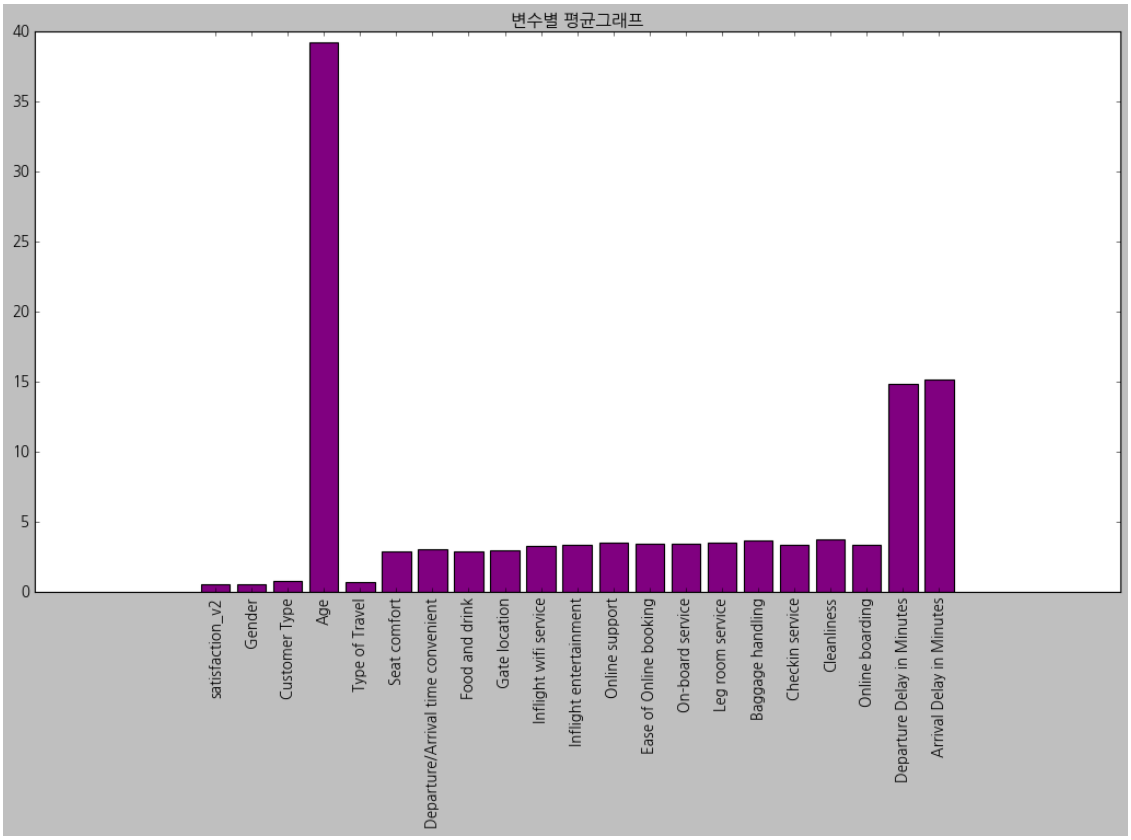
0.0.5 & & ('Flight Distance')

```
In [504]: continuous_data=data.drop(['Class_Eco','Class_Eco_plus','Flight Distance'],1)
          stats=continuous_data.describe()

In [505]: stats = continuous_data.describe()
          print(stats) ###
```

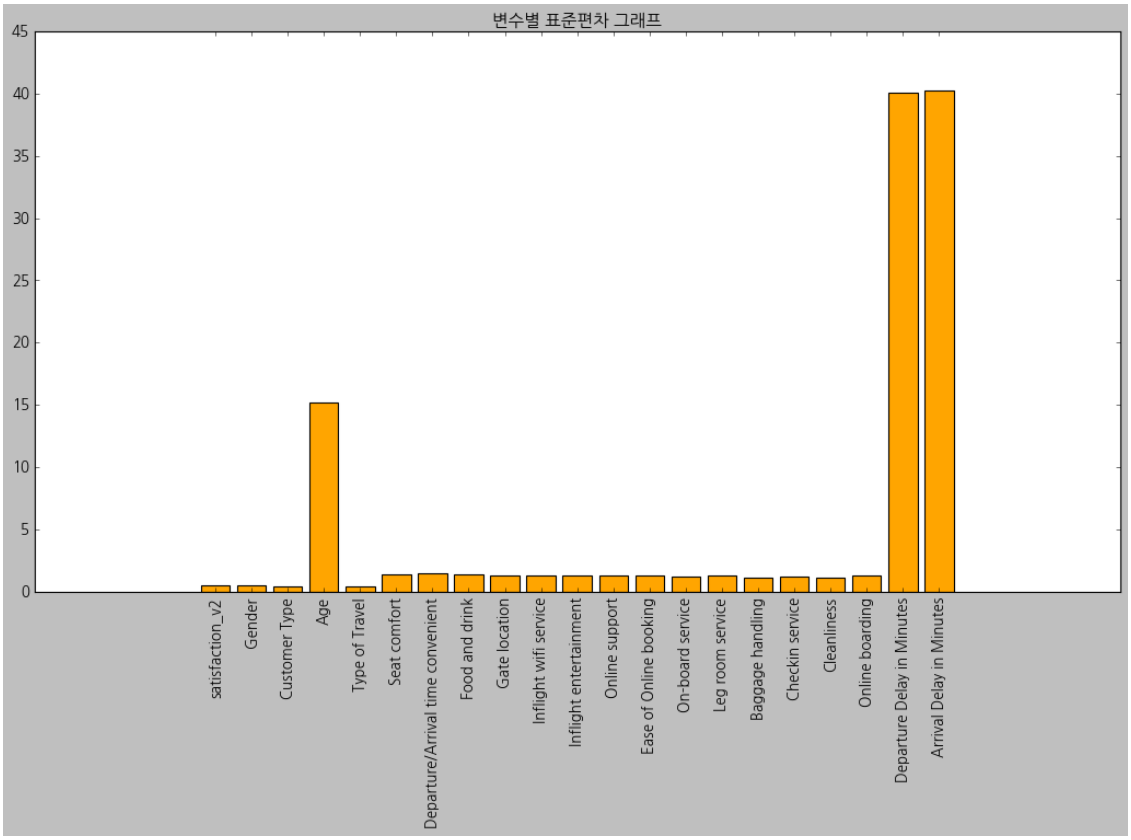
```
       satisfaction_v2        Gender  Customer Type           Age  \
count     12000.000000  12000.000000   12000.000000  12000.000000
mean          0.543750      0.510333       0.810417     39.205500
std           0.498103      0.499914       0.391988     15.197846
min           0.000000      0.000000       0.000000      7.000000
25%           0.000000      0.000000       1.000000     27.000000
50%           1.000000      1.000000       1.000000     39.000000
75%           1.000000      1.000000       1.000000     51.000000
max           1.000000      1.000000       1.000000     85.000000

       Type of Travel  Seat comfort  Departure/Arrival time convenient  \
count    12000.000000  12000.000000                       12000.000000
mean         0.688333      2.861333                           3.012083
std          0.463194      1.388500                           1.524565
min          0.000000      0.000000                           0.000000
25%          0.000000      2.000000                           2.000000
50%          1.000000      3.000000                           3.000000
75%          1.000000      4.000000                           4.000000
max          1.000000      5.000000                           5.000000

       Food and drink  Gate location  Inflight wifi service  ...  \
count    12000.000000   12000.000000           12000.000000  ...
mean         2.863500       3.000667               3.242167  ...
std          1.443101       1.302680               1.318712  ...
min          0.000000       1.000000               0.000000  ...
25%          2.000000       2.000000               2.000000  ...
```

```
50%          3.000000       3.000000              3.000000  ...
75%          4.000000       4.000000              4.000000  ...
max          5.000000       5.000000              5.000000  ...

       Online support  Ease of Online booking  On-board service  \
count    12000.000000            12000.000000      12000.000000
mean         3.511500                3.461000          3.470750
std          1.308757                1.306887          1.261458
min          1.000000                1.000000          1.000000
25%          3.000000                2.000000          3.000000
50%          4.000000                4.000000          4.000000
75%          5.000000                5.000000          4.000000
max          5.000000                5.000000          5.000000

       Leg room service  Baggage handling  Checkin service   Cleanliness  \
count      12000.000000      12000.000000     12000.000000  12000.000000
mean           3.471917          3.696833         3.357083      3.706750
std            1.297633          1.163774         1.267626      1.156734
min            0.000000          1.000000         1.000000      1.000000
25%            2.000000          3.000000         3.000000      3.000000
50%            4.000000          4.000000         3.000000      4.000000
75%            5.000000          5.000000         4.000000      5.000000
max            5.000000          5.000000         5.000000      5.000000

       Online boarding  Departure Delay in Minutes  Arrival Delay in Minutes
count     12000.000000                12000.000000              12000.000000
mean          3.354833                   14.845583                 15.161917
std           1.304186                   40.105745                 40.258029
min           1.000000                    0.000000                  0.000000
25%           2.000000                    0.000000                  0.000000
50%           4.000000                    0.000000                  0.000000
75%           4.000000                   13.000000                 13.000000
max           5.000000                 1592.000000               1584.000000

[8 rows x 21 columns]
```

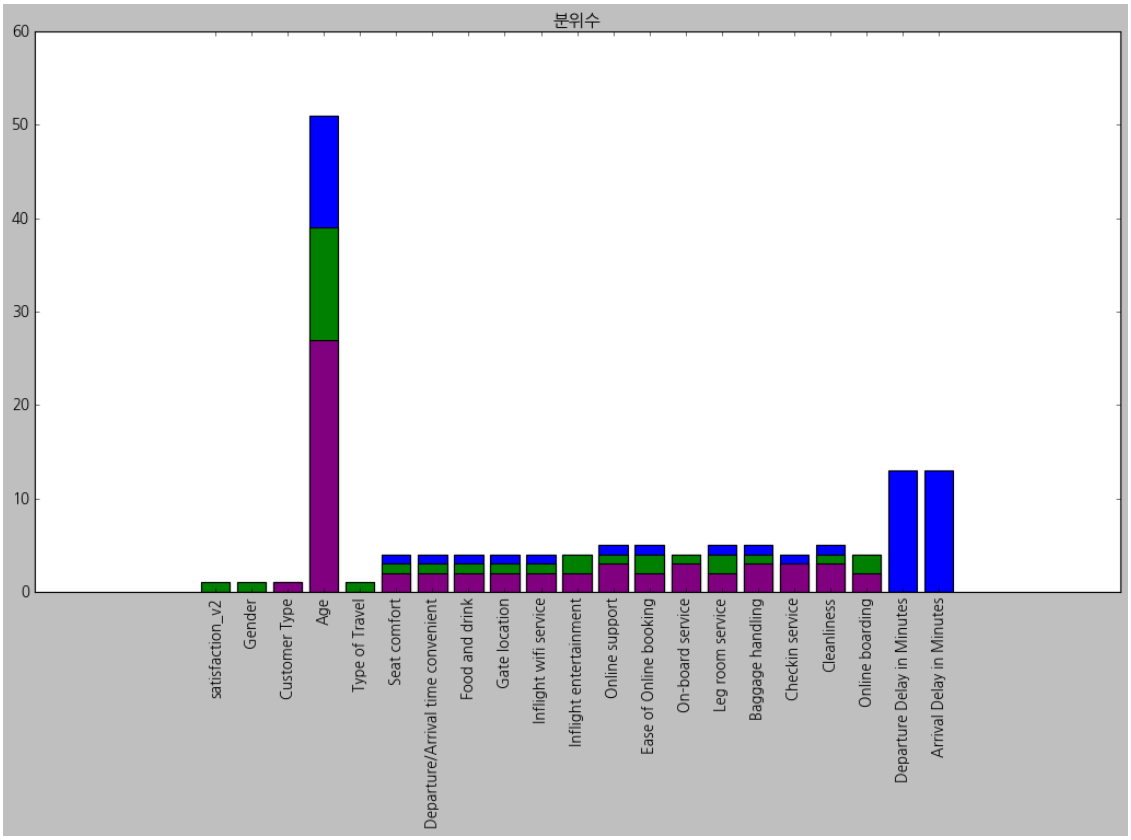
```
In [506]: ###
          fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = continuous_data.columns
          x_pos = np.arange(len(objects))
          ax1 = plt.bar(x_pos, stats.loc['mean'],color="purple" ,alpha=1)
          plt.xticks(x_pos, objects)
          plt.xticks(rotation=90);
          plt.title(' ', size=14)

Out[506]: Text(0.5, 1.0, ' ')
```




```
In [507]: ###
          fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = continuous_data.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[507]: Text(0.5, 1.0, '  ')
```




```
In [508]: #
          fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = continuous_data.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[508]: Text(0.5, 1.0, '')
```



0.0.6 0

```
In [509]: num_zeros = []
          for i in range(0, len(data.columns)):
              num_nonzero = len(data.iloc[:,i].nonzero()[0])
              num_zeros.append(data.shape[0] - num_nonzero)
```

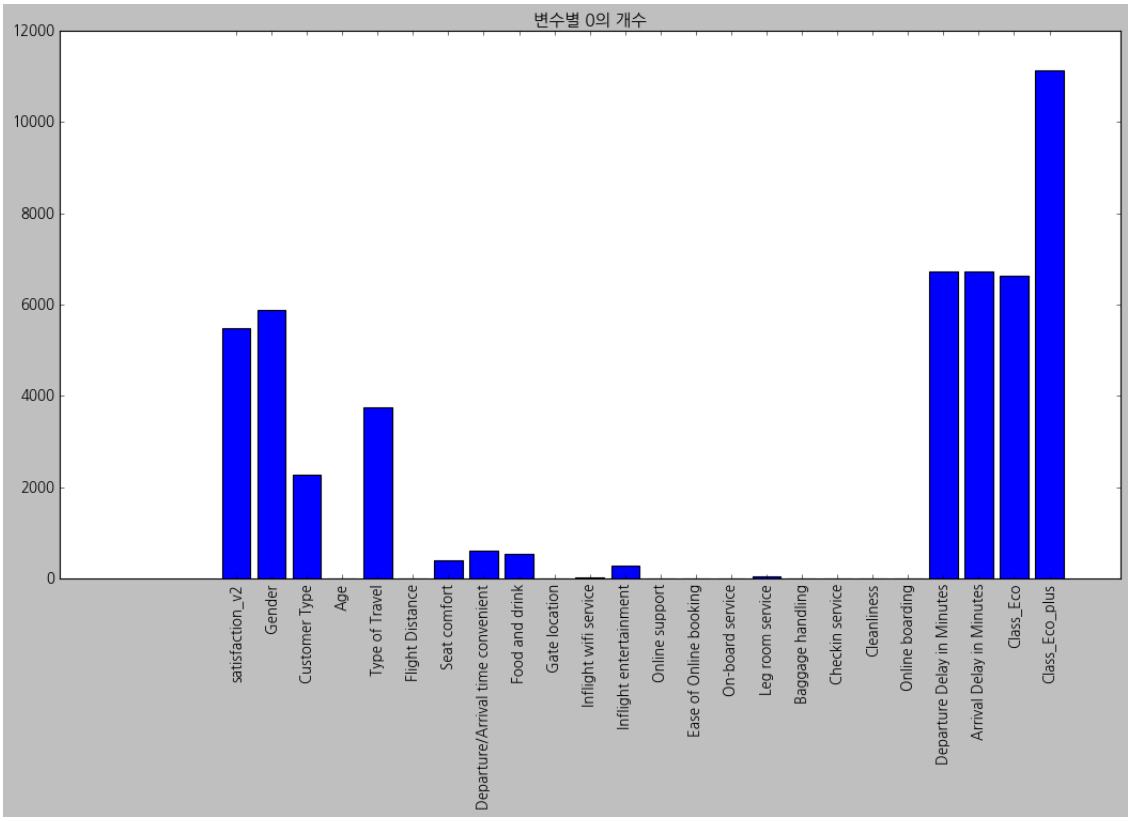
```
C:\Users\jang\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: FutureWarning: Series.nonze
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [510]: num_zeros=pd.Series(num_zeros)
          num_zeros.index=data.columns ##
          print(num_zeros)
```

```
satisfaction_v2                    5475
Gender                             5876
Customer Type                      2275
Age                                   0
Type of Travel                     3740
Flight Distance                       0
Seat comfort                        401
```

```
Departure/Arrival time convenient      603
Food and drink                         529
Gate location                            0
Inflight wifi service                   12
Inflight entertainment                 277
Online support                           0
Ease of Online booking                   0
On-board service                         0
Leg room service                        42
Baggage handling                         0
Checkin service                          0
Cleanliness                              0
Online boarding                          0
Departure Delay in Minutes            6729
Arrival Delay in Minutes              6733
Class_Eco                             6628
Class_Eco_plus                       11129
dtype: int64
```

```
In [511]: # Plot number of zero values for each feature in order.
          fig = plt.figure(figsize=(16,8))
          ax1 = fig.add_subplot(111)
          objects = data.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[511]: Text(0.5, 1.0, ' 0 ')
```



0.0.7

BOXPLOT

```
In [512]: data.describe().T['max']##age,flight distance, departure delay in minutes,arrival de

Out[512]: satisfaction_v2                       1.0
          Gender                                1.0
          Customer Type                         1.0
          Age                                  85.0
          Type of Travel                        1.0
          Flight Distance                    6837.0
          Seat comfort                          5.0
          Departure/Arrival time convenient     5.0
          Food and drink                        5.0
          Gate location                         5.0
          Inflight wifi service                 5.0
          Inflight entertainment                5.0
          Online support                        5.0
          Ease of Online booking                5.0
          On-board service                      5.0
          Leg room service                      5.0
```

```
Baggage handling                      5.0
Checkin service                       5.0
Cleanliness                           5.0
Online boarding                       5.0
Departure Delay in Minutes         1592.0
Arrival Delay in Minutes           1584.0
Class_Eco                             1.0
Class_Eco_plus                        1.0
Name: max, dtype: float64
```

In [513]:
```
continuous_columns=data[['Age','Flight Distance','Departure Delay in Minutes','Arriva
```

In [514]:
```
scaler=preprocessing.StandardScaler()
scaler.fit(data[continuous_columns])
standard_data=pd.DataFrame(scaler.transform(data[continuous_columns]),
                           columns=continuous_columns,index=data.index)
standard_data.describe()
```

```
C:\Users\jang\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:645: DataConversionWarn
  return self.partial_fit(X, y)
C:\Users\jang\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: Data
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[514]:

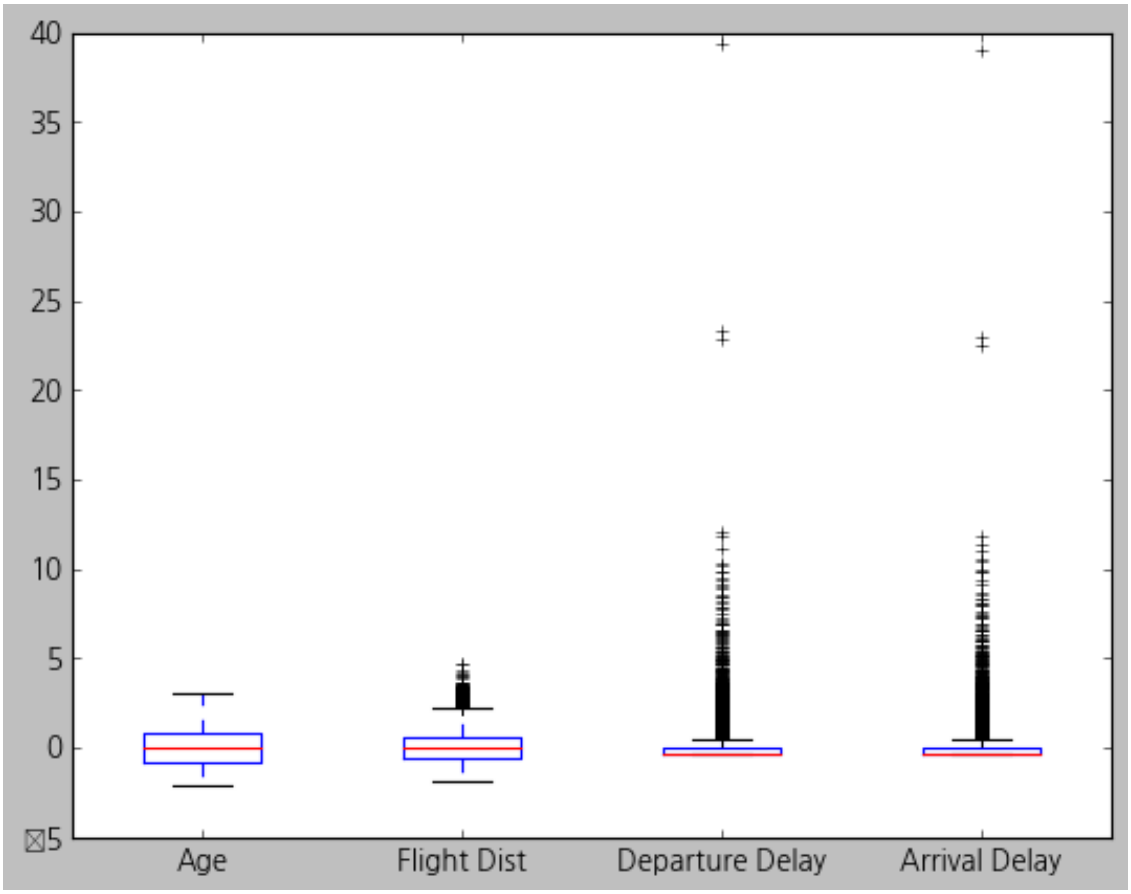
	Age	Flight Distance	Departure Delay in Minutes	\
count	1.200000e+04	1.200000e+04	1.200000e+04	
mean	-4.651834e-17	6.550316e-18	6.162108e-16	
std	1.000042e+00	1.000042e+00	1.000042e+00	
min	-2.119172e+00	-1.891656e+00	-3.701764e-01	
25%	-8.031407e-01	-5.947595e-01	-3.701764e-01	
50%	-1.352222e-02	-5.259825e-02	-3.701764e-01	
75%	7.760963e-01	5.324678e-01	-4.601985e-02	
max	3.013349e+00	4.726416e+00	3.932654e+01	

	Arrival Delay in Minutes
count	1.200000e+04
mean	-1.046478e-16
std	1.000042e+00
min	-3.766341e-01
25%	-3.766341e-01
50%	-3.766341e-01
75%	-5.370374e-02
max	3.897119e+01

In [515]:
```
plt.boxplot((standard_data['Age'],standard_data['Flight Distance'],standard_data['Dep
plt.xticks([1,2,3,4],['Age','Flight Dist','Departure Delay','Arrival Delay'])
```

Out[515]:
```
([<matplotlib.axis.XTick at 0x233cd94ae10>,
  <matplotlib.axis.XTick at 0x233d56d7b38>,
```

