**from** **sklearn.model\_selection** **import** train\_test\_split *#트레인, 테스트 데이터 분류*

**from** **sklearn.svm** **import** SVC *#SVC 서포트 벡터 머신 라이브러리(분류화)*

**from** **sklearn.metrics** **import** accuracy\_score, classification\_report *#metrics 는 confusion metrics 의 라이브러리 : accuracy/recall/precision 등 모델 평가*

**from** **sklearn.neighbors** **import** KNeighborsClassifier *#KNN 분류 라이브러리*

**import** **matplotlib.pyplot** **as** **plt**

**import** **seaborn** **as** **sns**

%**matplotlib** inline

**1. 와인판별**

In [2]:

*#와인 상관관계 분석*

*#데이터프레임 내의 각 항목간의 상관관계를 분석*

*#요구사항 : 함수 5개이상 사용, 그래프 2개*

*#훈련, 검증, KNN, 정답률 계산, 모델 테스트 -> 갈무리*

***목적 : 와인 피처를 통해, 와인의 Quality를 맞춰보장!***

In [5]:

wine\_data = pd.read\_csv('wine.csv', encoding='UTF-8')

In [7]:

wine\_data

Out[7]:

|  | **fixed acidity** | **volatile acidity** | **citric acid** | **residual sugar** | **chlorides** | **free sulfur dioxide** | **total sulfur dioxide** | **density** | **pH** | **sulphates** | **alcohol** | **quality** | **class** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 11.6 | 0.580 | 0.66 | 2.20 | 0.074 | 10.0 | 47.0 | 1.00080 | 3.25 | 0.57 | 9.0 | 3 | 0 |
| **1** | 10.4 | 0.610 | 0.49 | 2.10 | 0.200 | 5.0 | 16.0 | 0.99940 | 3.16 | 0.63 | 8.4 | 3 | 0 |
| **2** | 7.4 | 1.185 | 0.00 | 4.25 | 0.097 | 5.0 | 14.0 | 0.99660 | 3.63 | 0.54 | 10.7 | 3 | 0 |
| **3** | 10.4 | 0.440 | 0.42 | 1.50 | 0.145 | 34.0 | 48.0 | 0.99832 | 3.38 | 0.86 | 9.9 | 3 | 0 |
| **4** | 8.3 | 1.020 | 0.02 | 3.40 | 0.084 | 6.0 | 11.0 | 0.99892 | 3.48 | 0.49 | 11.0 | 3 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1594** | 8.6 | 0.420 | 0.39 | 1.80 | 0.068 | 6.0 | 12.0 | 0.99516 | 3.35 | 0.69 | 11.7 | 8 | 1 |
| **1595** | 5.5 | 0.490 | 0.03 | 1.80 | 0.044 | 28.0 | 87.0 | 0.99080 | 3.50 | 0.82 | 14.0 | 8 | 1 |
| **1596** | 7.2 | 0.330 | 0.33 | 1.70 | 0.061 | 3.0 | 13.0 | 0.99600 | 3.23 | 1.10 | 10.0 | 8 | 1 |
| **1597** | 7.2 | 0.380 | 0.31 | 2.00 | 0.056 | 15.0 | 29.0 | 0.99472 | 3.23 | 0.76 | 11.3 | 8 | 1 |
| **1598** | 7.4 | 0.360 | 0.30 | 1.80 | 0.074 | 17.0 | 24.0 | 0.99419 | 3.24 | 0.70 | 11.4 | 8 | 1 |

1599 rows × 13 columns

In [10]:

wine\_data.describe() *#와인 데이터 간단 산술적 묘사*

Out[10]:

|  | **fixed acidity** | **volatile acidity** | **citric acid** | **residual sugar** | **chlorides** | **free sulfur dioxide** | **total sulfur dioxide** | **density** | **pH** | **sulphates** | **alcohol** | **quality** | **class** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 |
| **mean** | 8.319637 | 0.527821 | 0.270976 | 2.538806 | 0.087467 | 15.874922 | 46.467792 | 0.996747 | 3.311113 | 0.658149 | 10.422983 | 5.636023 | 0.534709 |
| **std** | 1.741096 | 0.179060 | 0.194801 | 1.409928 | 0.047065 | 10.460157 | 32.895324 | 0.001887 | 0.154386 | 0.169507 | 1.065668 | 0.807569 | 0.498950 |
| **min** | 4.600000 | 0.120000 | 0.000000 | 0.900000 | 0.012000 | 1.000000 | 6.000000 | 0.990070 | 2.740000 | 0.330000 | 8.400000 | 3.000000 | 0.000000 |
| **25%** | 7.100000 | 0.390000 | 0.090000 | 1.900000 | 0.070000 | 7.000000 | 22.000000 | 0.995600 | 3.210000 | 0.550000 | 9.500000 | 5.000000 | 0.000000 |
| **50%** | 7.900000 | 0.520000 | 0.260000 | 2.200000 | 0.079000 | 14.000000 | 38.000000 | 0.996750 | 3.310000 | 0.620000 | 10.200000 | 6.000000 | 1.000000 |
| **75%** | 9.200000 | 0.640000 | 0.420000 | 2.600000 | 0.090000 | 21.000000 | 62.000000 | 0.997835 | 3.400000 | 0.730000 | 11.100000 | 6.000000 | 1.000000 |
| **max** | 15.900000 | 1.580000 | 1.000000 | 15.500000 | 0.611000 | 72.000000 | 289.000000 | 1.003690 | 4.010000 | 2.000000 | 14.900000 | 8.000000 | 1.000000 |

In [11]:

wine\_data['quality'].value\_counts() *#와인 Quality 별 카운트*

Out[11]:

5 681

6 638

7 199

4 53

8 18

3 10

Name: quality, dtype: int64

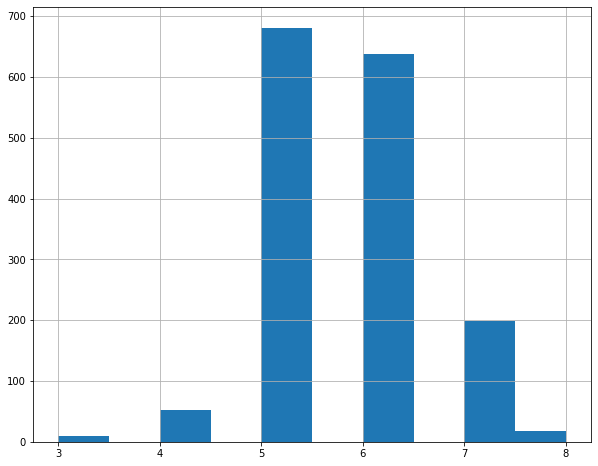
In [15]:

*#와인 quality 질 별 그래프*

wine\_data['quality'].hist(bins=10, figsize=(10,8))

Out[15]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x26bc6a21088>



**1-1. 와인 상관관계 분석**

In [16]:

wine\_data.corr()

Out[16]:

|  | **fixed acidity** | **volatile acidity** | **citric acid** | **residual sugar** | **chlorides** | **free sulfur dioxide** | **total sulfur dioxide** | **density** | **pH** | **sulphates** | **alcohol** | **quality** | **class** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **fixed acidity** | 1.000000 | -0.256131 | 0.671703 | 0.114777 | 0.093705 | -0.153794 | -0.113181 | 0.668047 | -0.682978 | 0.183006 | -0.061668 | 0.124052 | 0.095093 |
| **volatile acidity** | -0.256131 | 1.000000 | -0.552496 | 0.001918 | 0.061298 | -0.010504 | 0.076470 | 0.022026 | 0.234937 | -0.260987 | -0.202288 | -0.390558 | -0.321441 |
| **citric acid** | 0.671703 | -0.552496 | 1.000000 | 0.143577 | 0.203823 | -0.060978 | 0.035533 | 0.364947 | -0.541904 | 0.312770 | 0.109903 | 0.226373 | 0.159129 |
| **residual sugar** | 0.114777 | 0.001918 | 0.143577 | 1.000000 | 0.055610 | 0.187049 | 0.203028 | 0.355283 | -0.085652 | 0.005527 | 0.042075 | 0.013732 | -0.002160 |
| **chlorides** | 0.093705 | 0.061298 | 0.203823 | 0.055610 | 1.000000 | 0.005562 | 0.047400 | 0.200632 | -0.265026 | 0.371260 | -0.221141 | -0.128907 | -0.109494 |
| **free sulfur dioxide** | -0.153794 | -0.010504 | -0.060978 | 0.187049 | 0.005562 | 1.000000 | 0.667666 | -0.021946 | 0.070377 | 0.051658 | -0.069408 | -0.050656 | -0.061757 |
| **total sulfur dioxide** | -0.113181 | 0.076470 | 0.035533 | 0.203028 | 0.047400 | 0.667666 | 1.000000 | 0.071269 | -0.066495 | 0.042947 | -0.205654 | -0.185100 | -0.231963 |
| **density** | 0.668047 | 0.022026 | 0.364947 | 0.355283 | 0.200632 | -0.021946 | 0.071269 | 1.000000 | -0.341699 | 0.148506 | -0.496180 | -0.174919 | -0.159110 |
| **pH** | -0.682978 | 0.234937 | -0.541904 | -0.085652 | -0.265026 | 0.070377 | -0.066495 | -0.341699 | 1.000000 | -0.196648 | 0.205633 | -0.057731 | -0.003264 |
| **sulphates** | 0.183006 | -0.260987 | 0.312770 | 0.005527 | 0.371260 | 0.051658 | 0.042947 | 0.148506 | -0.196648 | 1.000000 | 0.093595 | 0.251397 | 0.218072 |
| **alcohol** | -0.061668 | -0.202288 | 0.109903 | 0.042075 | -0.221141 | -0.069408 | -0.205654 | -0.496180 | 0.205633 | 0.093595 | 1.000000 | 0.476166 | 0.434751 |
| **quality** | 0.124052 | -0.390558 | 0.226373 | 0.013732 | -0.128907 | -0.050656 | -0.185100 | -0.174919 | -0.057731 | 0.251397 | 0.476166 | 1.000000 | 0.848279 |
| **class** | 0.095093 | -0.321441 | 0.159129 | -0.002160 | -0.109494 | -0.061757 | -0.231963 | -0.159110 | -0.003264 | 0.218072 | 0.434751 | 0.848279 | 1.000000 |

**1-2 : Fixed Acidity(산도)와 PH(산성도)은 높은(-0.68)의 음의 상관관계를 가진다.**

**1-2 : Fixed Acidity와 Citric acid(신맛을 내는 산도)는 높은 양의 상관관계를 가진다.**

**1-2 : Quality는 3 ~ 8등급까지 존재하며, Class는 0 ~ 1이 존재. 둘은 높은 양의 상관관계(0.84)를 갖는다.**

In [17]:

*#히트맵 -> 상관도 시각화*

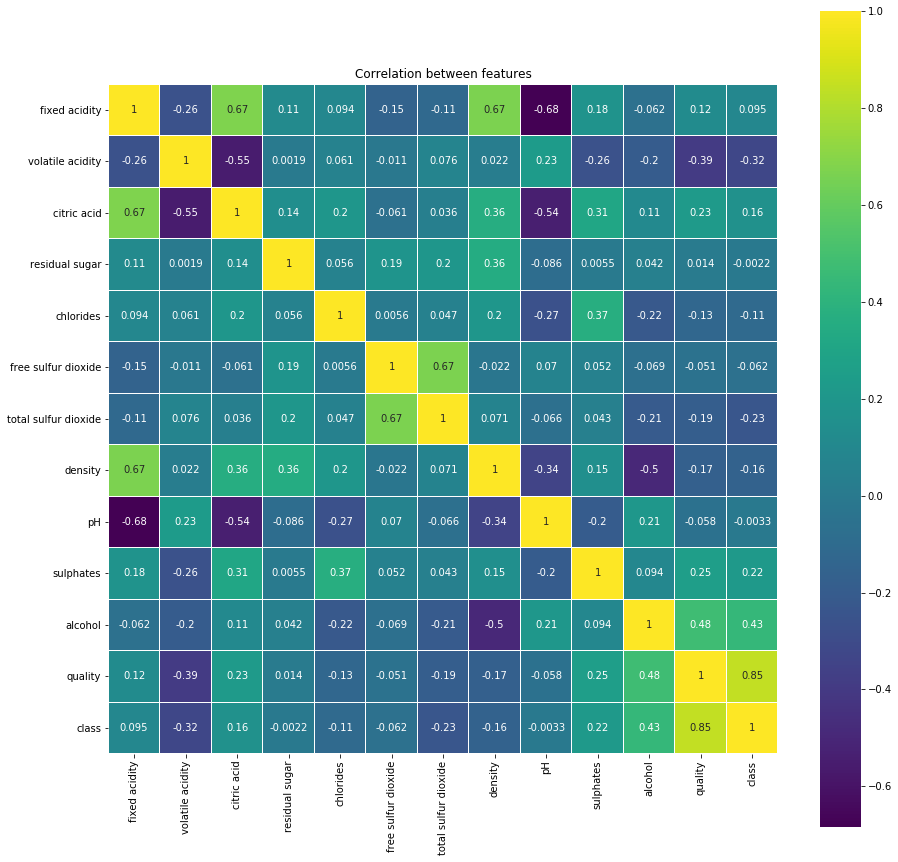
plt.figure(figsize=(15, 15))

sns.heatmap(wine\_data.corr(), linewidths=0.01, square=**True**,

annot=**True**, cmap=plt.cm.viridis, linecolor="white")

plt.title('Correlation between features')

plt.show()