```
 <matplotlib.axis.XTick at 0x233ce1edcc0>,
 <matplotlib.axis.XTick at 0x233cd84e3c8>],
<a list of 4 Text xticklabel objects>)
```




```
In [516]: deletes=list(standard_data['Departure Delay in Minutes'][standard_data['Departure De

In [517]: data['Departure Delay in Minutes'][deletes] ####15

Out[517]: id
          63689     951
          8345      933
          73471    1592
          Name: Departure Delay in Minutes, dtype: int64

In [518]: data=data.drop(deletes)
```

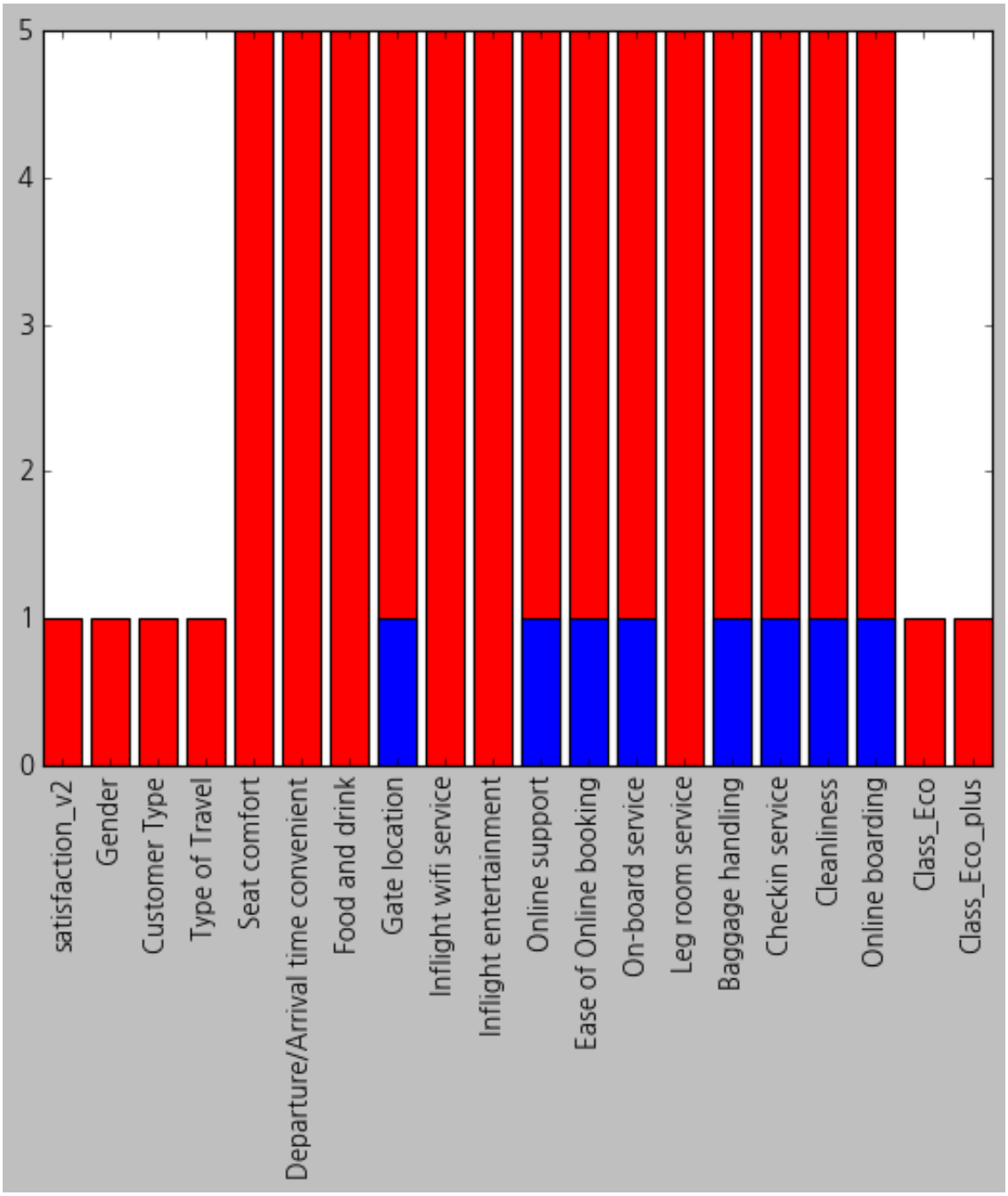
15 .

0.0.8 discrete case

```
In [519]:  discrete_columns=data.drop(continuous_columns,1).columns
```

```
In [520]: plt.bar(discrete_columns,data[discrete_columns].max(),color='red')
          plt.bar(discrete_columns,data[discrete_columns].min(),color='blue')
          plt.xticks(discrete_columns,rotation='vertical')
```

```
Out[520]: ([<matplotlib.axis.XTick at 0x233cdc485c0>,
            <matplotlib.axis.XTick at 0x233cdc502b0>,
            <matplotlib.axis.XTick at 0x233cdc486a0>,
            <matplotlib.axis.XTick at 0x233cd7b6908>,
            <matplotlib.axis.XTick at 0x233cd721390>,
            <matplotlib.axis.XTick at 0x233cd721908>,
            <matplotlib.axis.XTick at 0x233cd721e10>,
            <matplotlib.axis.XTick at 0x233cd70fd30>,
            <matplotlib.axis.XTick at 0x233cd714f28>,
            <matplotlib.axis.XTick at 0x233cd7214a8>,
            <matplotlib.axis.XTick at 0x233cd6f62b0>,
            <matplotlib.axis.XTick at 0x233cd86e160>,
            <matplotlib.axis.XTick at 0x233cd850a20>,
            <matplotlib.axis.XTick at 0x233cd850b70>,
            <matplotlib.axis.XTick at 0x233cdf365c0>,
            <matplotlib.axis.XTick at 0x233cdf36b00>,
            <matplotlib.axis.XTick at 0x233cde75be0>,
            <matplotlib.axis.XTick at 0x233cdf36e48>,
            <matplotlib.axis.XTick at 0x233cdf360f0>,
            <matplotlib.axis.XTick at 0x233cd721208>],
           <a list of 20 Text xticklabel objects>)
```

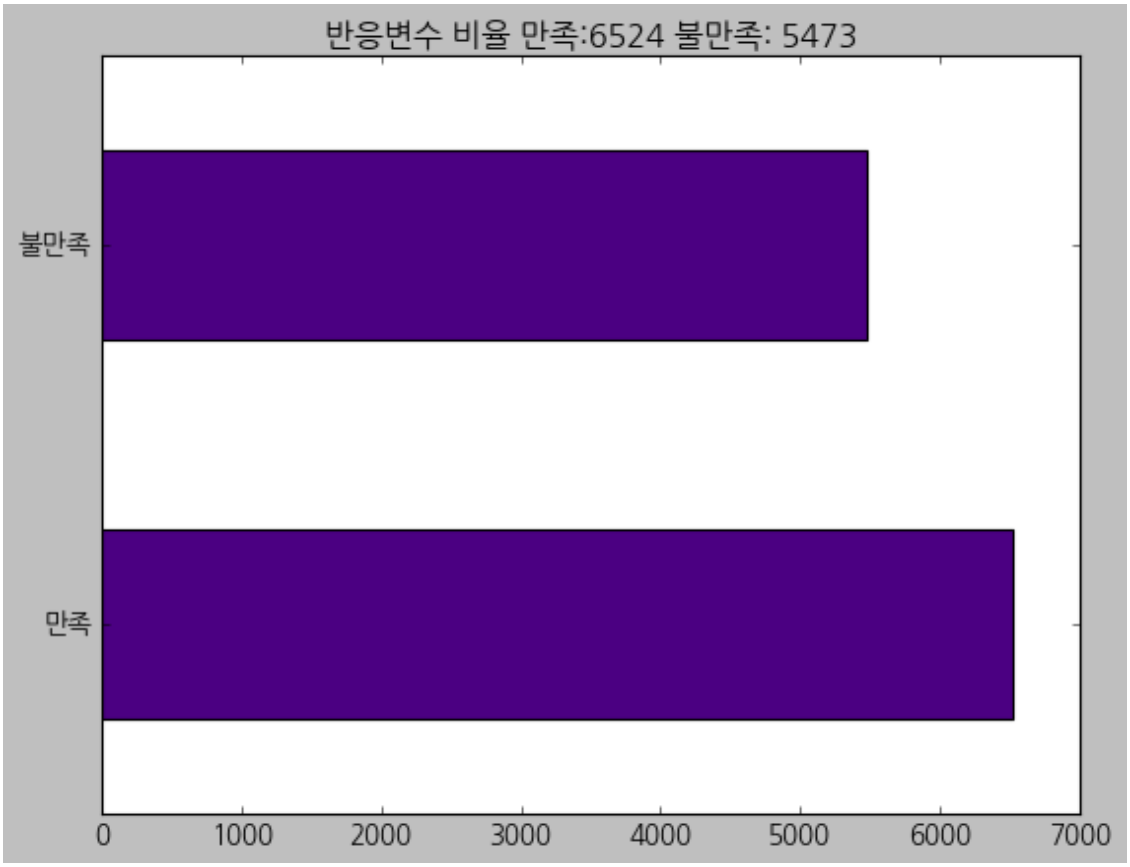



```
In [521]: unique_Dict={}
          for i in discrete_columns:
              unique_Dict[i]=list(np.sort(data[i].unique()))
          unique_Dict
```

```
Out[521]: {'satisfaction_v2': [0, 1],
           'Gender': [0, 1],
           'Customer Type': [0, 1],
           'Type of Travel': [0, 1],
           'Seat comfort': [0, 1, 2, 3, 4, 5],
           'Departure/Arrival time convenient': [0, 1, 2, 3, 4, 5],
           'Food and drink': [0, 1, 2, 3, 4, 5],
           'Gate location': [1, 2, 3, 4, 5],
           'Inflight wifi service': [0, 1, 2, 3, 4, 5],
           'Inflight entertainment': [0, 1, 2, 3, 4, 5],
           'Online support': [1, 2, 3, 4, 5],
           'Ease of Online booking': [1, 2, 3, 4, 5],
           'On-board service': [1, 2, 3, 4, 5],
           'Leg room service': [0, 1, 2, 3, 4, 5],
           'Baggage handling': [1, 2, 3, 4, 5],
           'Checkin service': [1, 2, 3, 4, 5],
           'Cleanliness': [1, 2, 3, 4, 5],
           'Online boarding': [1, 2, 3, 4, 5],
           'Class_Eco': [0, 1],
           'Class_Eco_plus': [0, 1]}
```

0.0.9

```
In [522]: data['satisfaction_v2'].value_counts().plot(kind='barh',color='indigo')
          plt.yticks([0,1],["",""])
          counts=(data['satisfaction_v2'].value_counts().values[0],data['satisfaction_v2'].valu
          plt.title("  :%d : %d"%counts)

Out[522]: Text(0.5, 1.0, '  :6524 : 5473')
```



0.0.10

```
In [523]: corr_sat=abs(data.corr()).stack()['satisfaction_v2'].sort_values()[::-1]
          print(corr_sat)
```

```
satisfaction_v2                   1.000000
Inflight entertainment            0.531620
Ease of Online booking            0.430382
Online support                    0.394540
On-board service                  0.343112
Online boarding                   0.341930
Leg room service                  0.303594
Customer Type                     0.294204
Seat comfort                      0.266109
Class_Eco                         0.265459
Checkin service                   0.253339
Baggage handling                  0.251268
Cleanliness                       0.243887
Inflight wifi service             0.227387
Gender                            0.217215
```

```
Food and drink                          0.139887
Age                                     0.116492
Type of Travel                          0.107411
Arrival Delay in Minutes                0.080946
Departure Delay in Minutes              0.073835
Class_Eco_plus                          0.061691
Flight Distance                         0.037088
Departure/Arrival time convenient       0.019733
Gate location                           0.000095
dtype: float64
```

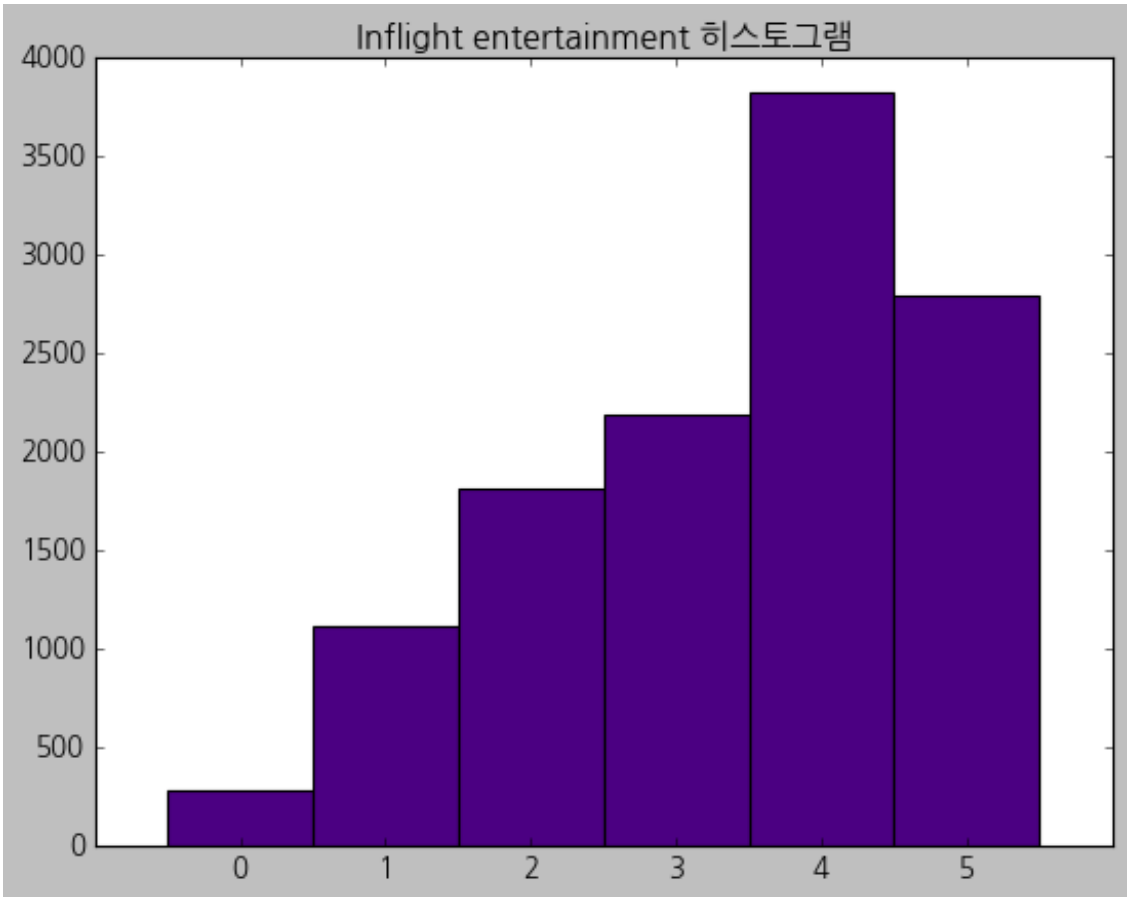
```
In [524]: high_correlation=corr_sat.index[1:11]
```

```
In [525]: data_0=data[data['satisfaction_v2']==0]
          data_1=data[data['satisfaction_v2']==1]
```

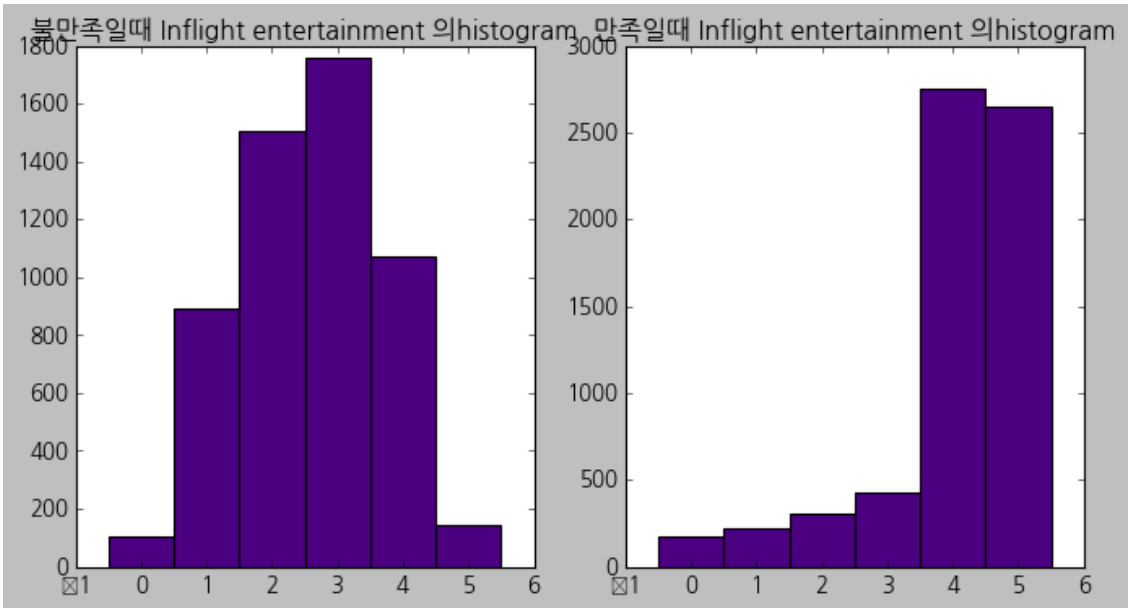
0.0.11 Inflight entertainment

```
In [526]: plt.figure()
          bins=np.arange(-0.5,6.5,1)
          plt.hist(data[high_correlation[0]],bins,align='mid',color='indigo',rwidth=1.0)
          plt.xticks(data[high_correlation[0]].unique())
          plt.title("%s "%high_correlation[0])
```

```
Out[526]: Text(0.5, 1.0, 'Inflight entertainment ')
```




```
In [527]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(data_0[high_correlation[0]],bins,color='indigo')
          axes1.set_title(" %s histogram"%high_correlation[0])
          axes2=plt.subplot(1,2,2)
          axes2.hist(data_1[high_correlation[0]],bins,color='indigo')
          axes2.set_title(" %s histogram"%high_correlation[0])

Out[527]: Text(0.5, 1.0, ' Inflight entertainment histogram')
```



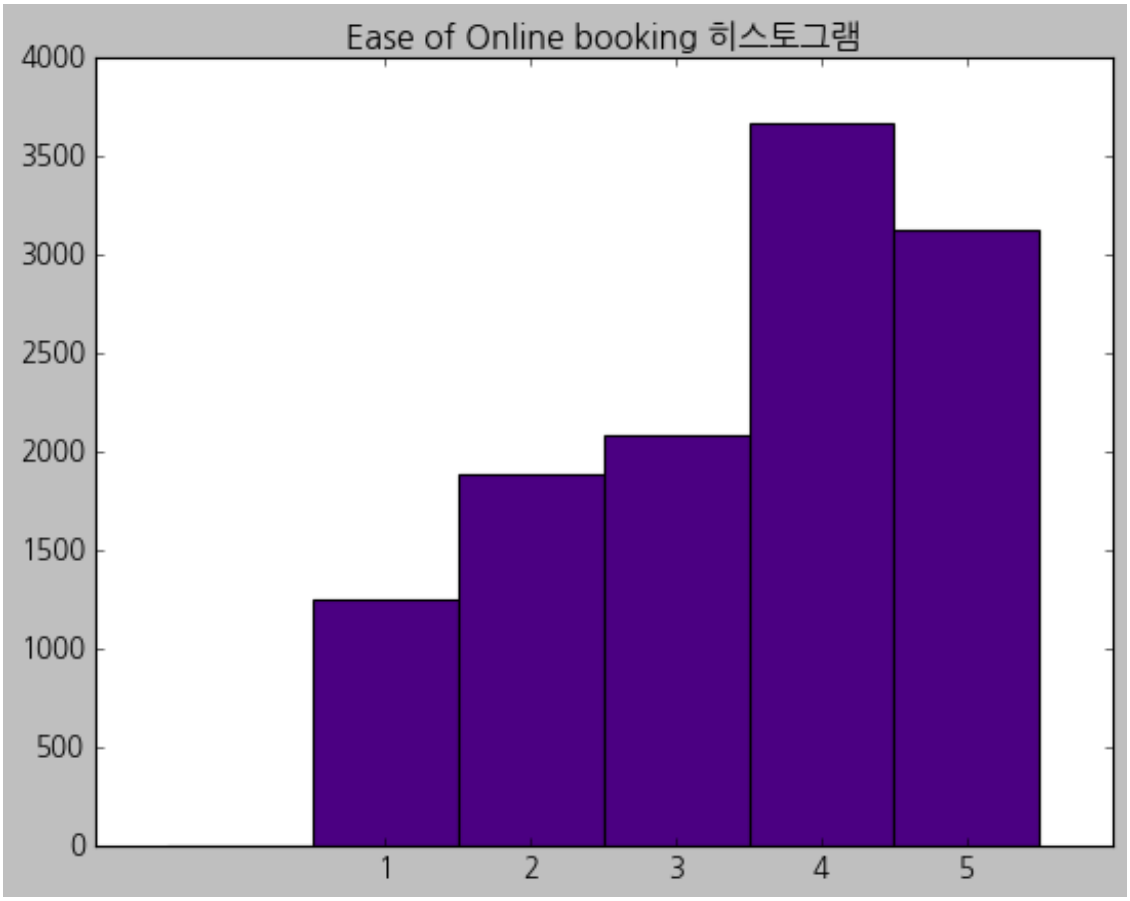

```
In [528]: table=pd.concat([data_0[high_correlation[0]].value_counts(),data_1[high_correlation[
          table.columns=[0,1]
          table=table.fillna(1)
          for i in range(table.shape[0]):
              print("%s "%high_correlation[0],table.index[i],"",table.sum(1).iloc[i],"  ",table
```

```
Inflight entertainment  0  277    1.6634615384615385
Inflight entertainment  1  1111    0.24831460674157305
Inflight entertainment  2  1813    0.20545212765957446
Inflight entertainment  3  2182    0.24188958451906659
Inflight entertainment  4  3823    2.5629077353215286
Inflight entertainment  5  2791    18.248275862068965
```

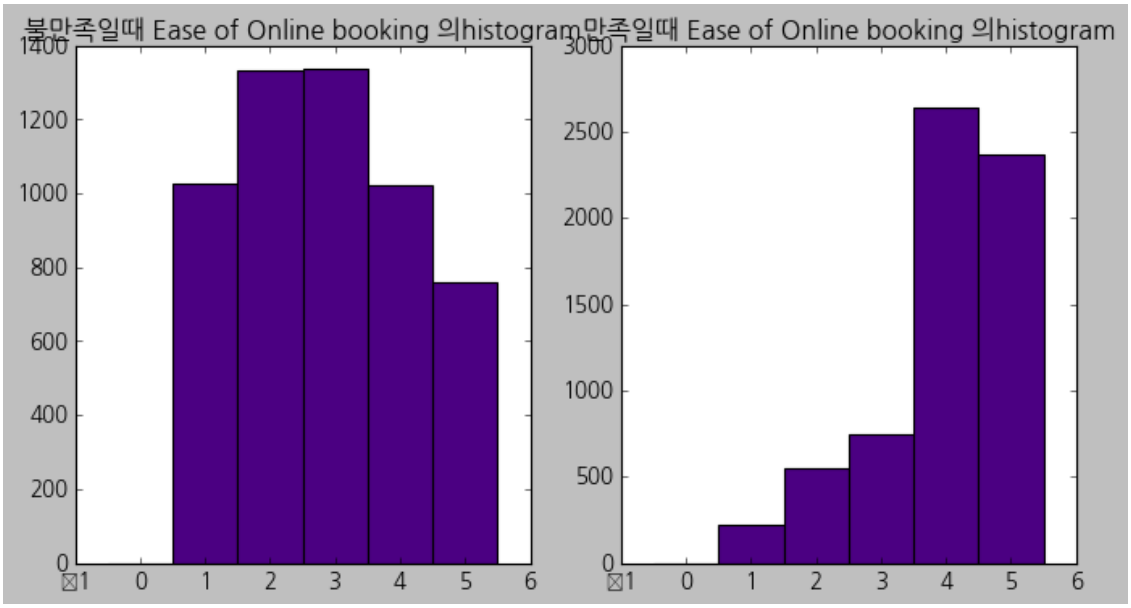
0.0.12 Ease of Online booking

```
In [529]: plt.figure()
          plt.hist(data[high_correlation[1]],bins,align='mid',color='indigo')
          plt.xticks(data[high_correlation[1]].unique())
          plt.title("%s "%high_correlation[1])
```

```
Out[529]: Text(0.5, 1.0, 'Ease of Online booking ')
```




```
In [530]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(data_0[high_correlation[1]],bins,color='indigo')
          axes1.set_title(" %s histogram"%high_correlation[1])
          axes2=plt.subplot(1,2,2)
          axes2.hist(data_1[high_correlation[1]],bins,color='indigo')
          axes2.set_title(" %s histogram"%high_correlation[1])

Out[530]: Text(0.5, 1.0, ' Ease of Online booking histogram')
```



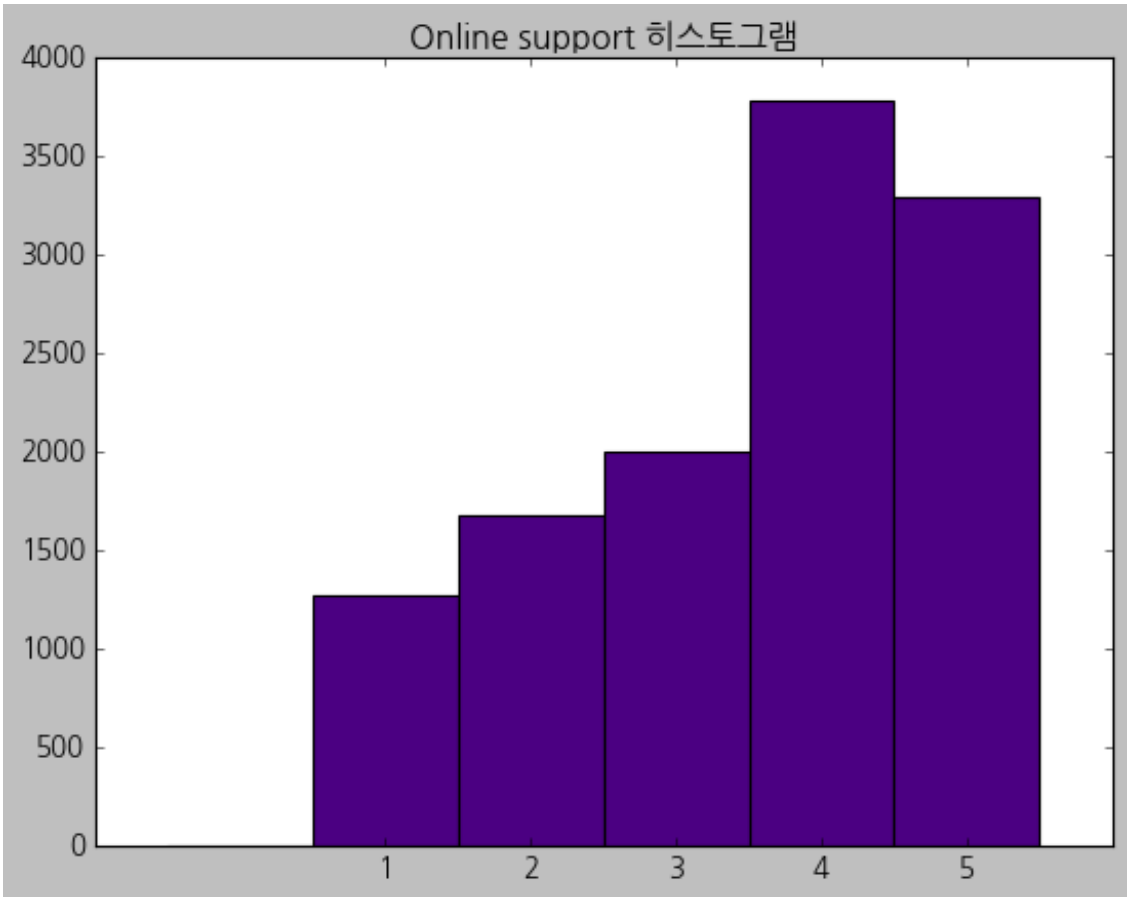

```
In [531]: table=pd.concat([data_0[high_correlation[1]].value_counts(),data_1[high_correlation[
          table.columns=[0,1]
          table=table.fillna(1)
          for i in range(table.shape[0]):
              print("%s "%high_correlation[1],table.index[i],"",table.sum(1).iloc[i],"  ",table
```

```
Ease of Online booking  1  1248    0.21875
Ease of Online booking  2  1884    0.4154770848985725
Ease of Online booking  3  2078    0.5553892215568862
Ease of Online booking  4  3662    2.583170254403131
Ease of Online booking  5  3125    3.111842105263158
```

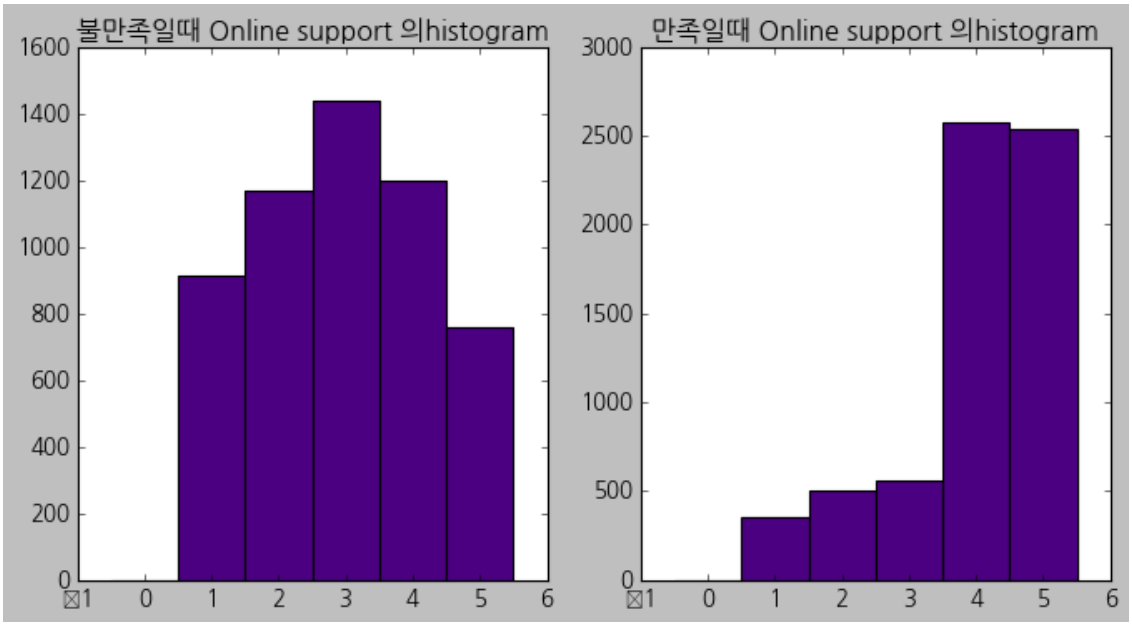
0.0.13 Online support

```
In [532]: plt.figure()
          plt.hist(data[high_correlation[2]],bins,align='mid',color='indigo')
          plt.xticks(data[high_correlation[2]].unique())
          plt.title("%s "%high_correlation[2])
```

```
Out[532]: Text(0.5, 1.0, 'Online support ')
```




```
In [533]: a=plt.figure(figsize=(10,5))
          axes1=plt.subplot(1,2,1)
          axes1.hist(data_0[high_correlation[2]],bins,color='indigo')
          axes1.set_title(" %s histogram"%high_correlation[2])
          axes2=plt.subplot(1,2,2)
          axes2.hist(data_1[high_correlation[2]],bins,color='indigo')
          axes2.set_title(" %s histogram"%high_correlation[2])

Out[533]: Text(0.5, 1.0, ' Online support histogram')
```




```
In [534]: table=pd.concat([data_0[high_correlation[2]].value_counts(),data_1[high_correlation[
          table.columns=[0,1]
          table=table.fillna(1)
          for i in range(table.shape[0]):
              print("%s "%high_correlation[2],table.index[i],"",table.sum(1).iloc[i],"  ",table
```

```
Online support  1  1271    0.38907103825136613
Online support  2  1671    0.43310463121783876
Online support  3  1991    0.3864902506963788
Online support  4  3775    2.1458333333333335
Online support  5  3289    3.3505291005291005
```

0.0.14 MinMax scaler

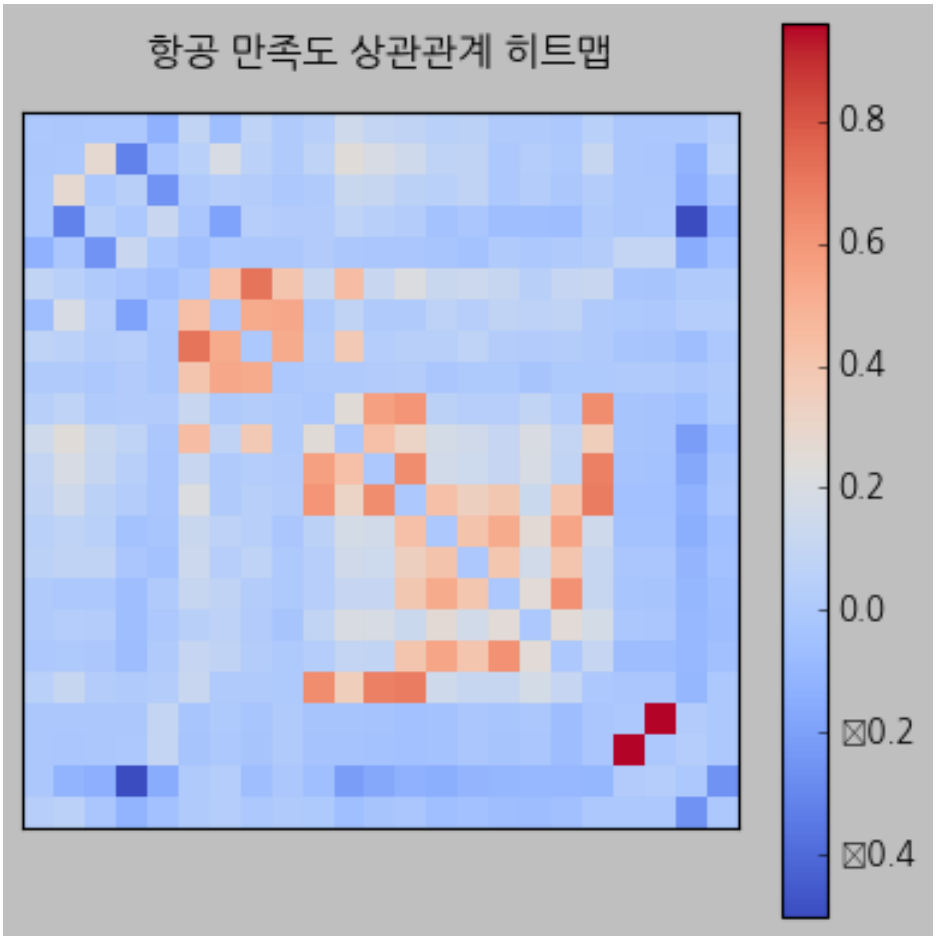
```
In [535]: X=data.drop('satisfaction_v2',1)
          y=data['satisfaction_v2']
```

```
In [536]: min_max=preprocessing.MinMaxScaler()
          min_max.fit(X)
          X=pd.DataFrame(min_max.transform(X),columns=data.drop('satisfaction_v2',1).columns)
```

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

0.0.15

```
In [537]: corr_X=X.corr() ####   0
          for i in range(len(X.columns)):
              corr_X.iloc[i,i]=0

In [538]: plt.matshow(corr_X,cmap='coolwarm')
          plt.colorbar()
          plt.title("   ")
          plt.xticks([])
          plt.yticks([])

Out[538]: ([], <a list of 0 Text yticklabel objects>)
```




```
In [539]: corr_X_stack=corr_X.stack()
          corr_X_stack[corr_X_stack>0.6]

Out[539]: Seat comfort                Food and drink                0.719802
          Food and drink              Seat comfort                  0.719802
```

```
Inflight wifi service       Ease of Online booking        0.606855
                            Online boarding               0.638152
Online support              Ease of Online booking        0.635854
                            Online boarding               0.679442
Ease of Online booking      Inflight wifi service         0.606855
                            Online support                0.635854
                            Online boarding               0.692356
Baggage handling            Cleanliness                   0.623317
Cleanliness                 Baggage handling              0.623317
Online boarding             Inflight wifi service         0.638152
                            Online support                0.679442
                            Ease of Online booking        0.692356
Departure Delay in Minutes  Arrival Delay in Minutes      0.961269
Arrival Delay in Minutes    Departure Delay in Minutes    0.961269
dtype: float64
```

0.1 .

```
In [540]: from statsmodels.stats.outliers_influence import variance_inflation_factor
          vifdata=X.copy()
          vif = pd.DataFrame()
          vif["VIF Factor"] = [variance_inflation_factor(vifdata.values, i) for i in range(vif
          vif["features"] = vifdata.columns
          vif
```

Out[540]:

	VIF Factor	features
0	2.136663	Gender
1	7.139145	Customer Type
2	6.262177	Age
3	4.464305	Type of Travel
4	4.712663	Flight Distance
5	13.124873	Seat comfort
6	8.957879	Departure/Arrival time convenient
7	13.815501	Food and drink
8	5.751200	Gate location
9	13.314338	Inflight wifi service
10	14.031827	Inflight entertainment
11	11.449720	Online support
12	16.466484	Ease of Online booking
13	8.611286	On-board service
14	10.700058	Leg room service
15	11.813318	Baggage handling
16	5.382686	Checkin service
17	12.698733	Cleanliness
18	11.767033	Online boarding
19	15.430323	Departure Delay in Minutes

```
20   15.498198          Arrival Delay in Minutes
21    2.290718                        Class_Eco
22    1.201717                   Class_Eco_plus
```

0.1.1 vif => vif (10)

```
In [541]: 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()].feature
                  delcolumns[vif[vif['VIF Factor']==vif['VIF Factor'].max()].features.values[0
              else:
                  break
```

```
In [542]: delcolumns
```

```
Out[542]: {'Ease of Online booking': 16.46648359885526,
           'Arrival Delay in Minutes': 15.497269810063267,
           'Inflight entertainment': 13.8091447247996,
           'Food and drink': 12.868297117235837,
           'Inflight wifi service': 12.240731028401376,
           'Cleanliness': 12.199287107098018,
           'Leg room service': 10.25044821591339}
```

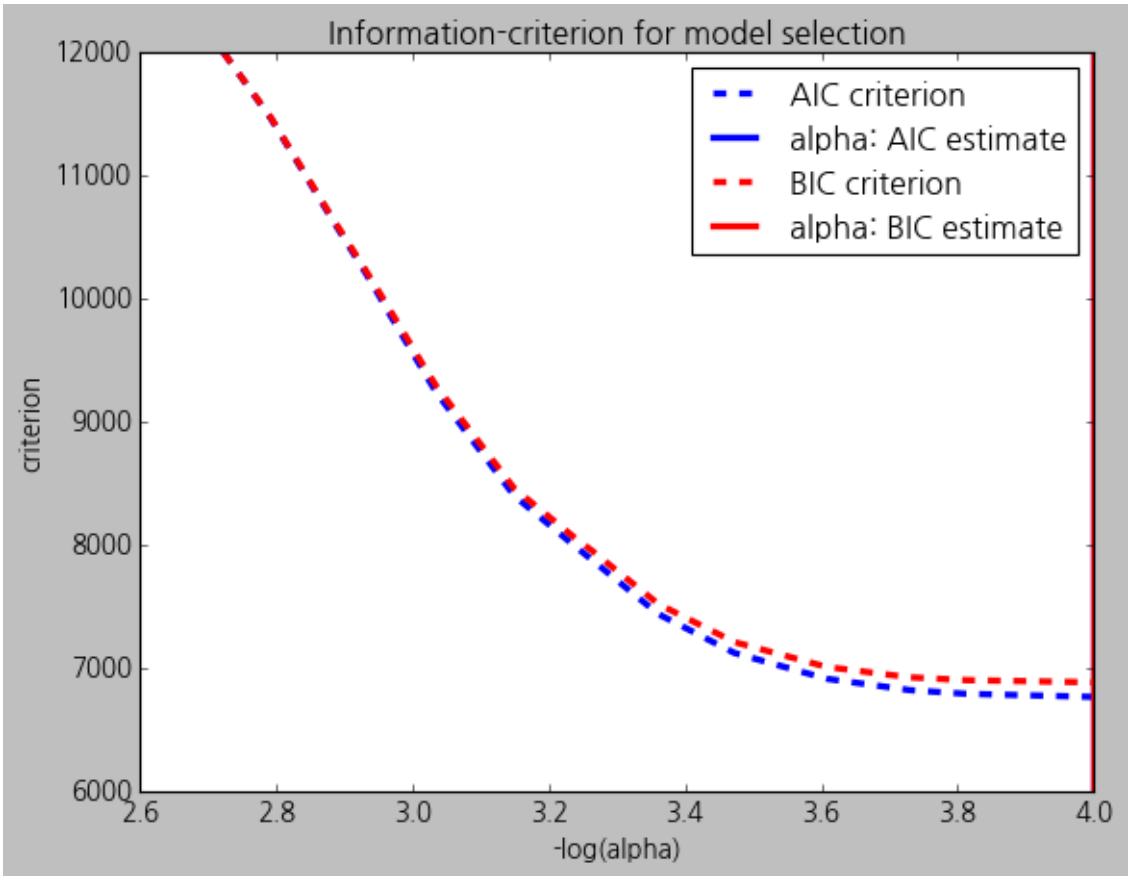
```
In [543]: vif
```