**학습 피처 : fixed acidity, citric acid, residual sugar, density, pH, alcohol, class**

**정답 레이블 : quality**

In [18]:

x = wine\_data.loc[:,['fixed acidity','citric acid', 'residual sugar', 'density', 'pH', 'alcohol', 'class']] *#피처 = 파라미터*

In [19]:

y= wine\_data.loc[:,['quality']] *#피처 = 파라미터*

In [20]:

x

Out[20]:

|  | **fixed acidity** | **citric acid** | **residual sugar** | **density** | **pH** | **alcohol** | **class** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 11.6 | 0.66 | 2.20 | 1.00080 | 3.25 | 9.0 | 0 |
| **1** | 10.4 | 0.49 | 2.10 | 0.99940 | 3.16 | 8.4 | 0 |
| **2** | 7.4 | 0.00 | 4.25 | 0.99660 | 3.63 | 10.7 | 0 |
| **3** | 10.4 | 0.42 | 1.50 | 0.99832 | 3.38 | 9.9 | 0 |
| **4** | 8.3 | 0.02 | 3.40 | 0.99892 | 3.48 | 11.0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **1594** | 8.6 | 0.39 | 1.80 | 0.99516 | 3.35 | 11.7 | 1 |
| **1595** | 5.5 | 0.03 | 1.80 | 0.99080 | 3.50 | 14.0 | 1 |
| **1596** | 7.2 | 0.33 | 1.70 | 0.99600 | 3.23 | 10.0 | 1 |
| **1597** | 7.2 | 0.31 | 2.00 | 0.99472 | 3.23 | 11.3 | 1 |
| **1598** | 7.4 | 0.30 | 1.80 | 0.99419 | 3.24 | 11.4 | 1 |

1599 rows × 7 columns

In [21]:

y

Out[21]:

|  | **quality** |
| --- | --- |
| **0** | 3 |
| **1** | 3 |
| **2** | 3 |
| **3** | 3 |
| **4** | 3 |
| **...** | ... |
| **1594** | 8 |
| **1595** | 8 |
| **1596** | 8 |
| **1597** | 8 |
| **1598** | 8 |

1599 rows × 1 columns

In [22]:

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y, test\_size = 0.2, train\_size = 0.8, shuffle = **True**)

*#트레인(머신러닝 학습), 테스트(모델에 입증 데이터), 트레인 검증(머신러닝 학습 정답 데이터), 테스트 검증(모델이 맞춰야 할 데이터)*

**1-3 : 머신러닝 훈련 데이터 Split**

In [23]:

x\_train *#머신러닝 훈련 데이터*

Out[23]:

|  | **fixed acidity** | **citric acid** |  | **residual sugar** | **density** | **pH** | **alcohol** | **class** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1429** | 10.4 | 0.63 |  | 2.8 | 0.99980 | 3.26 | 11.2 | 1 |
| **1253** | 8.2 | 0.10 |  | 2.1 | 0.99638 | 3.29 | 10.9 | 1 |
| **1202** | 7.0 | 0.12 |  | 1.8 | 0.99588 | 3.22 | 9.5 | 1 |
| **853** | 12.4 | 0.58 |  | 3.0 | 1.00080 | 3.16 | 11.5 | 1 |
| **453** | 9.9 | 0.41 |  | 2.3 | 0.99820 | 3.21 | 9.5 | 0 |
| **...** | ... | ... |  | ... | ... | ... | ... | ... |
| **1486** | 10.2 | 0.42 |  | 2.0 | 0.99566 | 3.14 | 11.1 | 1 |
| **1376** | 5.4 | 0.09 |  | 1.7 | 0.99402 | 3.67 | 11.6 | 1 |
| **845** | 10.3 | 0.42 |  | 2.8 | 0.99900 | 3.28 | 9.5 | 1 |
| **1172** | 10.4 | 0.45 |  | 2.0 | 0.99774 | 3.22 | 11.4 | 1 |
| **1128** | 9.3 | 0.36 |  | 1.7 | 0.99702 | 3.27 | 10.9 | 1 |

1279 rows × 7 columns

In [24]:

y\_train *#머신러닝 훈련 데이터 정답*

Out[24]:

|  | **quality** |
| --- | --- |
| **1429** | 7 |
| **1253** | 6 |
| **1202** | 6 |
| **853** | 6 |
| **453** | 5 |
| **...** | ... |
| **1486** | 7 |
| **1376** | 6 |
| **845** | 6 |
| **1172** | 6 |
| **1128** | 6 |

1279 rows × 1 columns

In [25]:

x\_test *#모델 테스트 데이터*

Out[25]:

|  | **fixed acidity** | **citric acid** | **residual sugar** | **density** | **pH** | **alcohol** | **class** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **717** | 8.0 | 0.30 | 1.9 | 0.99717 | 3.39 | 9.0 | 0 |
| **463** | 7.3 | 0.10 | 2.6 | 0.99562 | 3.30 | 10.5 | 0 |
| **756** | 7.6 | 0.15 | 2.8 | 0.99550 | 3.17 | 10.2 | 1 |
| **837** | 10.3 | 0.44 | 2.4 | 0.99940 | 3.19 | 9.5 | 1 |
| **291** | 13.0 | 0.65 | 2.6 | 0.99960 | 3.05 | 10.6 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **646** | 8.2 | 0.20 | 1.4 | 0.99460 | 3.11 | 10.0 | 0 |
| **446** | 7.2 | 0.00 | 3.0 | 0.99586 | 3.43 | 10.0 | 0 |
| **1092** | 8.5 | 0.27 | 1.9 | 0.99518 | 3.16 | 11.1 | 1 |
| **386** | 8.1 | 0.23 | 2.6 | 0.99700 | 3.37 | 11.3 | 0 |
| **576** | 9.9 | 0.19 | 5.8 | 0.99878 | 3.14 | 9.4 | 0 |

320 rows × 7 columns

In [26]:

y\_test *#모델 테스트 데이터의 정답*

Out[26]:

|  | **quality** |
| --- | --- |
| **717** | 5 |
| **463** | 5 |
| **756** | 6 |
| **837** | 6 |
| **291** | 5 |
| **...** | ... |
| **646** | 5 |
| **446** | 5 |
| **1092** | 6 |
| **386** | 5 |
| **576** | 5 |