```
Out[543]:     VIF Factor                         features
          0     2.084419                           Gender
          1     6.522945                    Customer Type
          2     6.129958                              Age
          3     4.119284                 Type of Travel
          4     4.535324                Flight Distance
          5     7.173796                   Seat comfort
          6     8.183512  Departure/Arrival time convenient
          7     5.155573                   Gate location
          8     9.349988                  Online support
          9     7.033486                On-board service
          10    8.526102                Baggage handling
          11    4.911955                 Checkin service
          12    8.037106                 Online boarding
          13    1.185062        Departure Delay in Minutes
          14    2.092704                        Class_Eco
          15    1.177107                   Class_Eco_plus
```

```
In [544]: ###   aic
```

In [545]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LassoCV, LassoLarsCV, LassoLarsIC
from sklearn import datasets

EPSILON = 1e-4

X = pd.DataFrame(vifdata.copy(),columns=vifdata.columns)

y = y

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')
```

Out[545]:
```
Text(0.5, 1.0, 'Information-criterion for model selection')
```




```
In [546]: aic_var=vifdata.columns[model_aic.coef_!=0]

In [547]: print(pd.Series(dict(zip(vifdata.columns,abs(model_aic.coef_)))).sort_values(ascendi
```

```
Customer Type                        0.147491
Seat comfort                         0.124562
Departure/Arrival time convenient    0.081363
On-board service                     0.078037
Gender                               0.076553
Online support                       0.065166
Checkin service                      0.061111
Type of Travel                       0.060844
Class_Eco                            0.059732
Online boarding                      0.056422
Baggage handling                     0.043231
Class_Eco_plus                       0.038367
Departure Delay in Minutes           0.021530
Flight Distance                      0.020235
Age                                  0.012840
Gate location                        0.009915
```

```
dtype: float64
```

aic bic

1

1.0.1 imort

```
In [548]: from sklearn.model_selection import train_test_split
          import sklearn
          from sklearn.neighbors import KNeighborsClassifier
          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
          from sklearn.metrics import confusion_matrix
          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.metrics import accuracy_score
          from sklearn.model_selection import GridSearchCV
```

1.0.2

```
In [549]: variable=aic_var
          print(" ")
          print(variable)
          print(" :",len(variable))
```

```
Index(['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Flight Distance',
       'Seat comfort', 'Departure/Arrival time convenient', 'Gate location',
       'Online support', 'On-board service', 'Baggage handling',
       'Checkin service', 'Online boarding', 'Departure Delay in Minutes',
       'Class_Eco', 'Class_Eco_plus'],
      dtype='object')
 : 16
```

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

```
In [550]: X=pd.DataFrame(X,columns=aic_var)
```

```
In [551]: 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)
          X_val,X_test,y_val,y_test=train_test_split(X_val, y_val, test_size=0.5, random_state
          print('Validation Set Shape: ', X_val.shape,y_val.shape)
          print("Test Set shape:",X_test.shape,y_test.shape)
```

```
X and y Input Data:    (11997, 16) (11997,)
Training Set Shape:    (8397, 16) (8397,)
Validation Set Shape:  (1800, 16) (1800,)
Test Set shape: (1800, 16) (1800,)
```

```
In [552]: print("training set   0:", sum(y_train==0),"  1:",sum(y_train==1))
          print("Validation set   0:", sum(y_val==0),"  1:",sum(y_val==1))
          print("Test set   0:", sum(y_test==0),"  1:",sum(y_test==1))
```

```
training set   0: 3811   1: 4586
Validation set   0: 826   1: 974
Test set   0: 836   1: 964
```

1.0.3

```
In [553]: 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=Fa
    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(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
```

1.1 Logistic Regression

```
In [554]: from sklearn import linear_model
          import numpy as np
          import scipy.stats as stat

          class LogisticReg:
              def __init__(self,*args,**kwargs):#,**kwargs):
                  self.model = linear_model.LogisticRegression(*args,**kwargs)#,**args)

              def fit(self,X,y):
                  self.model.fit(X,y)
                  #### Get p-values for the fitted model ####
                  denom = (2.0*(1.0+np.cosh(self.model.decision_function(X))))
                  denom = np.tile(denom,(X.shape[1],1)).T
                  F_ij = np.dot((X/denom).T,X) ## Fisher Information Matrix
                  Cramer_Rao = np.linalg.inv(F_ij) ## Inverse Information Matrix
                  sigma_estimates = np.sqrt(np.diagonal(Cramer_Rao))
                  z_scores = self.model.coef_[0]/sigma_estimates # z-score for eaach model coe
                  p_values = [stat.norm.sf(abs(x))*2 for x in z_scores] ### two tailed test fo

                  self.z_scores = z_scores
                  self.p_values = p_values
                  self.sigma_estimates = sigma_estimates
                  self.F_ij = F_ij

          a=LogisticReg()
          a.fit(X_train,y_train)
```

```
C:\Users\jang\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
```

```
  FutureWarning)
```

```
In [555]: clf_Log = LogisticRegression()
          clf_Log=clf_Log.fit(X_train, y_train)
          y_test_score = clf_Log.decision_function(X_test)
          y_score = clf_Log.decision_function(X_val)
          y_pred_Log = clf_Log.predict(X_val)
          pd.DataFrame({'Columns':X_train.columns,'Coefficients':np.hstack(clf_Log.coef_),"p_Va
```

```
C:\Users\jang\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning:
  FutureWarning)
```

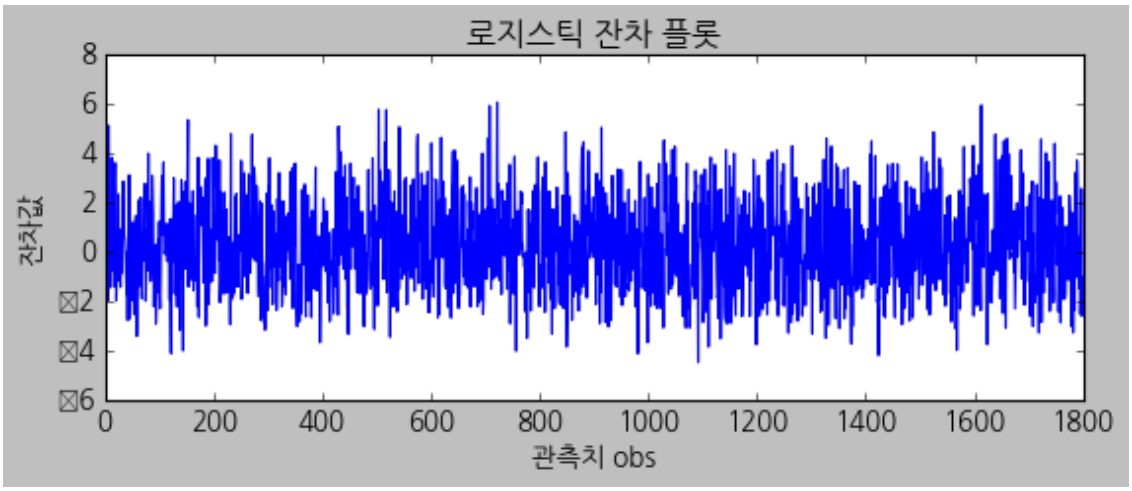
Out[555]:

	Columns	Coefficients	p_Value
0	Gender	0.552474	0.000
1	Customer Type	0.922497	0.000
2	Age	-0.033884	0.298
3	Type of Travel	0.439030	0.000
4	Flight Distance	-0.113213	0.001
5	Seat comfort	0.793209	0.000
6	Departure/Arrival time convenient	-0.510503	0.000
7	Gate location	-0.032541	0.362
8	Online support	0.416263	0.000
9	On-board service	0.539116	0.000
10	Baggage handling	0.258404	0.000
11	Checkin service	0.439569	0.000
12	Online boarding	0.387884	0.000
13	Departure Delay in Minutes	-0.145883	0.000
14	Class_Eco	-0.377584	0.000
15	Class_Eco_plus	-0.219845	0.000

```
In [556]: residual=y_test-y_test_score
```

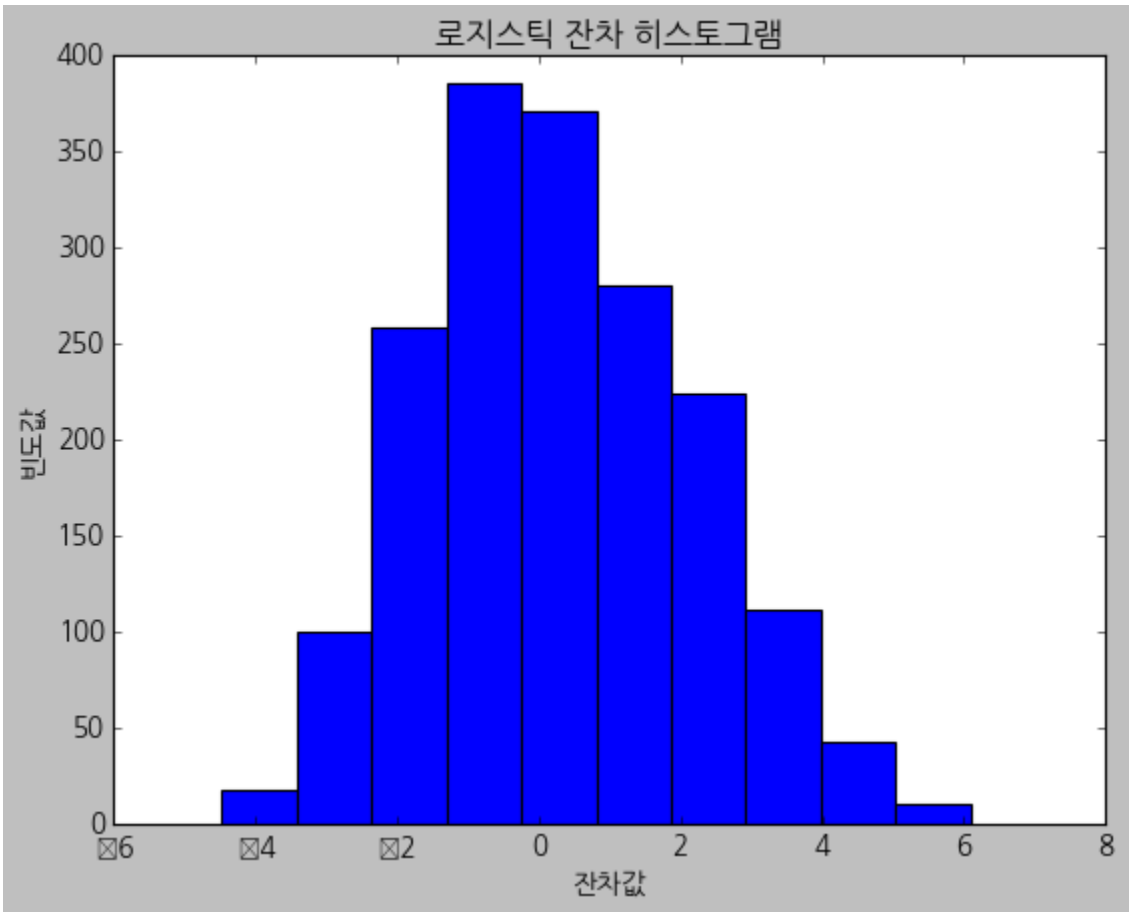
```
In [557]: plt.figure()
          plt.subplot(2,1,1)
          plt.plot(range(len(residual)),residual)
          plt.title("  ")
          plt.xlabel(" obs")
          plt.ylabel("")
```

```
Out[557]: Text(0, 0.5, '')
```



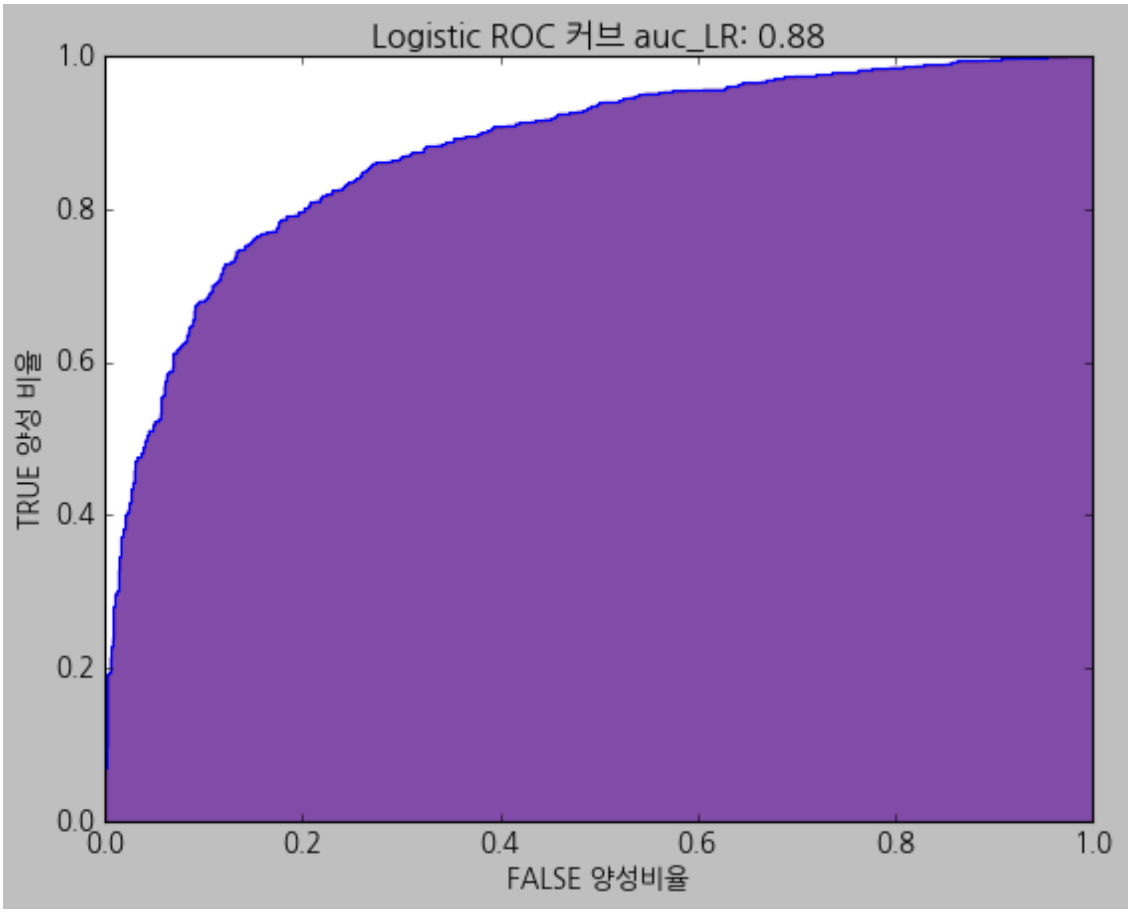

```
In [558]: plt.hist(residual)
          plt.title("  ")###
          plt.xlabel("")
          plt.ylabel("")
```

```
Out[558]: Text(0, 0.5, '')
```



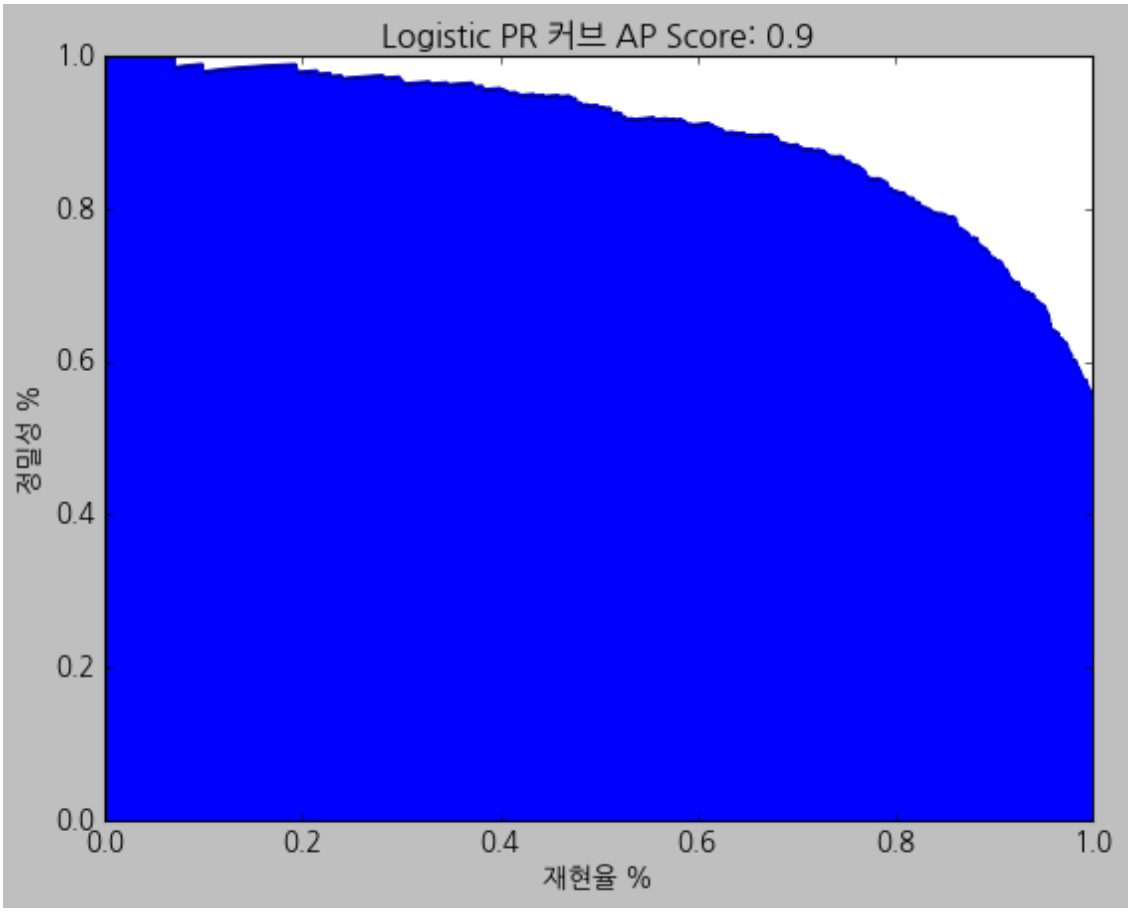
```
In [559]: fpr, tpr, thresholds = roc_curve(y_val,y_score)
          auc_LR=np.round(roc_auc_score(y_val,y_score),2)
          plt.plot(fpr,tpr)
          plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
          plt.title("Logistic ROC  auc_LR: %s" %auc_LR)
          plt.xlabel("FALSE ")
          plt.ylabel("TRUE  ")
```

```
Out[559]: Text(0, 0.5, 'TRUE  ')
```




```
In [560]: precision, recall, thresholds = precision_recall_curve(y_val, y_score)
          ap_score_Log=np.round(average_precision_score(y_val, y_score),2)
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title("Logistic PR  AP Score: %s"%ap_score_Log)
          plt.xlabel(" %")
          plt.ylabel(" %")
```

Out[560]: Text(0, 0.5, ' %')



In [561]: calc_lift(X_train,y_train,clf_Log,bins=10)

Out[561]:

	(%)	LIFT	%	\		
1	840	97.976190	1.793951	823	840	0.98
2	840	95.000000	1.739457	798	1680	0.96
3	839	92.133492	1.686971	773	2519	0.95
4	840	84.523810	1.547637	710	3359	0.92
5	839	64.600715	1.182844	542	4198	0.87
6	840	42.142857	0.771639	354	5038	0.79
7	840	31.785714	0.581999	267	5878	0.73
8	839	18.831943	0.344814	158	6717	0.66
9	840	11.785714	0.215797	99	7557	0.60
10	840	7.380952	0.135146	62	8397	0.55

lift (%)

```
1             823     1.793951
2            1621     1.766704
3            2394     1.740148
4            3104     1.692006
5            3646     1.590246
6            4000     1.453757
7            4267     1.329178
8            4425     1.206224
9            4524     1.096133
10           4586     1.000000
```

```
In [562]: calc_lift(X_val,y_val,clf_Log,bins=10)
```

```
Out[562]:           (%)    LIFT               %  \

          1         180     98.888889  1.827515            178         180         0.99
          2         180     93.888889  1.735113            169         360         0.96
          3         180     85.000000  1.570842            153         540         0.93
          4         180     81.111111  1.498973            146         720         0.90
          5         180     61.111111  1.129363            110         900         0.84
          6         180     47.222222  0.872690             85        1080         0.78
          7         180     28.888889  0.533881             52        1260         0.71
          8         180     21.666667  0.400411             39        1440         0.65
          9         180     15.555556  0.287474             28        1620         0.59
          10        180      7.777778  0.143737             14        1800         0.54

                    lift (%)

          1             178     1.827515
          2             347     1.781314
          3             500     1.711157
          4             646     1.658111
          5             756     1.552361
          6             841     1.439083
          7             893     1.309768
          8             932     1.196099
          9             960     1.095140
          10            974     1.000000
```

1.1.1 decision tree classifer

```
In [563]: ###
          from sklearn.tree import DecisionTreeClassifier


          clf_DT = DecisionTreeClassifier()
          parameters={'max_depth':[3,5,7]}
          clf_DT=GridSearchCV(clf_DT,parameters,cv=5)
```

```
clf_DT.fit(X_train,y_train)
print(clf_DT.best_params_)
clf_DT=clf_DT.best_estimator_


y_pred_DT = clf_DT.predict(X_val)
```

```
{'max_depth': 7}
```