320 rows × 1 columns

**1-4 : KNN 최근접 방식으로 머신러닝 학습화 및 Label 예측**

In [31]:

*#Knn 최근접 방식으로 머신러닝 학습화, 근처 이웃 7개 지정*

knn = KNeighborsClassifier(n\_neighbors=7)

*#훈련 데이터 -> 훈련 검증 데이터*

knn.fit(x\_train, y\_train)

*#KNN 변수에 x\_train(학습) 데이터로 fit 시켜 y\_train의 정답을 맞추도록 학습화시킴*

C:\Program Files\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

Out[31]:

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=7, p=2,

weights='uniform')

In [32]:

y\_pred = knn.predict(x\_test)

*#테스트 데이터 -> 테스트 데이터(x\_test) 를 입력시켜, y\_pred 안에 학습화 된 모델의 정답을 예측*

In [34]:

y\_pred *#x\_test를 통해 -> 머신러닝이 예측한 와인의 Quality 정답 데이터*

Out[34]:

array([5, 5, 6, 6, 5, 6, 5, 6, 5, 6, 6, 5, 5, 6, 6, 6, 6, 7, 6, 5, 5, 7,

6, 5, 5, 7, 7, 5, 6, 6, 6, 5, 6, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6,

5, 6, 5, 5, 5, 5, 7, 6, 5, 5, 6, 5, 6, 5, 6, 6, 5, 6, 6, 7, 5, 5,

6, 6, 5, 6, 5, 6, 6, 5, 5, 7, 5, 6, 6, 5, 6, 5, 6, 5, 6, 6, 6, 6,

5, 5, 5, 7, 6, 6, 5, 6, 6, 6, 5, 5, 6, 5, 5, 6, 5, 7, 5, 6, 5, 7,

5, 6, 5, 7, 5, 5, 5, 5, 6, 6, 5, 6, 6, 5, 5, 6, 5, 6, 5, 5, 7, 6,

5, 6, 6, 6, 5, 5, 6, 5, 6, 5, 5, 6, 6, 6, 7, 5, 6, 7, 5, 5, 5, 6,

6, 6, 7, 7, 5, 6, 5, 5, 5, 5, 6, 5, 6, 5, 5, 6, 5, 5, 5, 7, 5, 7,

6, 6, 6, 5, 6, 5, 5, 6, 5, 7, 5, 5, 6, 5, 6, 5, 6, 5, 6, 5, 5, 6,

5, 7, 5, 5, 5, 5, 5, 7, 5, 5, 6, 5, 7, 6, 6, 5, 5, 5, 6, 6, 5, 6,

6, 5, 6, 6, 5, 6, 5, 5, 6, 5, 6, 6, 6, 5, 7, 6, 5, 7, 5, 5, 6, 6,

5, 5, 7, 5, 7, 6, 7, 5, 6, 5, 5, 5, 5, 5, 6, 6, 7, 6, 7, 5, 5, 5,

6, 6, 6, 5, 6, 6, 6, 6, 6, 6, 5, 6, 6, 7, 5, 6, 5, 6, 6, 6, 7, 7,

5, 6, 5, 5, 6, 5, 6, 5, 7, 7, 6, 6, 6, 5, 5, 6, 6, 6, 6, 6, 6, 6,

5, 5, 6, 5, 6, 6, 5, 5, 5, 6, 5, 5], dtype=int64)

In [36]:

*#3. 내 훈련 데이터 정답 = y\_test와, 내 훈련(train) 데이터로 학습된 정답(y\_pred) 의 정답률 비교*

print('정답률= ', accuracy\_score(y\_test, y\_pred)) *#y\_test = 머신러닝 학습화 모델이 맞출 문제들, y\_pred = 훈련 모델이 학습하여 y\_test를 넣어 나온 답안 = y\_pred*

정답률= 0.821875

**1-4 : 와인 소몰리에 테스트**

In [38]:

*#와인 소몰리에 테스트 :/ fixed acidity, citric acid, residual sugar, density, pH, alcohol, class 를 넣어 예측해보자*

white = [8.3, 0.12, 1.7, 0.995, 3.22, 9.5, 1]

test = knn.predict\_proba([white])

*#와인 테스트*

In [40]:

print(test)

*#4번째 인덱스에 정확도 1을 보임.*

*#4번째 인덱스의 Quallity는 3,4,5,6,7,8 이기때문에 6등급일 확률이 100%이다.*

[[0. 0. 0. 1. 0. 0.]]

In [44]:

*#와인 소몰리에 테스트 :/ fixed acidity, citric acid, residual sugar, density, pH, alcohol, class 를 넣어 예측해보자*

red = [1, 1, 1.7, 0.995, 3.22, 9.5, 1]

test = knn.predict\_proba([red])

print(test)

*#와인 테스트*

[[0. 0.28571429 0.71428571 0. 0. 0. ]]

In [47]:

*#red의 등급 테스트는 4 등급 or 5등급이다. 허나, 5등급이 70%의 확률로 높다.*

**2번 : BMI : -----------------------------------------------------------------------------------------------**

**2. 키와 몸무게를 활용한 비만도(BMI) 예측**

In [305]:

bmi\_data = pd.read\_csv('bmi.csv', encoding='UTF-8')

In [306]:

bmi\_data

Out[306]:

|  | **height** | **weight** | **label** |
| --- | --- | --- | --- |
| **0** | 185 | 45 | 1 |
| **1** | 167 | 37 | 1 |
| **2** | 191 | 79 | 2 |
| **3** | 122 | 73 | 3 |
| **4** | 182 | 62 | 2 |
| **...** | ... | ... | ... |
| **19995** | 157 | 55 | 2 |
| **19996** | 170 | 64 | 2 |
| **19997** | 144 | 60 | 3 |
| **19998** | 184 | 42 | 1 |
| **19999** | 133 | 47 | 3 |

20000 rows × 3 columns

In [307]:

bmi\_data.describe() *#비만도 데이터 묘사*

Out[307]:

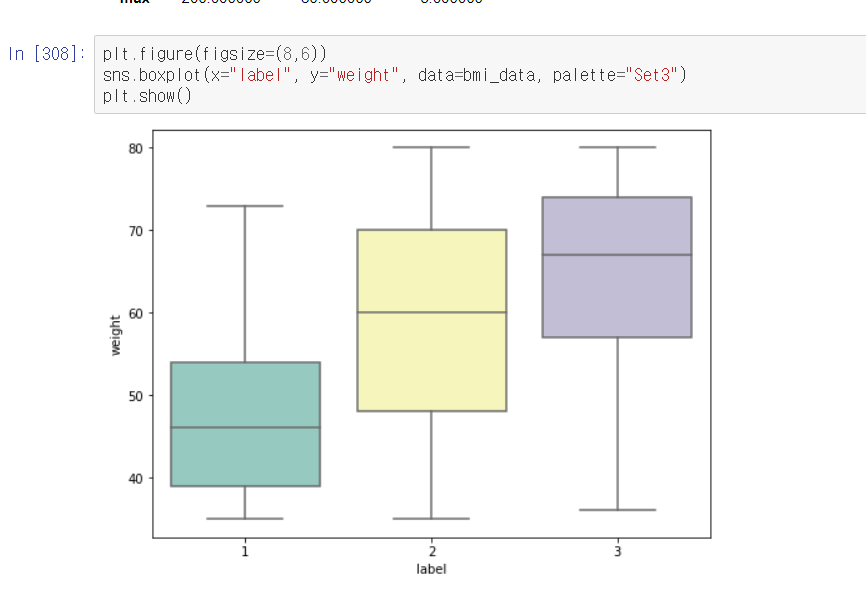
|  | **height** | **weight** | **label** |
| --- | --- | --- | --- |
| **count** | 20000.000000 | 20000.000000 | 20000.000000 |
| **mean** | 159.928600 | 57.532850 | 2.064600 |
| **std** | 23.471296 | 13.264759 | 0.837294 |
| **min** | 120.000000 | 35.000000 | 1.000000 |
| **25%** | 140.000000 | 46.000000 | 1.000000 |
| **50%** | 160.000000 | 57.000000 | 2.000000 |
| **75%** | 180.000000 | 69.000000 | 3.000000 |
| **max** | 200.000000 | 80.000000 | 3.000000 |

In [308]:

plt.figure(figsize=(8,6))

sns.boxplot(x="label", y="weight", data=bmi\_data, palette="Set3")

plt.show()



In [309]:

*#1. label 1,2,3 의 '몸무게' 4분위수 차이*

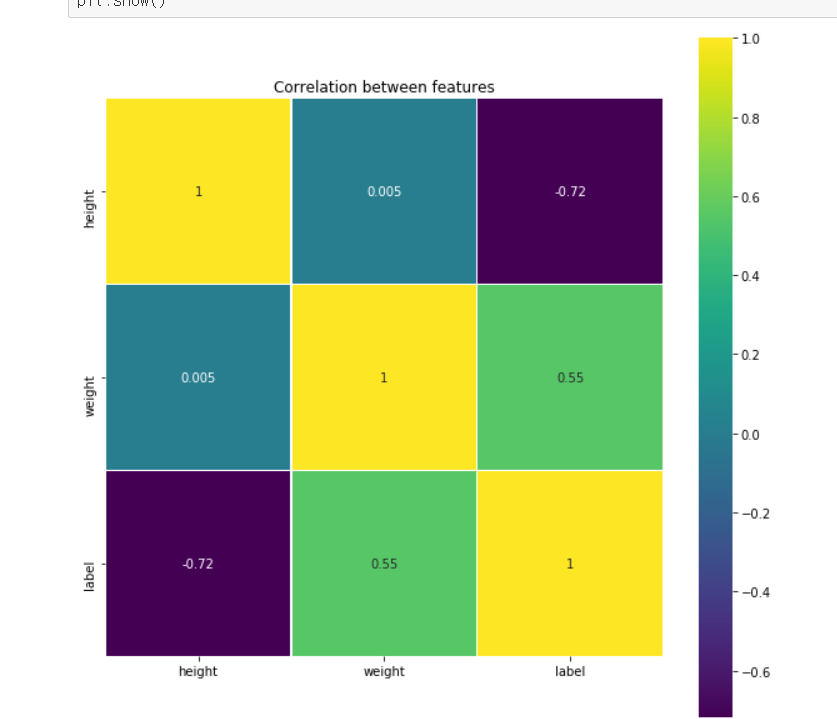
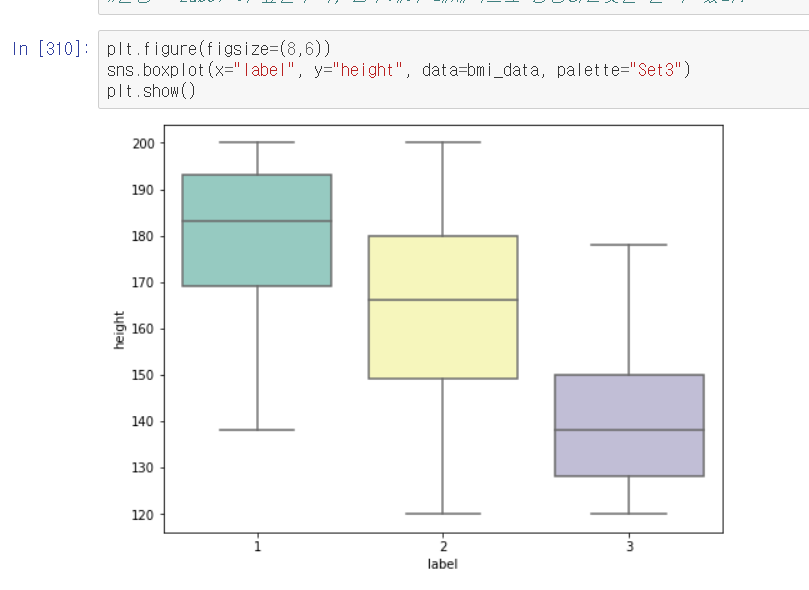
*#설명 : Label 이 높을수록, 몸무게가 대체적으로 상승하는것을 볼 수 있다.*

In [310]:

plt.figure(figsize=(8,6))

sns.boxplot(x="label", y="height", data=bmi\_data, palette="Set3")

plt.show()



In [311]:

*# 2. label 1,2,3 따라 '키'의 차이*

*# 설명 : 몸무게와는 반대되는 양상을 보이고 있다.*

*# 종합설명 : Label 이 높을수록 대체적으로 몸무게는 높고, 키는 작다.*

In [312]:

*#히트맵 -> 상관도 시각화*

plt.figure(figsize=(10, 10))

sns.heatmap(bmi\_data.corr(), linewidths=0.01, square=**True**,

annot=**True**, cmap=plt.cm.viridis, linecolor="white")