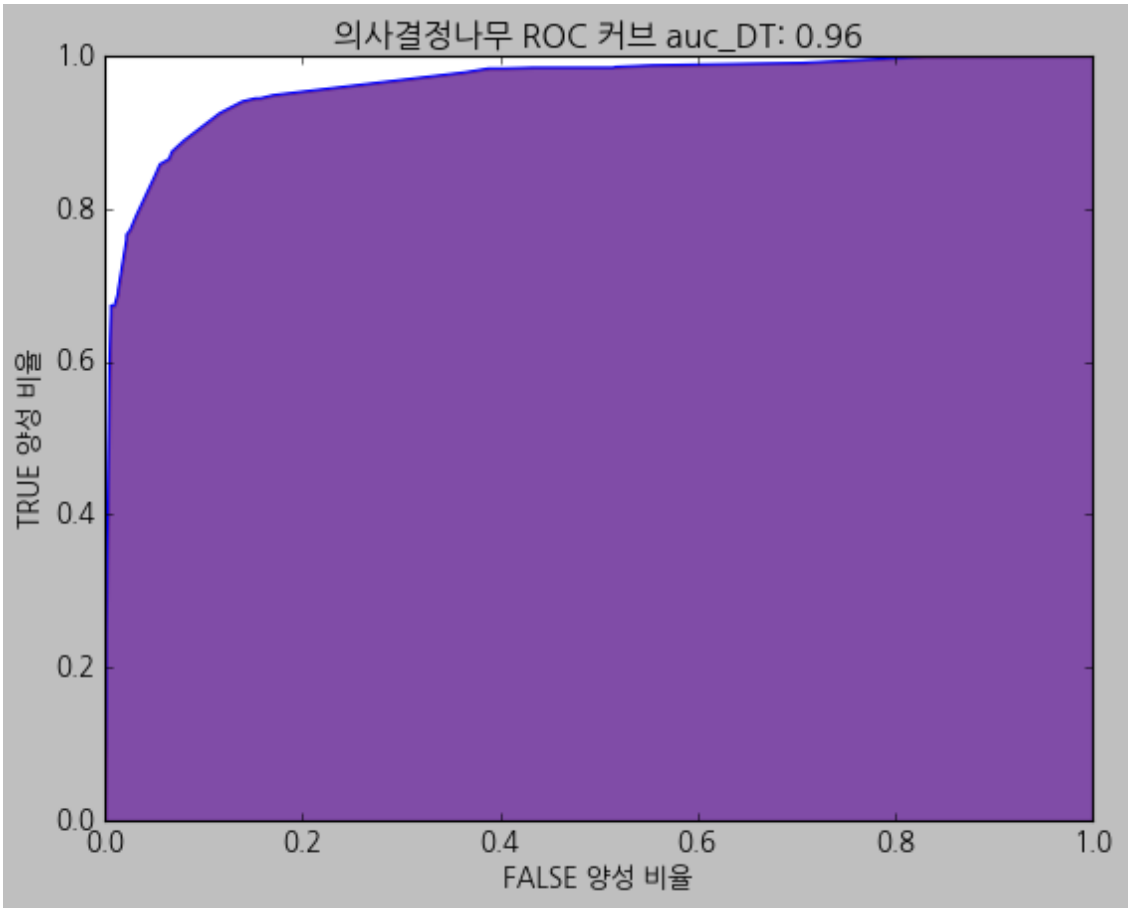
1.1.2 max deplth=7

```
In [564]: from sklearn.externals.six import StringIO
          from IPython.display import Image
          from sklearn.tree import export_graphviz
          import pydotplus
          dot_data = StringIO()
          export_graphviz(clf_DT, out_file=dot_data,
                          filled=True, rounded=True,
                          special_characters=True)
          graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
          Image(graph.create_png())
```

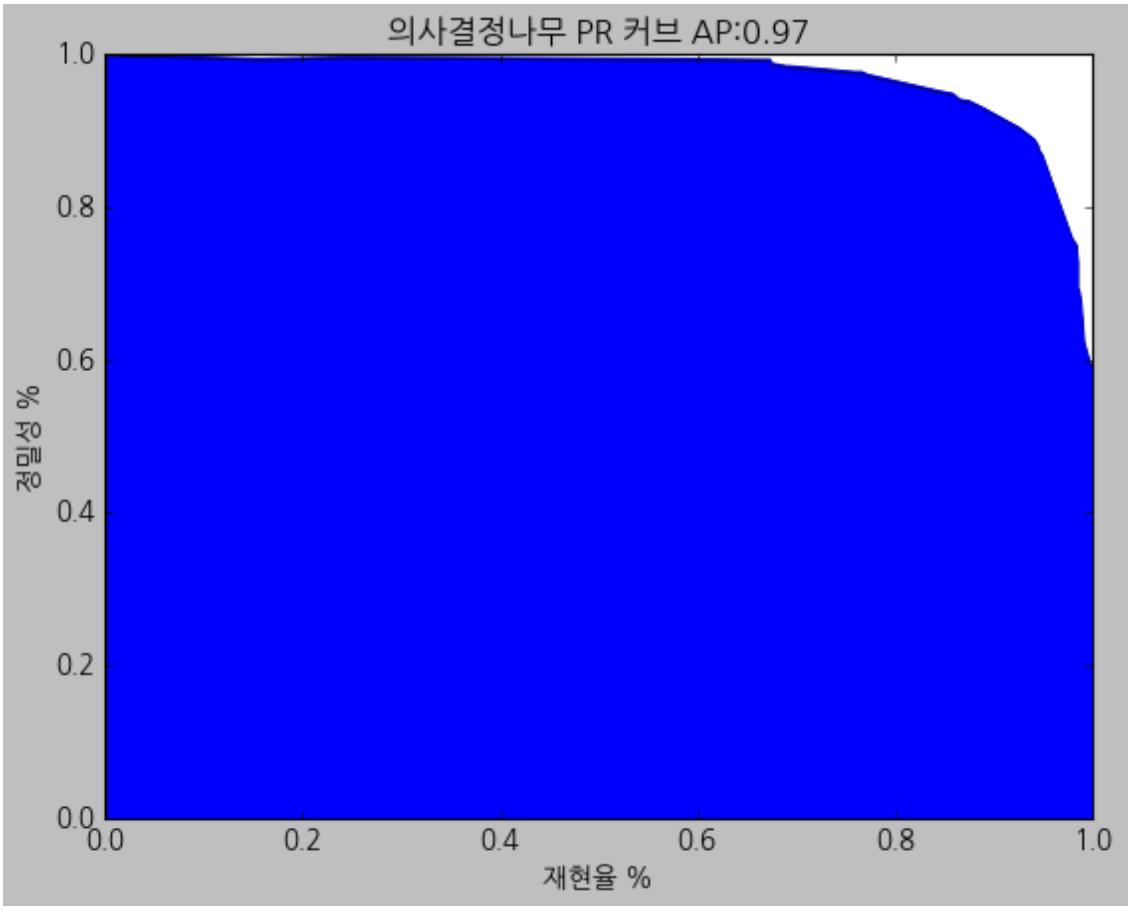
Out[564]:

```
In [565]: y_score = clf_DT.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          auc_DT=np.round(roc_auc_score(y_val,y_score),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(" ROC  auc_DT: %s"%auc_DT)
          plt.xlabel("FALSE  ")
          plt.ylabel("TRUE  ")
```

```
Out[565]: Text(0, 0.5, 'TRUE  ')
```




```
In [566]: y_score = clf_DT.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          precision, recall, thresholds = precision_recall_curve(y_val, y_score)
          ap_score_DT=np.round(average_precision_score(y_val, y_score),2)
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title(" PR  AP:%s"%ap_score_DT)
          plt.xlabel(" %")
          plt.ylabel(" %")

Out[566]: Text(0, 0.5, ' %')
```



1.2 KNN

```
In [567]: clf_KNN=KNeighborsClassifier()
          parameters={'n_neighbors':[5],'weights':['uniform','distance']}
          clf_KNN=GridSearchCV(clf_KNN,parameters,cv=5)
          clf_KNN.fit(X_train,y_train)
          print(clf_KNN.best_params_)
          clf_KNN=clf_KNN.best_estimator_


          y_pred_KNN = clf_KNN.predict(X_val)
```

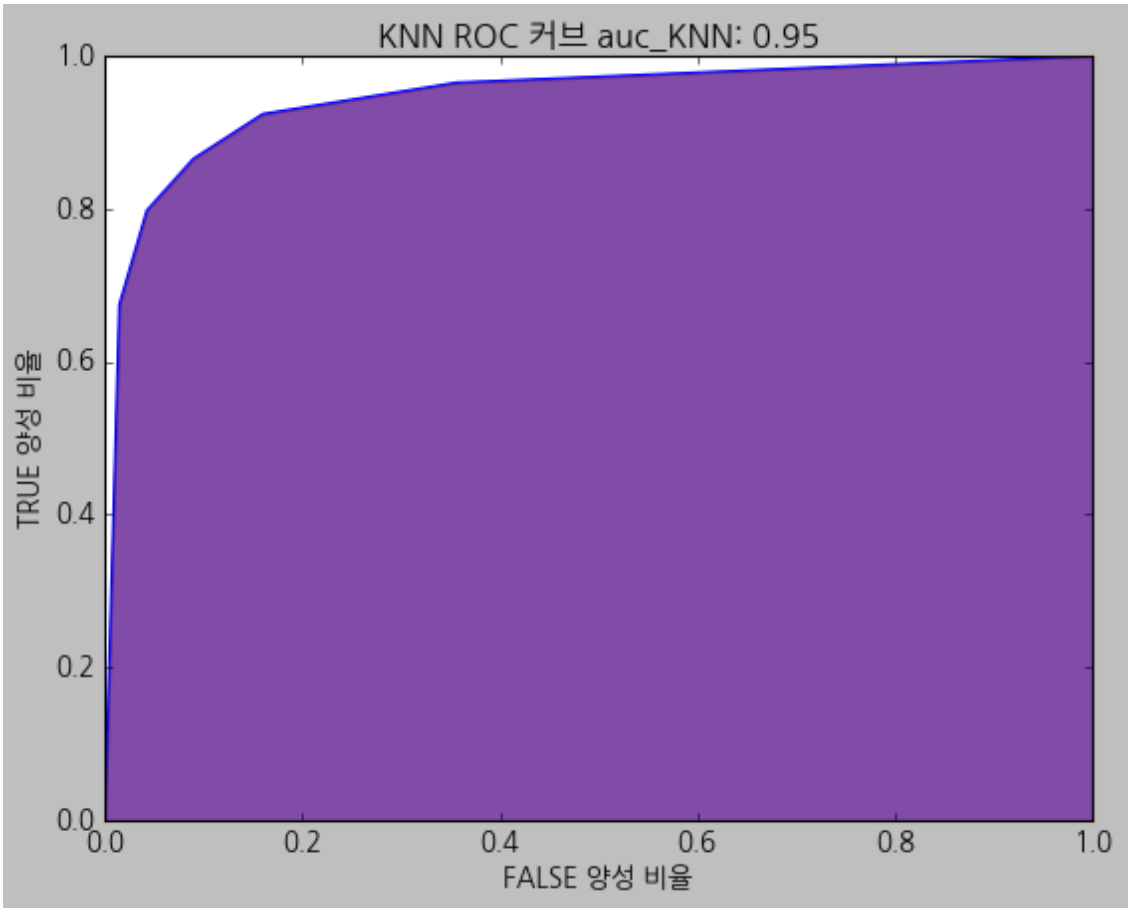
```
{'n_neighbors': 5, 'weights': 'uniform'}
```

1.2.1 uniform

```
In [568]: y_score = clf_KNN.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
```

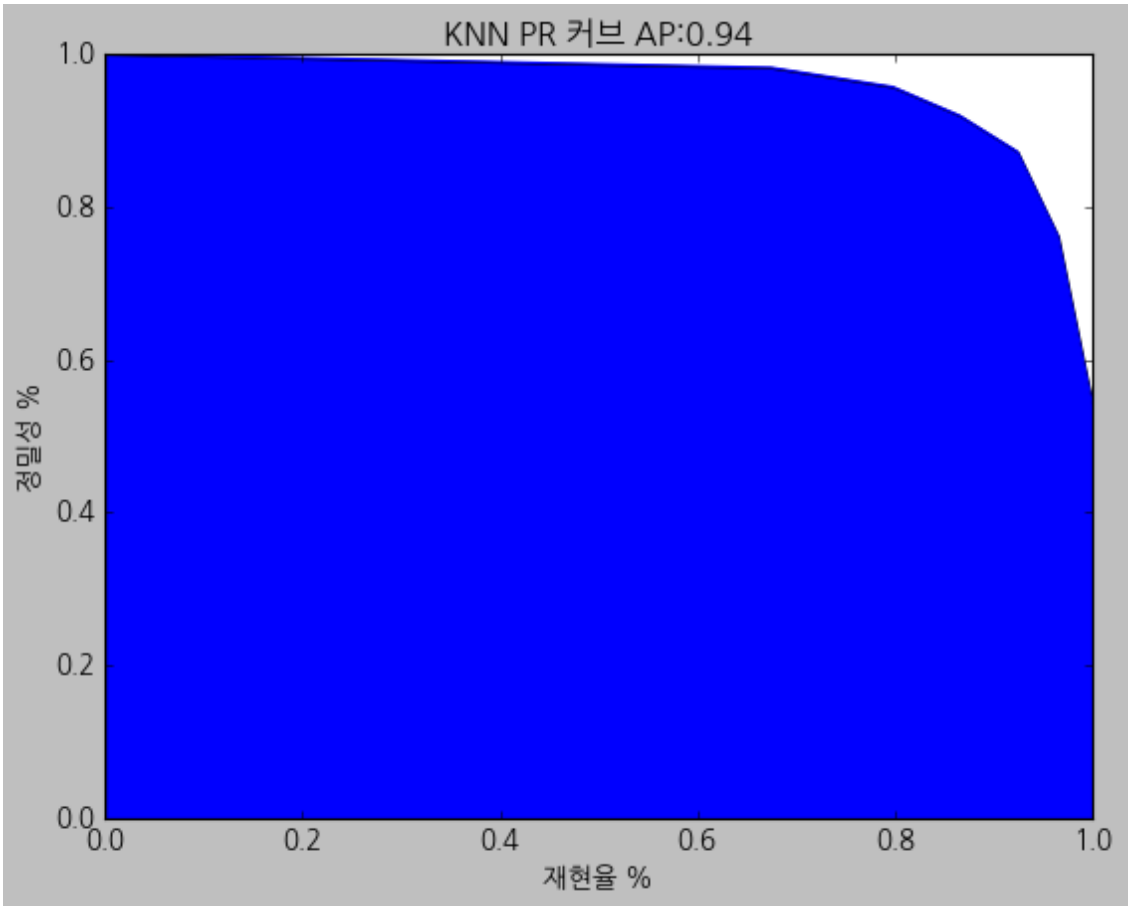
```
auc_KNN=np.round(roc_auc_score(y_val,y_score),2)
fpr, tpr, thresholds = roc_curve(y_val,y_score)
plt.plot(fpr,tpr)
plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
plt.title("KNN ROC  auc_KNN: %s"%auc_KNN)
plt.xlabel("FALSE  ")
plt.ylabel("TRUE  ")
```

Out[568]: Text(0, 0.5, 'TRUE ')



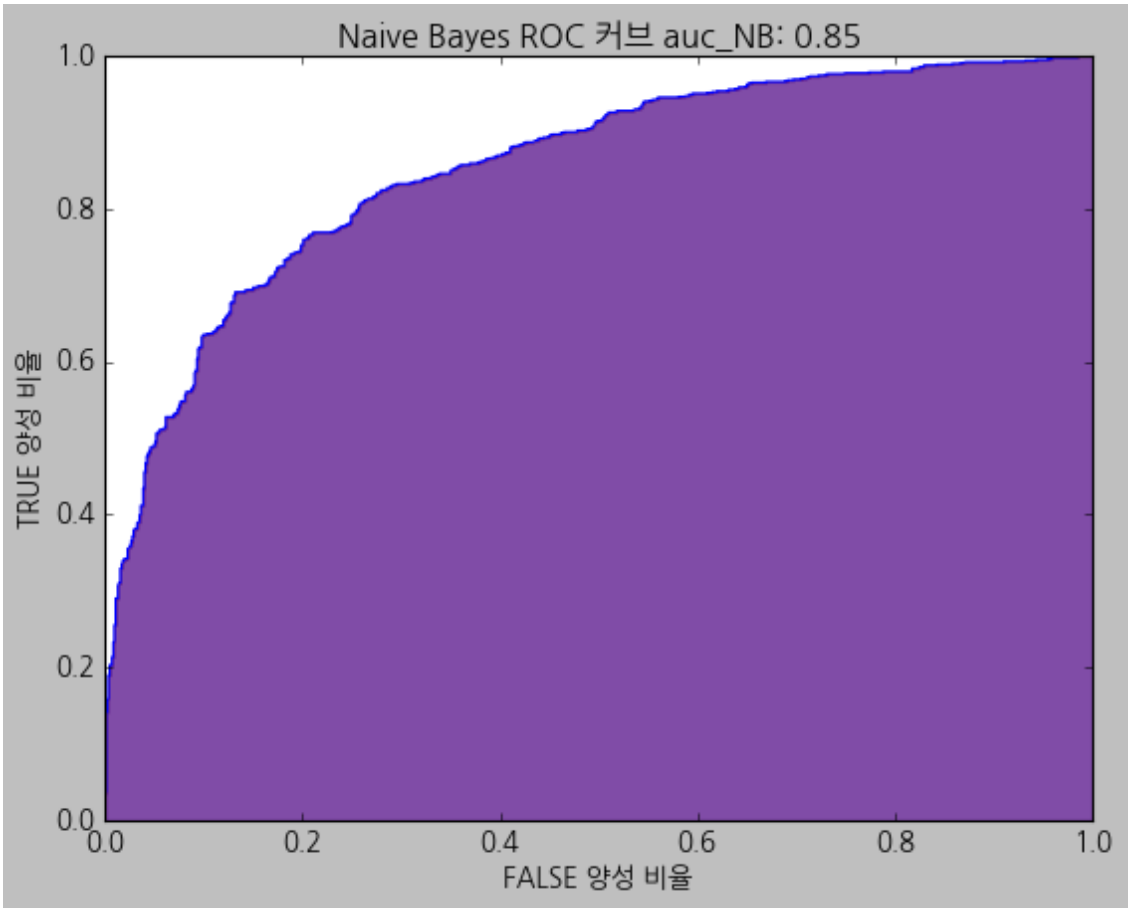
In [569]:
```
y_score = clf_KNN.predict_proba(X_val)
y_score=np.array(pd.DataFrame(y_score)[1])
precision, recall, thresholds = precision_recall_curve(y_val, y_score)
ap_score_KNN=np.round(average_precision_score(y_val, y_score),2)
plt.plot(recall,precision)
plt.fill_between(recall,precision)
plt.title("KNN PR  AP:%s"%ap_score_KNN)
plt.xlabel(" %")
plt.ylabel(" %")
```

```
Out[569]: Text(0, 0.5, ' %')
```

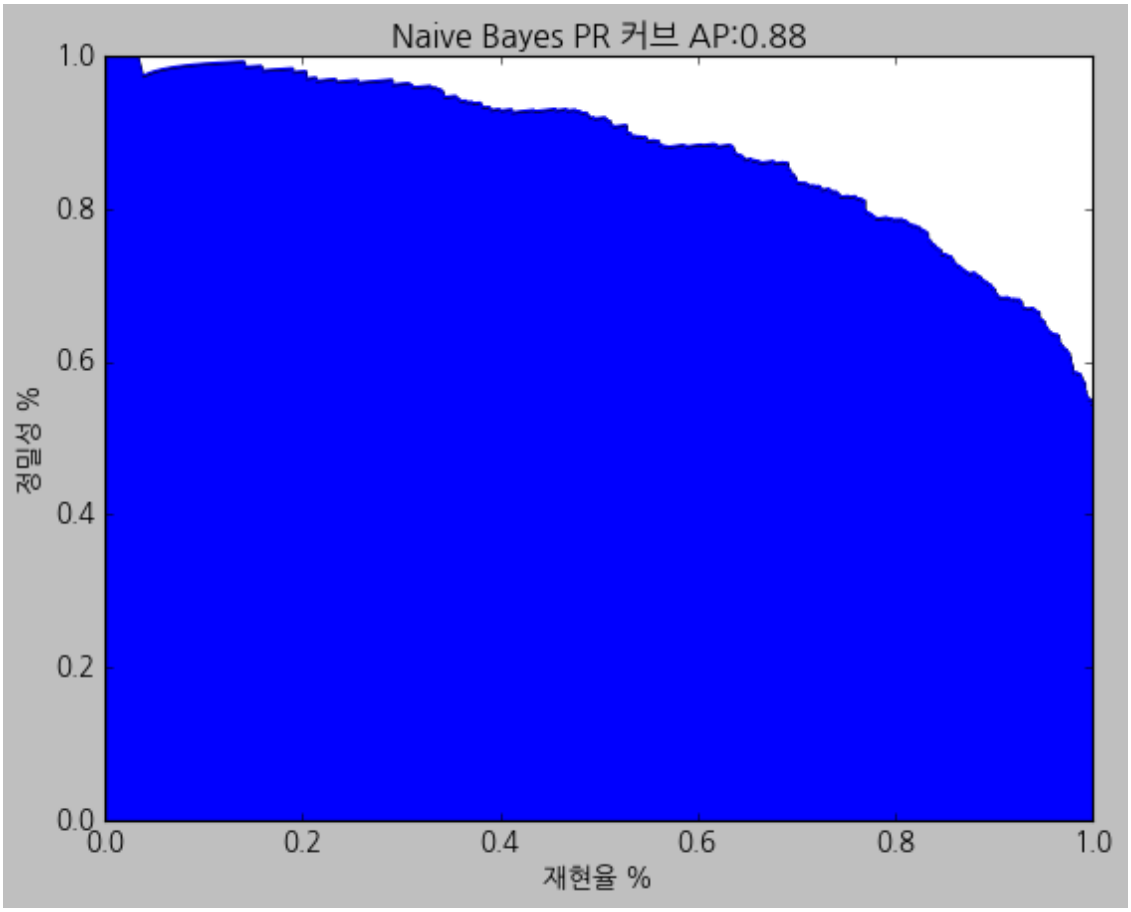


1.2.2 NAIVE BAYES CLASSIFIER

```
In [570]: clf_NB = BernoulliNB()
          clf_NB.fit(X_train, y_train)
          y_pred_NB = clf_NB.predict(X_val)

In [571]: y_score = clf_NB.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          auc_NB=np.round(roc_auc_score(y_val,y_score),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_NB: %s"%auc_NB)
          plt.xlabel("FALSE  ")
          plt.ylabel("TRUE  ")

Out[571]: Text(0, 0.5, 'TRUE  ')
```




```
In [572]: 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)
          ap_score_NB=np.round(average_precision_score(y_val, y_score),2)
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title("Naive Bayes PR  AP:%s"%ap_score_NB)
          plt.xlabel(" %")
          plt.ylabel(" %")

Out[572]: Text(0, 0.5, ' %')
```




```
In [573]: calc_lift(X_train,y_train,clf_NB)
```