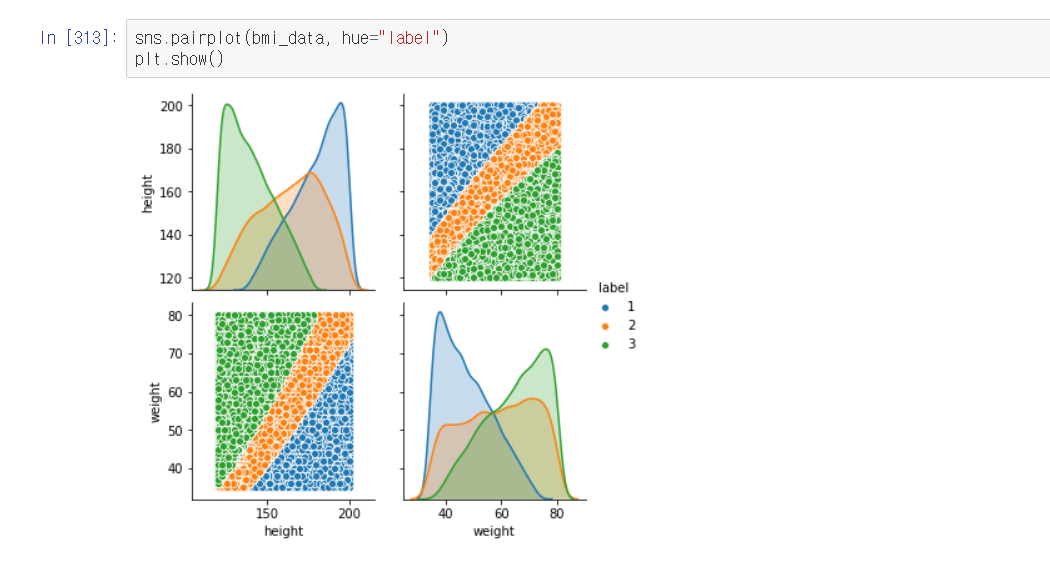
plt.title('Correlation between features')

plt.show()

In [313]:

sns.pairplot(bmi\_data, hue="label")

plt.show()



**1. 키와 라벨은 강한 음의 상관관계 -> 키가 클수록 라벨은 낮아짐**

**2. 키와 몸무게는 상관관계가 낮다.**

**3. 몸무게와 라벨은 보통의 상관관계를 갖는다.(0.5)**

In [314]:

x2 = bmi\_data.loc[:,['weight', 'height']] *#피처 = 파라미터*

y2 = bmi\_data.loc[:,['label']]

In [315]:

x2 *#학습 및 테스트 데이터*

Out[315]:

|  | **weight** | **height** |
| --- | --- | --- |
| **0** | 45 | 185 |
| **1** | 37 | 167 |
| **2** | 79 | 191 |
| **3** | 73 | 122 |
| **4** | 62 | 182 |
| **...** | ... | ... |
| **19995** | 55 | 157 |
| **19996** | 64 | 170 |
| **19997** | 60 | 144 |
| **19998** | 42 | 184 |
| **19999** | 47 | 133 |

20000 rows × 2 columns

In [316]:

y2 *#정답데이터*

Out[316]:

|  | **label** |
| --- | --- |
| **0** | 1 |
| **1** | 1 |
| **2** | 2 |
| **3** | 3 |
| **4** | 2 |
| **...** | ... |
| **19995** | 2 |
| **19996** | 2 |
| **19997** | 3 |
| **19998** | 1 |
| **19999** | 3 |

20000 rows × 1 columns

**학습, 학습정답, 테스트, 테스트데이터 분류**

In [317]:

x\_train2, x\_test2, y\_train2, y\_test2 = train\_test\_split(x2,y2, test\_size = 0.2, train\_size = 0.8, shuffle = **True**)

*#트레인(머신러닝 학습), 테스트(모델에 입증 데이터), 트레인 검증(머신러닝 학습 정답 데이터), 테스트 검증(모델이 맞춰야 할 데이터)*

In [318]:

x\_train2

Out[318]:

|  | **weight** | **height** |
| --- | --- | --- |
| **8020** | 67 | 141 |
| **9219** | 39 | 158 |
| **6639** | 77 | 160 |
| **9049** | 62 | 155 |
| **9130** | 39 | 140 |
| **...** | ... | ... |
| **13309** | 68 | 136 |
| **14378** | 54 | 173 |
| **10709** | 72 | 126 |
| **7529** | 58 | 149 |
| **19643** | 79 | 160 |

16000 rows × 2 columns

In [319]:

x\_test2

Out[319]:

|  | **weight** | **height** |
| --- | --- | --- |
| **12330** | 37 | 148 |
| **4026** | 60 | 147 |
| **9955** | 70 | 184 |
| **11783** | 63 | 123 |
| **1741** | 78 | 144 |
| **...** | ... | ... |
| **362** | 62 | 166 |
| **9460** | 68 | 186 |
| **18998** | 48 | 164 |
| **9149** | 38 | 129 |
| **18722** | 62 | 132 |

4000 rows × 2 columns

In [320]:

y\_train2

Out[320]:

|  | **label** |
| --- | --- |
| **8020** | 3 |
| **9219** | 1 |
| **6639** | 3 |
| **9049** | 3 |
| **9130** | 2 |
| **...** | ... |
| **13309** | 3 |
| **14378** | 1 |
| **10709** | 3 |
| **7529** | 3 |
| **19643** | 3 |

16000 rows × 1 columns

In [321]:

y\_test2

Out[321]:

|  | **label** |
| --- | --- |
| **12330** | 1 |
| **4026** | 3 |
| **9955** | 2 |
| **11783** | 3 |
| **1741** | 3 |
| **...** | ... |
| **362** | 2 |
| **9460** | 2 |
| **18998** | 1 |
| **9149** | 2 |
| **18722** | 3 |

4000 rows × 1 columns

In [411]:

*#학습기(Estimator) 객체 생성, SVM 서포트 벡터 머신으로 선을 그려 비만도 등급(label) 을 예측함*

clf2 = SVC() *#객체생성 -> 선을 그릴 수 있는 (X,Y 파라미터 담을)*

*#clf 객체에 x\_train(머신러닝 학습 데이터), y\_train(머신러닝 정답 데이터) 를 학습시킴*

clf2.fit(x\_train2, y\_train2)

C:\Program Files\Anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

y = column\_or\_1d(y, warn=True)

Out[411]:

SVC(C=1.0, break\_ties=False, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3, gamma='scale', kernel='rbf',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

In [412]:

y\_pred2 = clf2.predict(x\_test2)

*#테스트 데이터 -> 테스트 데이터(x\_test) 를 입력하여, y\_pred 는 객체안에 자동차의 타입을 예측한 결과를 가지고 있음*

In [413]:

y\_pred2

Out[413]:

array([1, 3, 2, ..., 1, 2, 3], dtype=int64)