Out[573]:

	(%)	LIFT	%	\		
1	828	98.309179	1.800048	814	828	0.98
2	852	93.309859	1.708510	795	1680	0.96
3	838	89.021480	1.629990	746	2518	0.94
4	841	73.127229	1.338965	615	3359	0.88
5	839	61.978546	1.134832	520	4198	0.83
6	840	48.690476	0.891526	409	5038	0.77
7	839	30.274136	0.554322	254	5877	0.71
8	839	23.361144	0.427744	196	6716	0.65
9	841	17.954816	0.328754	151	7557	0.60
10	840	10.238095	0.187460	86	8397	0.55

	lift (%)	
1	814	1.800048
2	1609	1.753626

```
3            2355     1.712479
4            2970     1.618962
5            3490     1.522205
6            3899     1.417050
7            4153     1.293887
8            4349     1.185684
9            4500     1.090318
10           4586     1.000000
```

```
In [574]: calc_lift(X_val,y_val,clf_NB)
```

```
Out[574]:          (%)    LIFT              %  \

1         180     98.333333  1.817248            177        180         0.98
2         180     91.111111  1.683778            164        360         0.95
3         180     85.555556  1.581109            154        540         0.92
4         180     72.777778  1.344969            131        720         0.87
5         180     60.000000  1.108830            108        900         0.82
6         179     44.692737  0.825944             80       1079         0.75
7         181     34.254144  0.633033             62       1260         0.70
8         180     28.888889  0.533881             52       1440         0.64
9         179     14.525140  0.268432             26       1619         0.59
10        181     11.049724  0.204204             20       1800         0.54

          lift (%)

1             177     1.817248
2             341     1.750513
3             495     1.694045
4             626     1.606776
5             734     1.507187
6             814     1.394172
7             876     1.284834
8             928     1.190965
9             954     1.088968
10            974     1.000000
```

1.3

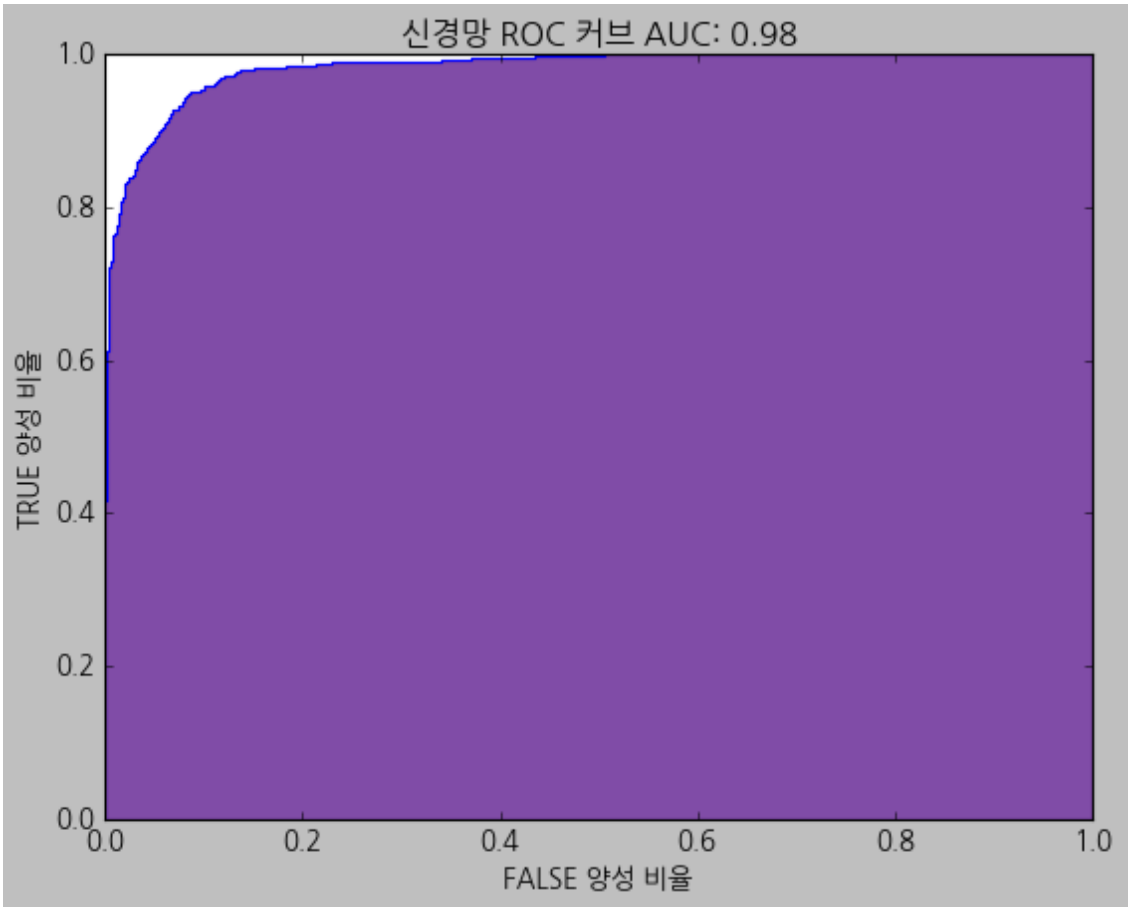
```
In [605]: clf_MLP = MLPClassifier(alpha=1e-05, hidden_layer_sizes=(128))
          clf_MLP.fit(X_train, y_train)
          y_pred_MLP = clf_MLP.predict(X_val)
```

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

```
In [615]: y_score = clf_MLP.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
```

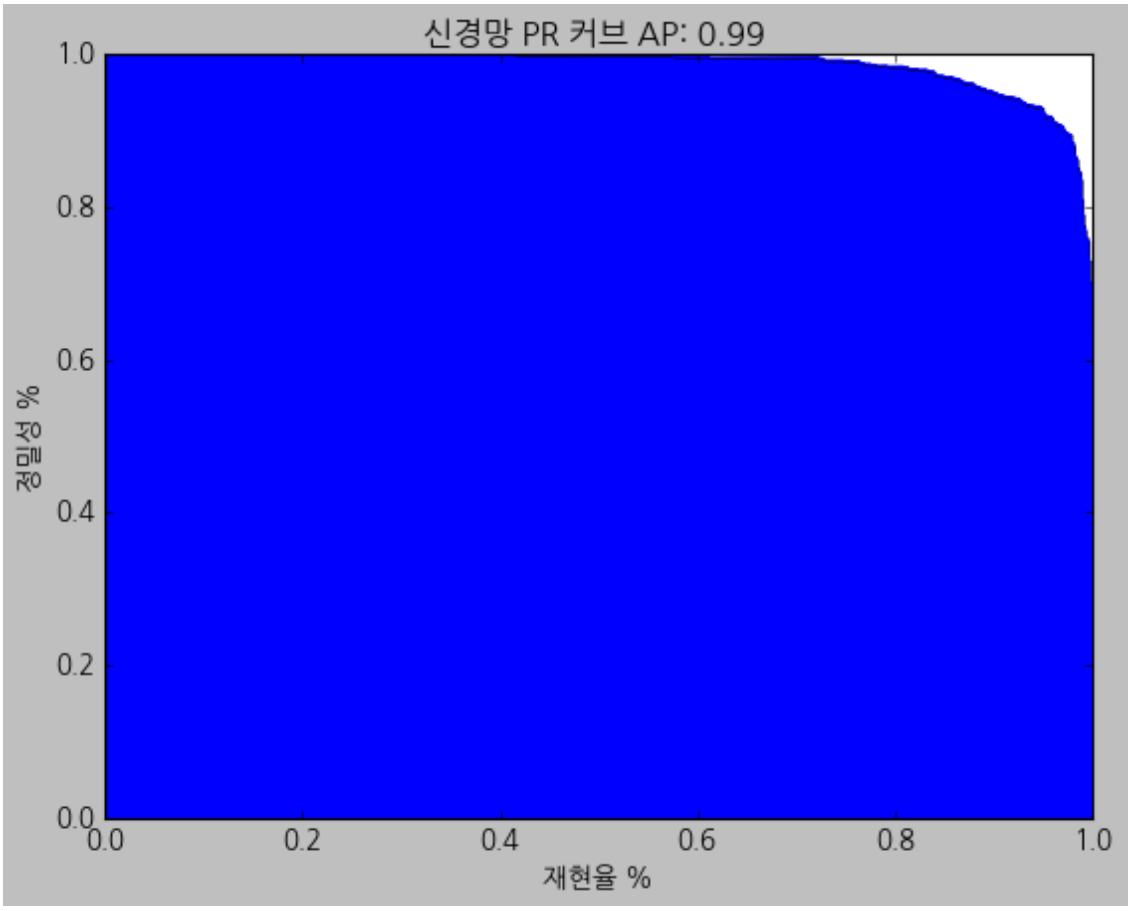
```
fpr, tpr, thresholds = roc_curve(y_val,y_score)
auc_MLP=np.round(roc_auc_score(y_val,y_score),2)
plt.plot(fpr,tpr)
plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
plt.title(" ROC  AUC: %s"%auc_MLP)
plt.xlabel("FALSE  ")
plt.ylabel("TRUE  ")
```

Out[615]: Text(0, 0.5, 'TRUE ')



In [616]:
```
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)
ap_score_MLP=np.round(average_precision_score(y_val, y_score),2)
plt.plot(recall,precision)
plt.fill_between(recall,precision)
plt.title(" PR  AP: %s "%ap_score_MLP)
plt.xlabel(" %")
plt.ylabel(" %")
```

Out[616]: Text(0, 0.5, ' %')



In [617]: calc_lift(X_train,y_train,clf_MLP)

Out[617]:

	(%)	LIFT	%	\		
1	840	100.000000	1.831007	840	840	1.00
2	840	100.000000	1.831007	840	1680	1.00
3	839	100.000000	1.831007	839	2519	1.00
4	840	99.642857	1.824468	837	3359	1.00
5	839	98.092968	1.796090	823	4198	1.00
6	840	44.642857	0.817414	375	5038	0.90
7	840	3.452381	0.063213	29	5878	0.78
8	839	0.238379	0.004365	2	6717	0.68
9	840	0.119048	0.002180	1	7557	0.61
10	840	0.000000	0.000000	0	8397	0.55

lift (%)

```
1              840     1.831007
2             1680     1.831007
3             2519     1.831007
4             3356     1.829372
5             4179     1.822720
6             4554     1.655103
7             4583     1.427613
8             4585     1.249839
9             4586     1.111155
10            4586     1.000000
```

In [618]:
```
calc_lift(X_val,y_val,clf_MLP)
```

Out[618]:

	(%)	LIFT	%	\		
1	180	100.000000	1.848049	180	180	1.00
2	180	100.000000	1.848049	180	360	1.00
3	180	99.444444	1.837782	179	540	1.00
4	180	97.222222	1.796715	175	720	0.99
5	180	81.666667	1.509240	147	900	0.96
6	180	52.222222	0.965092	94	1080	0.88
7	180	6.666667	0.123203	12	1260	0.77
8	180	3.333333	0.061602	6	1440	0.68
9	180	0.555556	0.010267	1	1620	0.60
10	180	0.000000	0.000000	0	1800	0.54

```
           lift (%)

1              180     1.848049
2              360     1.848049
3              539     1.844627
4              714     1.832649
5              861     1.767967
6              955     1.634155
7              967     1.418304
8              973     1.248717
9              974     1.111111
10             974     1.000000
```

1.4

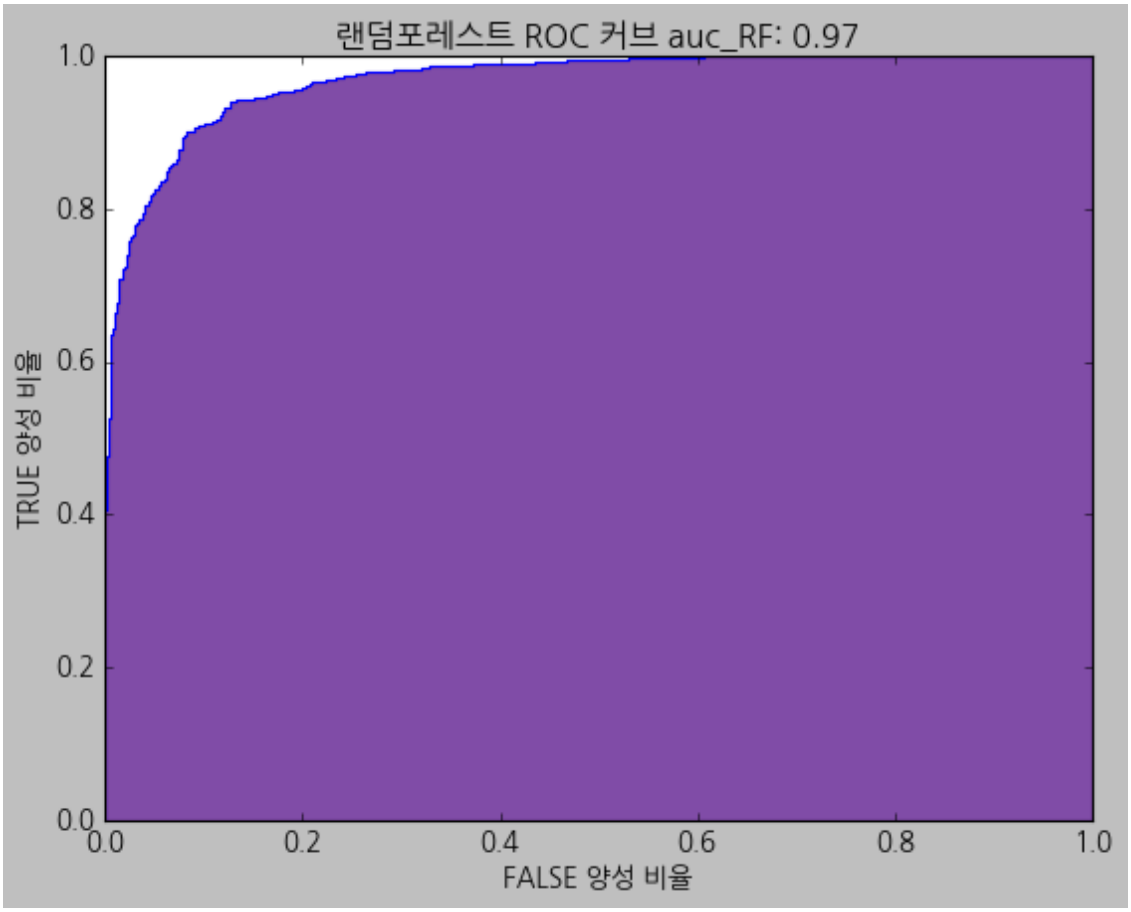
In [580]:
```
clf_RF = RandomForestClassifier(n_estimators=50, criterion='gini')
parameters={'max_depth':[3,5,7]}
clf_RF=GridSearchCV(clf_RF,parameters,cv=5)
clf_RF.fit(X_train,y_train)
print(clf_RF.best_params_)
clf_RF=clf_RF.best_estimator_
y_pred_RF = clf_RF.predict(X_val)
```

```
{'max_depth': 7}
```

1.4.1 max_depth=7 .

```
In [581]: 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_RF=np.round(roc_auc_score(y_val,y_score),2)
          plt.plot(fpr,tpr)
          plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
          plt.title(" ROC  auc_RF: %s"%auc_RF)
          plt.xlabel("FALSE  ")
          plt.ylabel("TRUE  ")
```

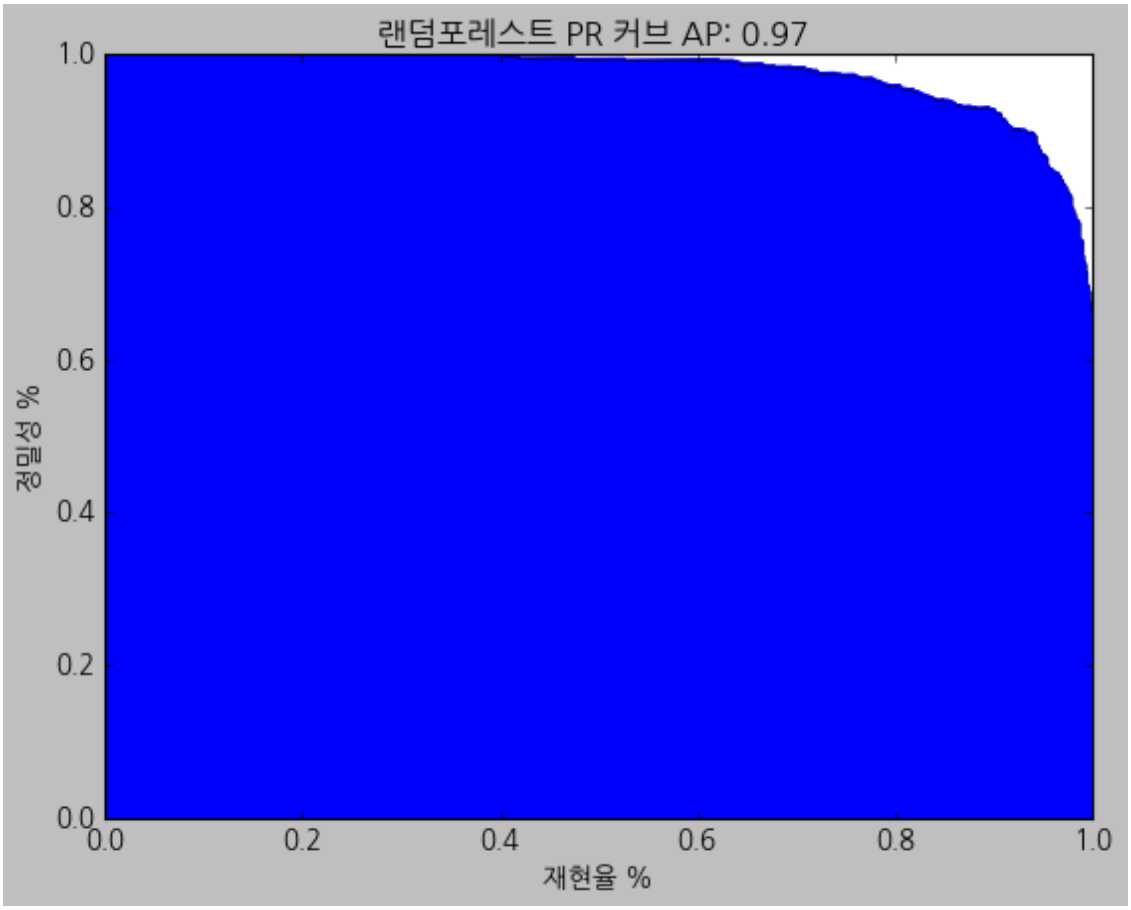
```
Out[581]: Text(0, 0.5, 'TRUE  ')
```




```
In [582]: 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)
```

```
ap_score_RF=np.round(average_precision_score(y_val, y_score),2)
plt.plot(recall,precision)
plt.fill_between(recall,precision)
plt.title(" PR  AP: %s "%ap_score_RF)
plt.xlabel(" %")
plt.ylabel(" %")
```

Out[582]: Text(0, 0.5, ' %')



In [583]: calc_lift(X_train,y_train,clf_RF)

Out[583]:

	(%)	LIFT	%	\		
1	839	100.000000	1.831007	839	839	1.00
2	841	100.000000	1.831007	841	1680	1.00
3	839	99.165673	1.815731	832	2519	1.00
4	840	95.833333	1.754715	805	3359	0.99
5	839	83.432658	1.527658	700	4198	0.96
6	840	48.809524	0.893706	410	5038	0.88
7	840	12.738095	0.233235	107	5878	0.77

```
8          839      3.814064  0.069836              32       6717         0.68
9          840      1.785714  0.032697              15       7557         0.61
10         840      0.595238  0.010899               5       8397         0.55

           lift (%)

1             839     1.831007
2            1680     1.831007
3            2512     1.825919
4            3317     1.808113
5            4017     1.752062
6            4427     1.608946
7            4534     1.412349
8            4566     1.244660
9            4581     1.109944
10           4586     1.000000
```

In [584]:
```
calc_lift(X_val,y_val,clf_RF)
```

Out[584]:
```
           (%)    LIFT               %  \

1          180    100.000000  1.848049             180        180         1.00
2          179    100.000000  1.848049             179        359         1.00
3          181     97.790055  1.807208             177        540         0.99
4          180     92.777778  1.714579             167        720         0.98
5          180     76.111111  1.406571             137        900         0.93
6          180     49.444444  0.913758              89       1080         0.86
7          180     17.777778  0.328542              32       1260         0.76
8          180      6.111111  0.112936              11       1440         0.68
9          180      0.555556  0.010267               1       1620         0.60
10         180      0.555556  0.010267               1       1800         0.54

           lift (%)

1             180     1.848049
2             359     1.848049
3             536     1.834360
4             703     1.804415
5             840     1.724846
6             929     1.589665
7             961     1.409504
8             972     1.247433
9             973     1.109970
10            974     1.000000
```

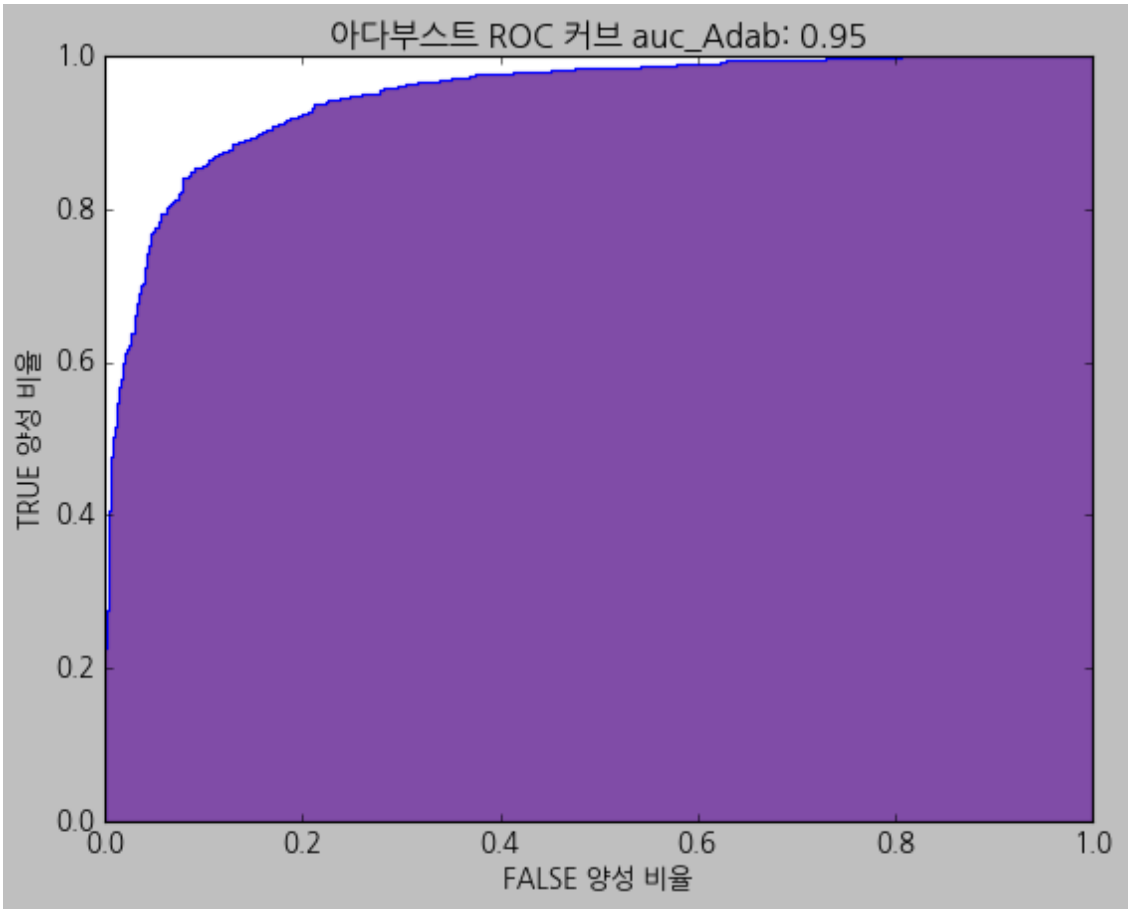
1.5

In [585]:
```
clf_AdaB = AdaBoostClassifier(n_estimators=100)
clf_AdaB.fit(X_train, y_train)
```

```
y_pred_AdaB = clf_AdaB.predict(X_val)
```

In [586]:
```
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_Adab=np.round(roc_auc_score(y_val,y_score),2)
plt.plot(fpr,tpr)
plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
plt.title(" ROC  auc_Adab: %s"%auc_Adab)
plt.xlabel("FALSE  ")
plt.ylabel("TRUE  ")
```

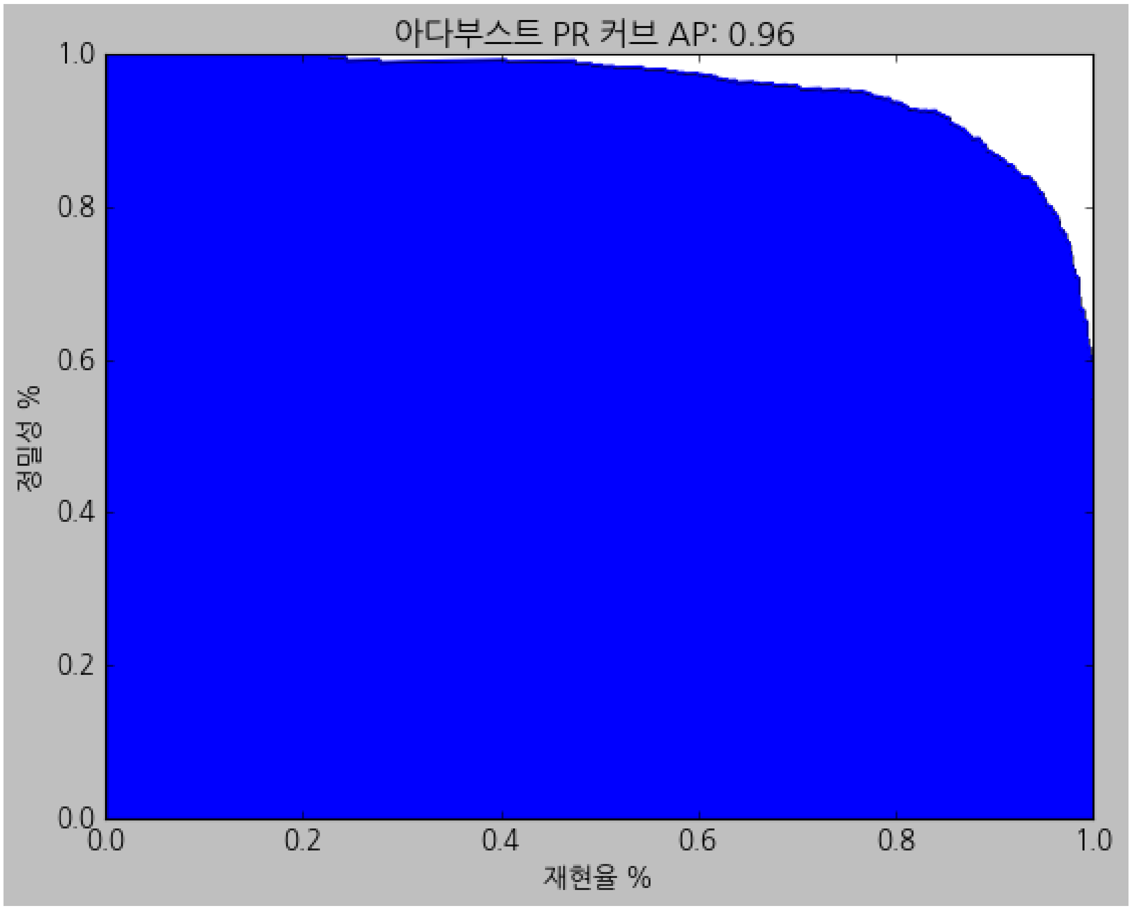
Out[586]:
```
Text(0, 0.5, 'TRUE  ')
```



In [587]:
```
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)
ap_score_ADA=np.round(average_precision_score(y_val, y_score),2)
plt.plot(recall,precision)
plt.fill_between(recall,precision)
```

```
plt.title(" PR  AP: %s "%ap_score_ADA)
plt.xlabel(" %")
plt.ylabel(" %")
```

Out[587]: Text(0, 0.5, ' %')



In [588]: calc_lift(X_train,y_train,clf_AdaB)

Out[588]:

	(%)	LIFT	%	\		
1	840	99.404762	1.820109	835	840	0.99
2	840	98.571429	1.804850	828	1680	0.99
3	839	96.901073	1.774266	813	2519	0.98
4	840	92.261905	1.689322	775	3359	0.97
5	839	77.711561	1.422904	652	4198	0.93
6	840	45.000000	0.823953	378	5038	0.85
7	840	20.595238	0.377100	173	5878	0.76
8	839	10.607867	0.194231	89	6717	0.68
9	840	4.047619	0.074112	34	7557	0.61
10	840	1.071429	0.019618	9	8397	0.55

```
            lift (%)

1                835     1.820109
2               1663     1.812479
3               2476     1.799752
4               3251     1.772136
5               3903     1.702340
6               4281     1.555884
7               4454     1.387429
8               4543     1.238390
9               4577     1.108975
10              4586     1.000000
```

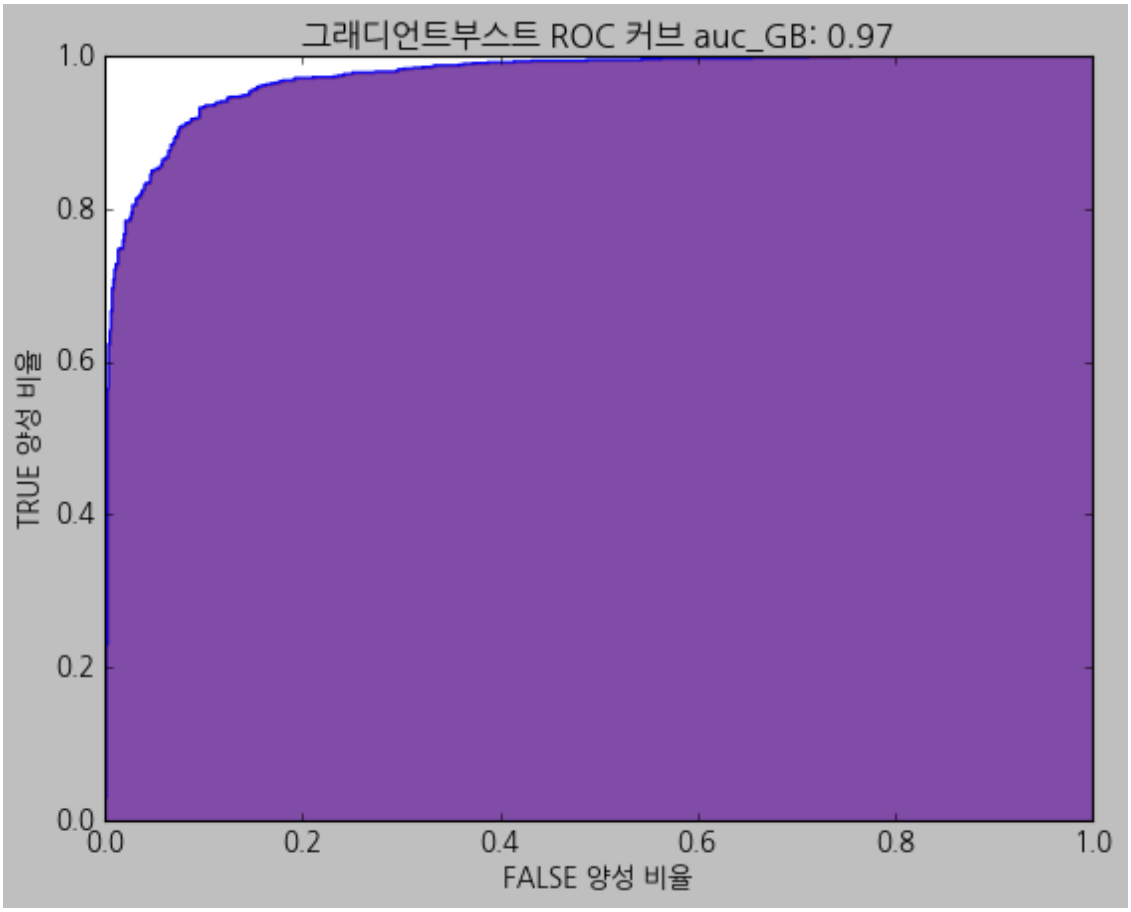
In [589]:
```
calc_lift(X_val,y_val,clf_AdaB)
```

Out[589]:
```
            (%)     LIFT              %  \

1           180  100.000000  1.848049          180        180        1.00
2           180   98.333333  1.817248          177        360        0.99
3           180   96.111111  1.776181          173        540        0.98
4           180   87.222222  1.611910          157        720        0.95
5           180   77.777778  1.437372          140        900        0.92
6           180   44.444444  0.821355           80       1080        0.84
7           180   23.888889  0.441478           43       1260        0.75
8           180    6.666667  0.123203           12       1440        0.67
9           180    5.555556  0.102669           10       1620        0.60
10          180    1.111111  0.020534            2       1800        0.54

            lift (%)

1                180     1.848049
2                357     1.832649
3                530     1.813826
4                687     1.763347
5                827     1.698152
6                907     1.552019
7                950     1.393370
8                962     1.234600
9                972     1.108830
10               974     1.000000
```

1.6

In [590]:
```
clf_GB = GradientBoostingClassifier(n_estimators=100, learning_rate=0.05, random_sta
clf_GB.fit(X_train, y_train)
y_pred_GB = clf_GB.predict(X_val)
```

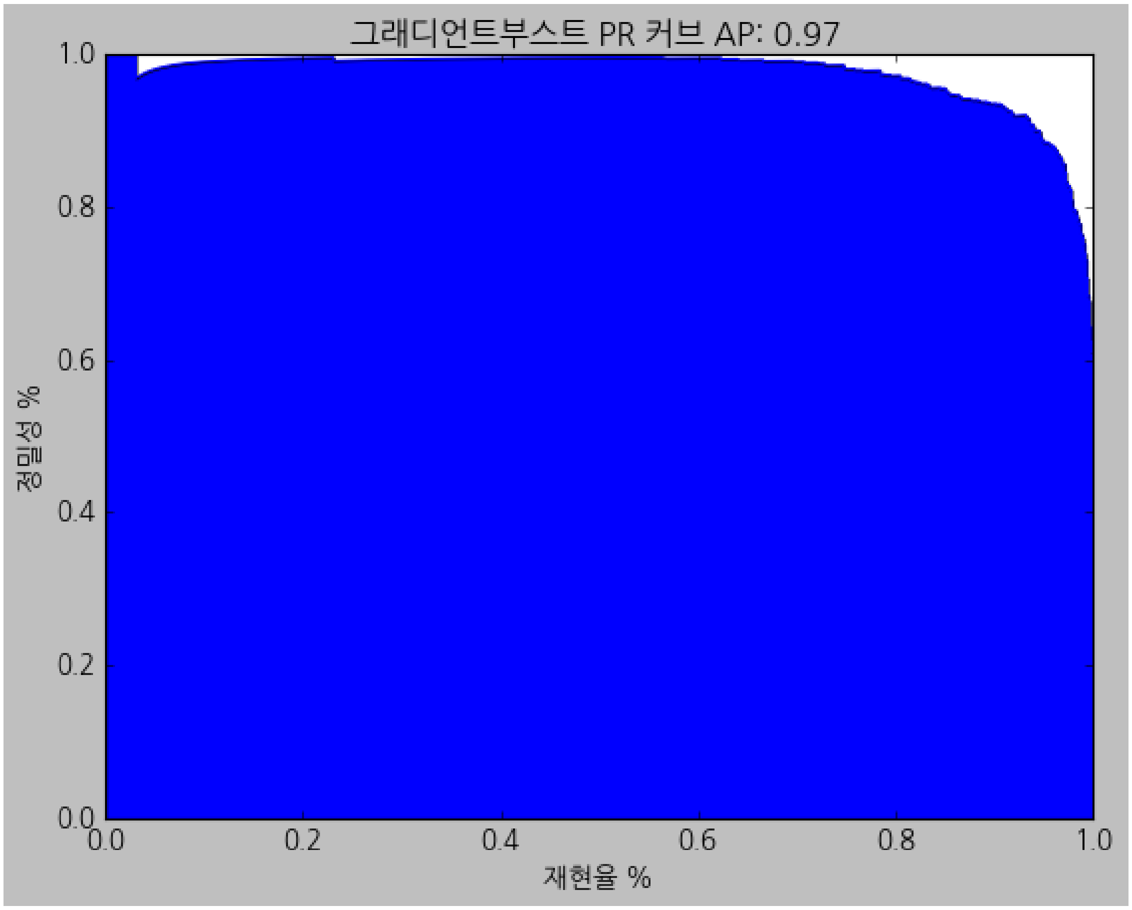
```
In [591]: y_score = clf_GB.predict_proba(X_val)
          y_score=np.array(pd.DataFrame(y_score)[1])
          fpr, tpr, thresholds = roc_curve(y_val,y_score)
          auc_GB=np.round(roc_auc_score(y_val,y_score),2)
          plt.plot(fpr,tpr)
          plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
          plt.title(" ROC  auc_GB: %s"%auc_GB)
          plt.xlabel("FALSE  ")
          plt.ylabel("TRUE  ")

Out[591]: Text(0, 0.5, 'TRUE  ')
```




```
In [592]: 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)
          ap_score_GB=np.round(average_precision_score(y_val, y_score),2)
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title(" PR  AP: %s "%ap_score_GB)
```

```
plt.xlabel(" %")
plt.ylabel(" %")
```

```
Out[592]: Text(0, 0.5, ' %')
```




```
In [593]: calc_lift(X_train,y_train,clf_GB)
```

```
Out[593]:          (%)    LIFT             %  \

          1        787    99.364676  1.819375          782       787        0.99
          2        893    99.776036  1.826907          891      1680        1.00
          3        838    99.880668  1.828822          837      2518        1.00
          4        841    97.265161  1.780932          818      3359        0.99
          5        833    82.232893  1.505690          685      4192        0.96
          6        846    48.345154  0.885203          409      5038        0.88
          7        839    12.038141  0.220419          101      5877        0.77
          8        834     5.755396  0.105382           48      6711        0.68
          9        844     1.658768  0.030372           14      7555        0.61
          10       842     0.118765  0.002175            1      8397        0.55
```

```
            lift (%)

1              782        1.819375
2             1673        1.823378
3             2510        1.825190
4             3328        1.814109
5             4013        1.752823
6             4422        1.607129
7             4523        1.409162
8             4571        1.247137
9             4585        1.111207
10            4586        1.000000
```

In [594]:
```python
calc_lift(X_val,y_val,clf_GB)
```

Out[594]:
```
            (%)      LIFT                %  \

1          163     99.386503  1.836712            162        163          0.99
2          197     99.492386  1.838668            196        360          0.99
3          180    100.000000  1.848049            180        540          1.00
4          180     95.000000  1.755647            171        720          0.98
5          178     76.404494  1.411993            136        898          0.94
6          182     51.648352  0.954487             94       1080          0.87
7          179     13.407821  0.247783             24       1259          0.76
8          181      4.419890  0.081682              8       1440          0.67
9          180      1.666667  0.030801              3       1620          0.60
10         180      0.000000  0.000000              0       1800          0.54

            lift (%)

1              162        1.836712
2              358        1.837782
3              538        1.841205
4              709        1.819815
5              845        1.738977
6              939        1.606776
7              963        1.413560
8              971        1.246150
9              974        1.111111
10             974        1.000000
```

1.6.1 SVM ( )

In [595]:
```python
clf_SVM = SVC(probability=True)
parameters={'C': [1, 10, 100]}
clf_SVM=GridSearchCV(clf_SVM,parameters,cv=5)
clf_SVM.fit(X_train,y_train)
print(clf_SVM.best_params_)
```

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

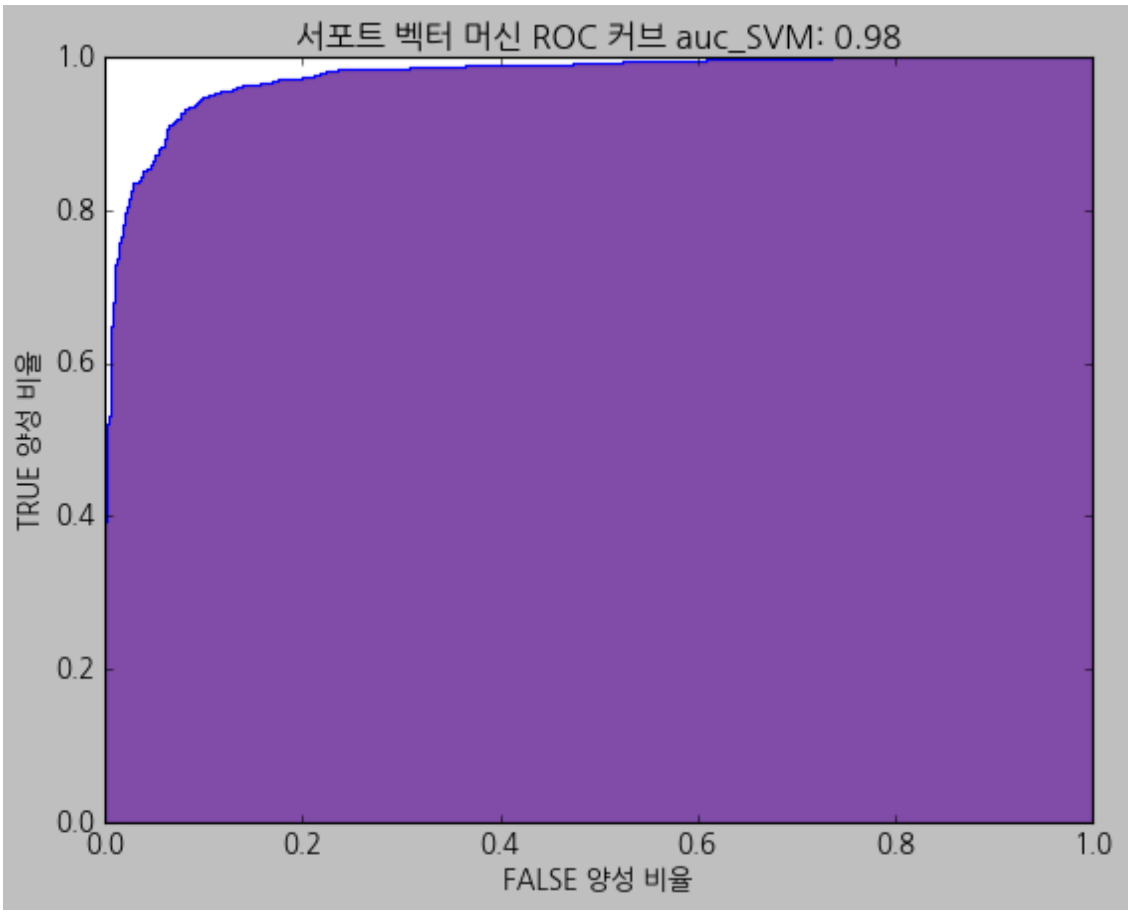
```
{'C': 10}
```

c=10

```
In [596]: 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_SVM=np.round(roc_auc_score(y_val,y_score),2)
          plt.plot(fpr,tpr)
          plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
```

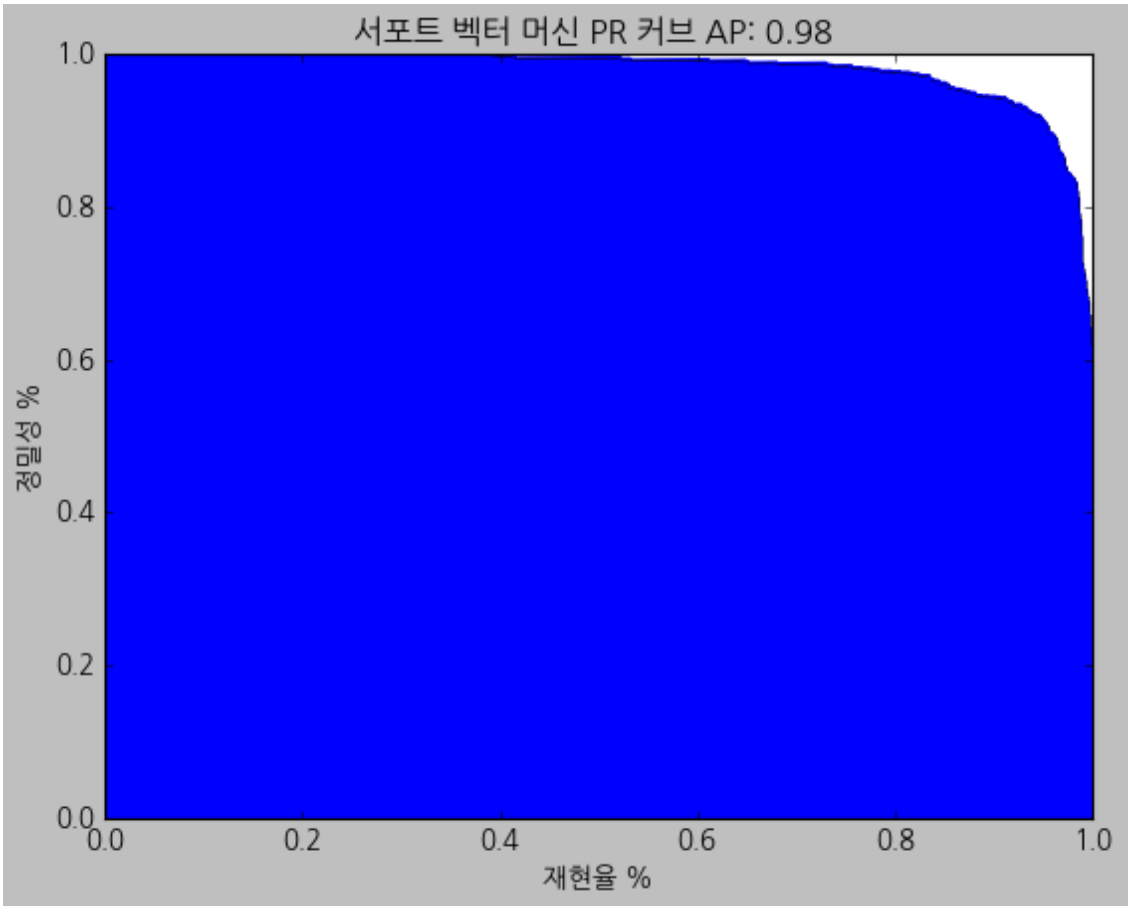
```
plt.title("   ROC  auc_SVM: %s"%auc_SVM)
plt.xlabel("FALSE  ")
plt.ylabel("TRUE  ")
```

```
Out[596]: Text(0, 0.5, 'TRUE  ')
```




```
In [597]: 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)
          ap_score_SVM=np.round(average_precision_score(y_val, y_score),2)
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title("   PR  AP: %s "%ap_score_SVM)
          plt.xlabel(" %")
          plt.ylabel(" %")
```

```
Out[597]: Text(0, 0.5, ' %')
```




```
In [598]: calc_lift(X_train,y_train,clf_SVM)
```

Out[598]:

	(%)	LIFT	%	\		
1	840	100.000000	1.831007	840	840	1.00
2	840	99.880952	1.828828	839	1680	1.00
3	839	100.000000	1.831007	839	2519	1.00
4	840	98.928571	1.811389	831	3359	1.00
5	839	98.808105	1.809184	829	4198	1.00
6	840	45.000000	0.823953	378	5038	0.90
7	840	0.952381	0.017438	8	5878	0.78
8	839	2.383790	0.043647	20	6717	0.68
9	840	0.238095	0.004360	2	7557	0.61
10	840	0.000000	0.000000	0	8397	0.55

	lift (%)	
1	840	1.831007
2	1679	1.829918

```
3            2518     1.830281
4            3349     1.825556
5            4178     1.822284
6            4556     1.655830
7            4564     1.421694
8            4584     1.249566
9            4586     1.111155
10           4586     1.000000
```

```
In [599]: calc_lift(X_val,y_val,clf_SVM)
```

```
Out[599]:          (%)    LIFT              %  \

          1         180    100.000000  1.848049            180        180         1.00
          2         180    100.000000  1.848049            180        360         1.00
          3         180     97.777778  1.806982            176        540         0.99
          4         180     97.222222  1.796715            175        720         0.99
          5         180     80.000000  1.478439            144        900         0.95
          6         180     47.777778  0.882957             86       1080         0.87
          7         180     11.666667  0.215606             21       1260         0.76
          8         180      4.444444  0.082136              8       1440         0.67
          9         180      1.666667  0.030801              3       1620         0.60
          10        180      0.555556  0.010267              1       1800         0.54

                    lift (%)

          1             180     1.848049
          2             360     1.848049
          3             536     1.834360
          4             711     1.824949
          5             855     1.755647
          6             941     1.610198
          7             962     1.410971
          8             970     1.244867
          9             973     1.109970
          10            974     1.000000
```

```
In [611]: print('                          ')
          print('--------------------------------')
          print('Naive Bayes                    Accuracy: %s '%np.round(accuracy_score(y_val,y
          print('Neural Network                 Accuracy: %s '%np.round(accuracy_score(y_val,y
          print('Logistic Regression            Accuracy: %s '%np.round(accuracy_score(y_val,y
          print('Random Forest                  Accuracy: %s '%np.round(accuracy_score(y_val,y
          print('AdaBoost                       Accuracy: %s '%np.round(accuracy_score(y_val,y
          print('GradientBoost                  Accuracy: %s '%np.round(accuracy_score(y_val,y
          print('Support Vector Machine         Accuracy: %s '%np.round(accuracy_score(y_val,y
```

```
          print('KNN                               Accuracy: %s '%np.round(accuracy_score(y_val,y
          print('Decision Tree                     Accuracy: %s '%np.round(accuracy_score(y_val,y


--------------------------------
Naive Bayes                     Accuracy: 0.77
Neural Network                  Accuracy: 0.93
Logistic Regression             Accuracy: 0.8
Random Forest                   Accuracy: 0.91
AdaBoost                        Accuracy: 0.88
GradientBoost                   Accuracy: 0.92
Support Vector Machine          Accuracy: 0.92
KNN                             Accuracy: 0.89
Decision Tree                   Accuracy: 0.9


In [612]: print('     Confusion_matrix                    ')
          print('--------------------------------')
          print('Naive Bayes        \n confusion_matrix:\n %s '%confusion_matrix(y_val,y_pred_
          print('Neural Network    \n confusion_matrix:\n %s '%confusion_matrix(y_val,y_pred_MN
          print('Logistic Regression \n confusion_matrix:\n %s '%confusion_matrix(y_val,y_pred_
          print('Random Forest     \n confusion_matrix:\n %s '%confusion_matrix(y_val,y_pred_RF
          print('AdaBoost          \n confusion_matrix:\n %s '%confusion_matrix(y_val,y_pred_Ad
          print('GradientBoost     \n confusion_matrix:\n %s '%confusion_matrix(y_val,y_pred_GB
          print('Support Vector Machine \n confusion_matrix:\n %s '%confusion_matrix(y_val,y_pr
          print('KNN     \n confusion_matrix:\n %s '%confusion_matrix(y_val,y_pred_KNN) )
          print('Decision Tree \n confusion_matrix:\n %s '%confusion_matrix(y_val,y_pred_DT) )

     Confusion_matrix
--------------------------------
Naive Bayes
 confusion_matrix:
 [[632 194]
 [221 753]]
Neural Network
 confusion_matrix:
 [[758  68]
 [ 54 920]]
Logistic Regression
 confusion_matrix:
 [[628 198]
 [169 805]]
Random Forest
 confusion_matrix:
 [[753  73]
 [ 96 878]]
AdaBoost
 confusion_matrix:
```

```
 [[722 104]
 [120 854]]
GradientBoost
 confusion_matrix:
 [[753  73]
 [ 80 894]]
Support Vector Machine
 confusion_matrix:
 [[757  69]
 [ 66 908]]
KNN
 confusion_matrix:
 [[752  74]
 [131 843]]
Decision Tree
 confusion_matrix:
 [[773  53]
 [132 842]]
```

```
In [613]: print('     AUC                    ')
          print('-------------------------------')
          print('Naive Bayes                    AUC: %s '%auc_NB )
          print('Neural Network                 AUC: %s '%auc_MLP )
          print('Logistic Regression            AUC: %s '%auc_LR )
          print('Random Forest                  AUC: %s '%auc_RF )
          print('AdaBoost                       AUC: %s '%auc_Adab )
          print('GradientBoost                  AUC: %s '%auc_GB )
          print('Support Vector Machine         AUC: %s '%auc_SVM )
          print('KNN                            AUC: %s '%auc_KNN )
          print('Decision Tree                  AUC: %s '%auc_DT )
```

```
     AUC
-------------------------------
Naive Bayes                    AUC: 0.85
Neural Network                 AUC: 0.98
Logistic Regression            AUC: 0.88
Random Forest                  AUC: 0.97
AdaBoost                       AUC: 0.95
GradientBoost                  AUC: 0.97
Support Vector Machine         AUC: 0.98
KNN                            AUC: 0.95
Decision Tree                  AUC: 0.96
```

```
In [614]: 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 )
          print('KNN                            AP  Score: %s '%ap_score_KNN )
          print('Decision Tree                  AP  Score: %s '%ap_score_DT )
```

```
      Validation AP Score
--------------------------------
Naive Bayes                    AP  Score: 0.88
Neural Network                 AP  Score: 0.99
Logistic Regression            AP  Score: 0.9
Random Forest                  AP  Score: 0.97
AdaBoost                       AP  Score: 0.96
GradientBoost                  AP  Score: 0.97
Support Vector Machine         AP  Score: 0.98
KNN                            AP  Score: 0.94
Decision Tree                  AP  Score: 0.97
```

1.6.2 :

```
In [132]: final_pred_MLP=clf_MLP.predict(X_test)
          print("  test set :%s"%np.round(accuracy_score(y_test,final_pred_MLP),2))
```

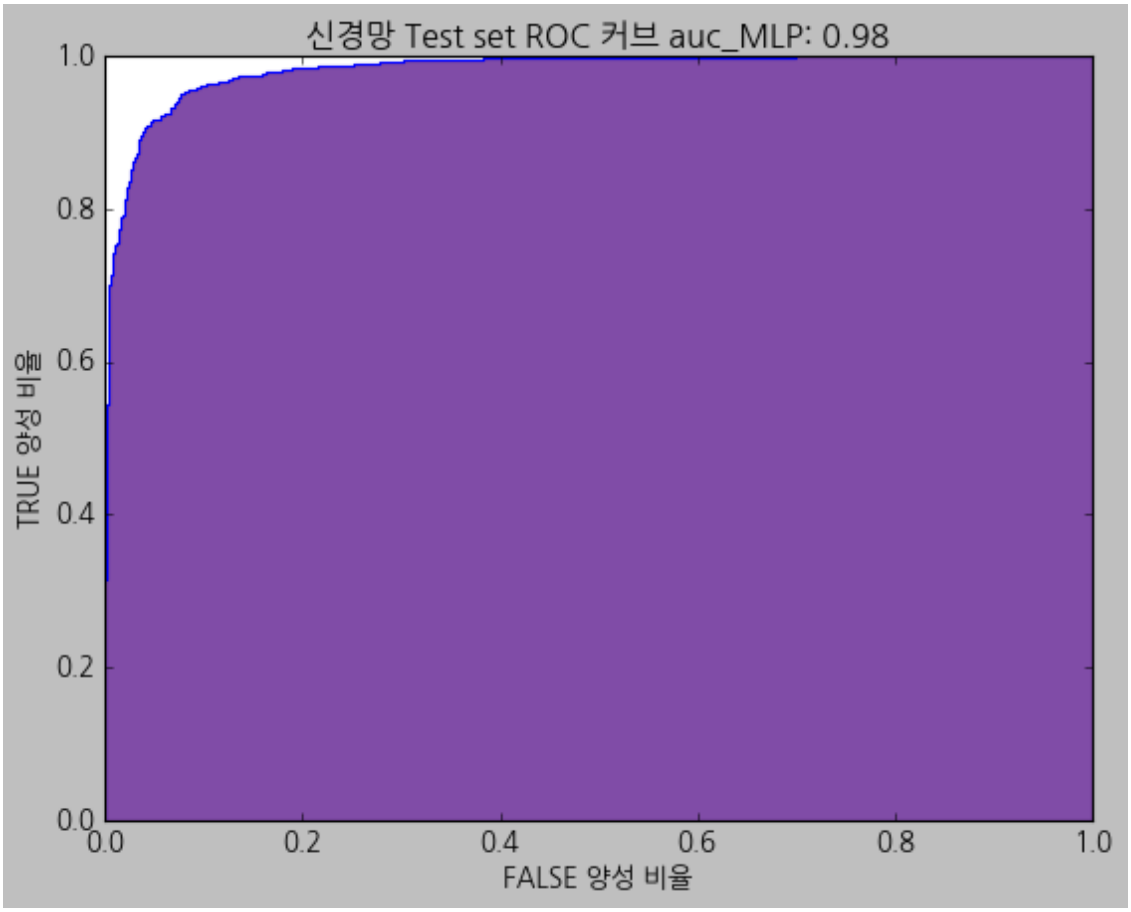
```
  test set :0.93
```

```
In [87]: print('Neural Network    \n confusion_matrix:\n %s '%confusion_matrix(y_test,final_pr
```

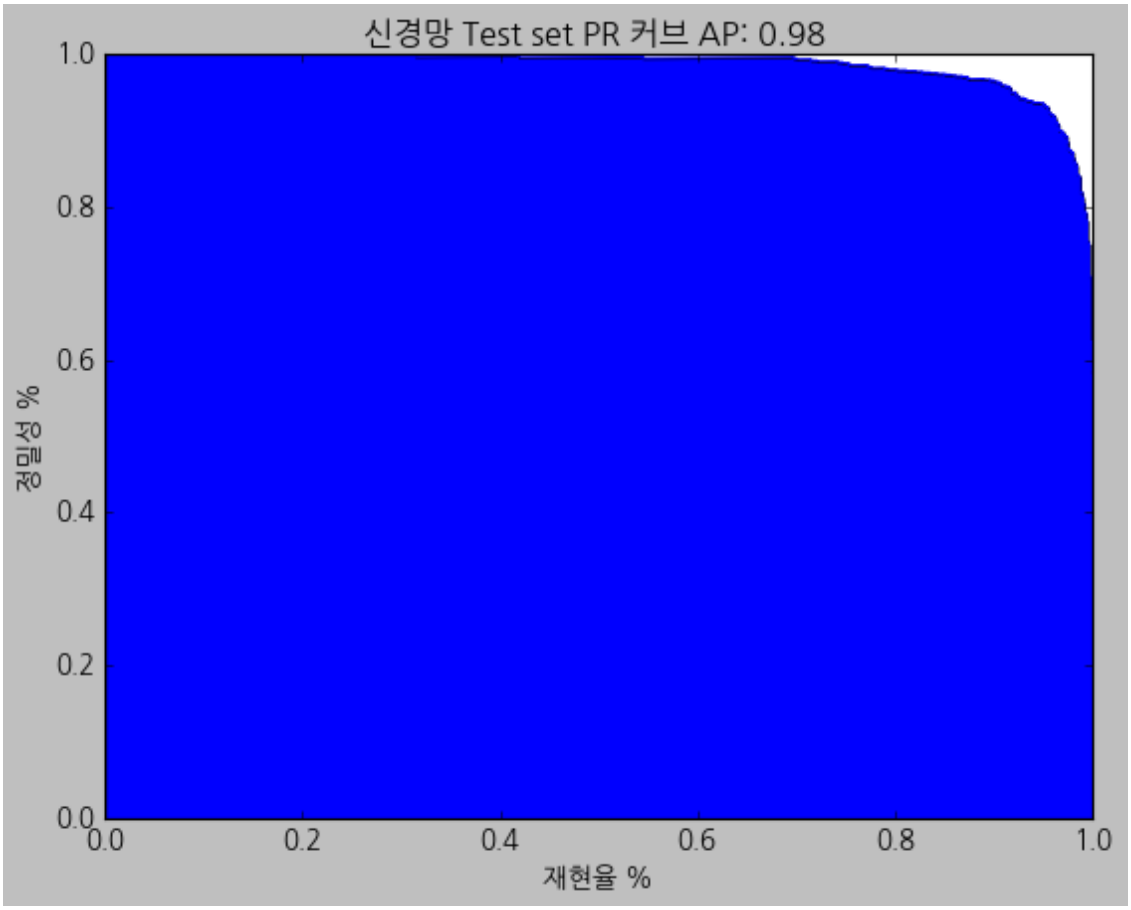
```
Neural Network
 confusion_matrix:
 [[769  67]
 [ 56 908]]
```

```
In [126]: y_score = clf_MLP.predict_proba(X_test)
          y_score=np.array(pd.DataFrame(y_score)[1])
          fpr, tpr, thresholds = roc_curve(y_test,y_score)
          auc_MLP=np.round(roc_auc_score(y_test,y_score),2)
          plt.plot(fpr,tpr)
          plt.fill_between(fpr,tpr,color='indigo',alpha=0.7)
          plt.title(" Test set ROC  auc_MLP: %s"%auc_MLP)
          plt.xlabel("FALSE  ")
          plt.ylabel("TRUE  ")
```

```
Out[126]: Text(0, 0.5, 'TRUE  ')
```




```
In [127]: y_score = clf_MLP.predict_proba(X_test)
          y_score=np.array(pd.DataFrame(y_score)[1])
          precision, recall, thresholds = precision_recall_curve(y_test, y_score)
          ap_score_MLP=np.round(average_precision_score(y_test, y_score),2)
          plt.plot(recall,precision)
          plt.fill_between(recall,precision)
          plt.title(" Test set PR  AP: %s "%ap_score_MLP)
          plt.xlabel(" %")
          plt.ylabel(" %")

Out[127]: Text(0, 0.5, ' %')
```




```
In [128]: calc_lift(X_test,y_test,clf_MLP)
```

Out[128]:

	(%)	LIFT	%	\		
1	180	100.000000	1.867220	180	180	1.00
2	180	99.444444	1.856846	179	360	1.00
3	180	98.888889	1.846473	178	540	0.99
4	180	97.777778	1.825726	176	720	0.99
5	180	86.111111	1.607884	155	900	0.96
6	180	41.666667	0.778008	75	1080	0.87
7	180	8.888889	0.165975	16	1260	0.76
8	180	1.666667	0.031120	3	1440	0.67
9	180	0.555556	0.010373	1	1620	0.59
10	180	0.555556	0.010373	1	1800	0.54

	lift (%)	
1	180	1.867220
2	359	1.862033

```
3              537     1.856846
4              713     1.849066
5              868     1.800830
6              943     1.630360
7              959     1.421162
8              962     1.247407
9              963     1.109959
10             964     1.000000
```

```
In [131]: variable_importances=list(zip(X_test.columns,np.round(clf_RF.feature_importances_,2))
          feature_importance_matrix=pd.DataFrame(variable_importances,columns=['Features','Impo
          print(feature_importance_matrix)
```

```
                          Features  Importance
15                    Class_Eco_plus        0.00
7                      Gate location        0.01
13         Departure Delay in Minutes        0.01
2                              Age           0.02
4                    Flight Distance         0.02
11                   Checkin service         0.02
6   Departure/Arrival time convenient        0.04
10                  Baggage handling         0.04
3                     Type of Travel        0.05
14                         Class_Eco        0.05
0                             Gender        0.06
9                   On-board service        0.08
12                   Online boarding        0.08
1                      Customer Type        0.10
8                    Online support         0.19
5                      Seat comfort         0.23
```

1.6.3 5: seat comfort,Online support ,Customer Type, online boarding, On-board service