In [414]:

*#3. 내 훈련 데이터 정답 = y\_test와, 내 훈련(train) 데이터로 학습된 정답(y\_pred) 의 정답률 비교*

print('정답률= ', accuracy\_score(y\_test2, y\_pred2)) *#y\_test = 내가 맞춰야할 답안, y\_pred = 훈련 모델로 학습되서 y\_test를 넣어 나온 답안 = y\_pred*

정답률= 0.99425

In [415]:

x\_test2

Out[415]:

|  | **weight** | **height** |
| --- | --- | --- |
| **12330** | 37 | 148 |
| **4026** | 60 | 147 |
| **9955** | 70 | 184 |
| **11783** | 63 | 123 |
| **1741** | 78 | 144 |
| **...** | ... | ... |
| **362** | 62 | 166 |
| **9460** | 68 | 186 |
| **18998** | 48 | 164 |
| **9149** | 38 | 129 |
| **18722** | 62 | 132 |

4000 rows × 2 columns

In [421]:

jiheon = [60, 180]

name = [30,140]

clf2.predict\_proba([jiheon])

*#왜 안되는지 모르겠음*

**---------------------------------------------------------------------------**

**AttributeError** Traceback (most recent call last)

**<ipython-input-421-bea27cee331a>** in <module>

1 jiheon **=** **[60,** **180]**

2 name **=** **[30,140]**

**----> 3** clf2**.**predict\_proba**([**jiheon**])**

4 **#왜 안되는지 모르겠음**

**C:\Program Files\Anaconda3\lib\site-packages\sklearn\svm\\_base.py** in predict\_proba**(self)**

634 datasets**.**

635 """

**--> 636** self**.**\_check\_proba**()**

637 **return** self**.**\_predict\_proba

638

**C:\Program Files\Anaconda3\lib\site-packages\sklearn\svm\\_base.py** in \_check\_proba**(self)**

601 **def** \_check\_proba**(**self**):**

602 **if** **not** self**.**probability**:**

**--> 603 raise AttributeError("predict\_proba is not available when "**

604 " probability=False")

605 **if** self**.**\_impl **not** **in** **('c\_svc',** **'nu\_svc'):**

**AttributeError**: predict\_proba is not available when probability=False

In [330]:

clf2.predict([name])

Out[330]:

array([1], dtype=int64)

**3. 목적 : School Data로 Label ( 1, 0 )Classfication**

In [360]:

school\_data = pd.read\_csv('school.csv', encoding='UTF-8') *#학교 데이터 불러오기*

In [361]:

school\_data

Out[361]:

|  | **teaching** | **international** | **research** | **citations** | **income** | **num\_students** | **student\_staff\_ratio** | **international\_students** | **total\_score** | **Label** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 12.4 | 95.6 | 10.6 | 13.3 | 33.3 | 5226 | 14.1 | 0.82 | 59.8 | 0 |
| **1** | 29.9 | 90.1 | 20.1 | 65.3 | 33.9 | 2473 | 15.6 | 0.63 | 59.8 | 0 |
| **2** | 24.7 | 82.6 | 18.9 | 43.5 | 28.4 | 2473 | 15.6 | 0.63 | 59.8 | 0 |
| **3** | 61.3 | 98.6 | 67.5 | 94.6 | 65.4 | 9666 | 10.5 | 0.54 | 76.1 | 1 |
| **4** | 62.4 | 98.8 | 57.0 | 95.0 | 49.8 | 9666 | 10.5 | 0.54 | 73.0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2598** | 36.7 | 72.7 | 38.8 | 71.6 | 33.9 | 0 | 0.0 | 0.00 | 50.4 | 0 |
| **2599** | 46.2 | 27.7 | 39.6 | 59.7 | 77.1 | 0 | 0.0 | 0.00 | 47.7 | 0 |
| **2600** | 43.3 | 52.0 | 49.5 | 41.7 | 99.9 | 3879 | 4.6 | 0.00 | 46.2 | 0 |
| **2601** | 43.9 | 23.9 | 38.9 | 56.6 | 87.8 | 0 | 0.0 | 0.00 | 45.8 | 0 |
| **2602** | 49.0 | 32.1 | 14.2 | 63.5 | 40.7 | 0 | 0.0 | 0.00 | 41.5 | 0 |

2603 rows × 10 columns

In [362]:

school\_data.corr()

Out[362]:

|  | **teaching** | **international** | **research** | **citations** | **income** | **num\_students** | **student\_staff\_ratio** | **international\_students** | **total\_score** | **Label** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **teaching** | 1.000000 | 0.137571 | 0.913120 | 0.503867 | 0.306127 | 0.043559 | -0.270348 | 0.273349 | 0.585123 | 0.762519 |
| **international** | 0.137571 | 1.000000 | 0.250562 | 0.324722 | 0.012195 | -0.141406 | 0.060470 | 0.790759 | 0.081301 | 0.184758 |
| **research** | 0.913120 | 0.250562 | 1.000000 | 0.525264 | 0.357491 | 0.065745 | -0.182574 | 0.334674 | 0.569877 | 0.767877 |
| **citations** | 0.503867 | 0.324722 | 0.525264 | 1.000000 | 0.042822 | -0.071660 | -0.129201 | 0.309218 | 0.251363 | 0.485385 |
| **income** | 0.306127 | 0.012195 | 0.357491 | 0.042822 | 1.000000 | -0.031859 | -0.005950 | 0.032428 | 0.148350 | 0.213245 |
| **num\_students** | 0.043559 | -0.141406 | 0.065745 | -0.071660 | -0.031859 | 1.000000 | 0.373442 | -0.126749 | 0.019004 | 0.047271 |
| **student\_staff\_ratio** | -0.270348 | 0.060470 | -0.182574 | -0.129201 | -0.005950 | 0.373442 | 1.000000 | -0.026489 | -0.137670 | -0.190075 |
| **international\_students** | 0.273349 | 0.790759 | 0.334674 | 0.309218 | 0.032428 | -0.126749 | -0.026489 | 1.000000 | 0.153235 | 0.269819 |
| **total\_score** | 0.585123 | 0.081301 | 0.569877 | 0.251363 | 0.148350 | 0.019004 | -0.137670 | 0.153235 | 1.000000 | 0.722390 |
| **Label** | 0.762519 | 0.184758 | 0.767877 | 0.485385 | 0.213245 | 0.047271 | -0.190075 | 0.269819 | 0.722390 | 1.000000 |

In [363]:

*#히트맵 -> 상관도 시각화*

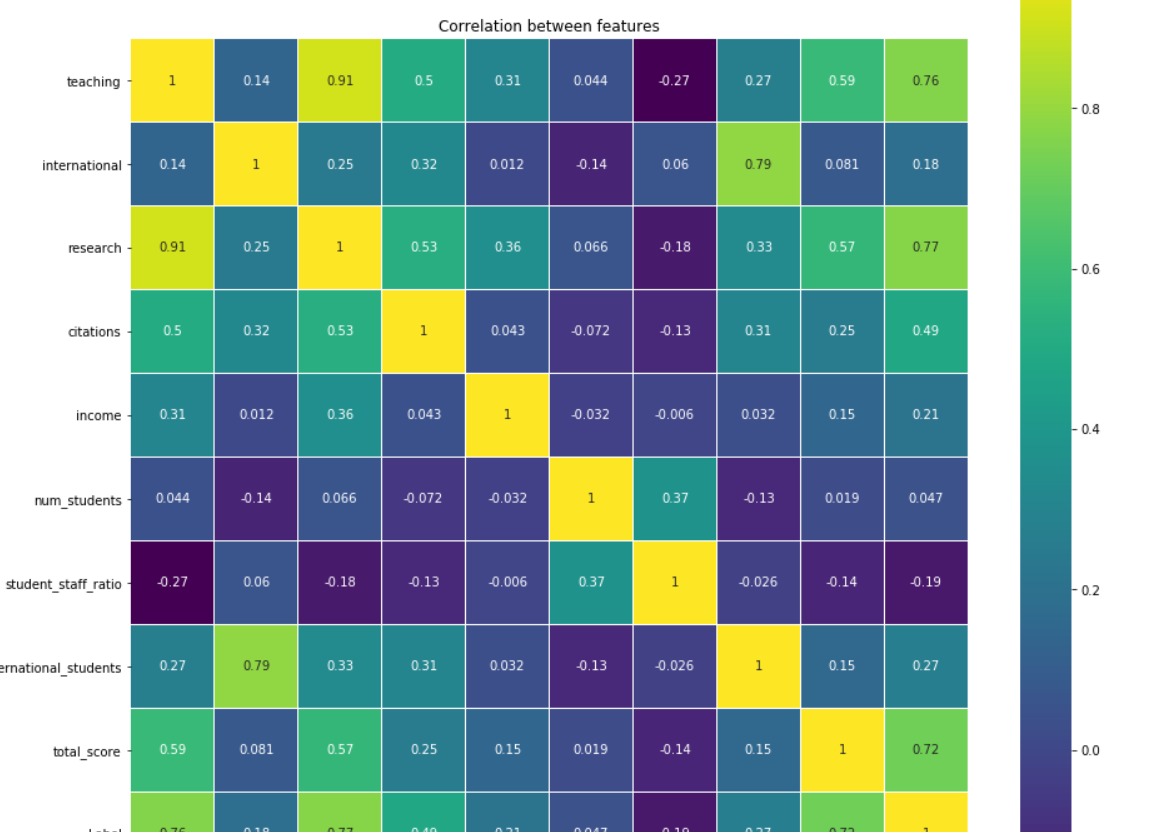
plt.figure(figsize=(15, 15))

sns.heatmap(school\_data.corr(), linewidths=0.01, square=**True**,

annot=**True**, cmap=plt.cm.viridis, linecolor="white")

plt.title('Correlation between features')

plt.show()



**1. Label 개/폐업 유무와 teaching(교육)의 상관도는 0.7로 높다.**

**-> teaching 이 증가할수록, 1에 가까워짐 -> 교육 점수가 좋을수록 학교는 존속(=1)**

**2. Label 과 관련된 상관도는 Reasearch, total\_score, citations 순으로 높다.**

**3. 피처는 label을 제외한 모든 피처(칼럼)으로 결정**

In [386]:

x3 = school\_data.iloc[:,0:9] *#피처 = 파라미터*

y3 = school\_data.loc[:,['Label']]

In [387]:

x3

Out[387]:

|  | **teaching** | **international** | **research** | **citations** | **income** | **num\_students** | **student\_staff\_ratio** | **international\_students** | **total\_score** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 12.4 | 95.6 | 10.6 | 13.3 | 33.3 | 5226 | 14.1 | 0.82 | 59.8 |
| **1** | 29.9 | 90.1 | 20.1 | 65.3 | 33.9 | 2473 | 15.6 | 0.63 | 59.8 |
| **2** | 24.7 | 82.6 | 18.9 | 43.5 | 28.4 | 2473 | 15.6 | 0.63 | 59.8 |
| **3** | 61.3 | 98.6 | 67.5 | 94.6 | 65.4 | 9666 | 10.5 | 0.54 | 76.1 |
| **4** | 62.4 | 98.8 | 57.0 | 95.0 | 49.8 | 9666 | 10.5 | 0.54 | 73.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2598** | 36.7 | 72.7 | 38.8 | 71.6 | 33.9 | 0 | 0.0 | 0.00 | 50.4 |
| **2599** | 46.2 | 27.7 | 39.6 | 59.7 | 77.1 | 0 | 0.0 | 0.00 | 47.7 |
| **2600** | 43.3 | 52.0 | 49.5 | 41.7 | 99.9 | 3879 | 4.6 | 0.00 | 46.2 |
| **2601** | 43.9 | 23.9 | 38.9 | 56.6 | 87.8 | 0 | 0.0 | 0.00 | 45.8 |
| **2602** | 49.0 | 32.1 | 14.2 | 63.5 | 40.7 | 0 | 0.0 | 0.00 | 41.5 |

2603 rows × 9 columns

In [388]:

y3 *#학교 존폐업 예측*

Out[388]:

|  | **Label** |
| --- | --- |
| **0** | 0 |
| **1** | 0 |
| **2** | 0 |
| **3** | 1 |
| **4** | 1 |
| **...** | ... |
| **2598** | 0 |
| **2599** | 0 |
| **2600** | 0 |
| **2601** | 0 |
| **2602** | 0 |

2603 rows × 1 columns

In [389]:

x\_train3, x\_test3, y\_train3, y\_test3 = train\_test\_split(x3,y3, test\_size = 0.2, train\_size = 0.8, shuffle = **True**)

*#트레인(머신러닝 학습), 테스트(모델에 입증 데이터), 트레인 검증(머신러닝 학습 정답 데이터), 테스트 검증(모델이 맞춰야 할 데이터)*

In [391]:

x\_train3

Out[391]:

|  | **teaching** | **international** | **research** | **citations** | **income** | **num\_students** | **student\_staff\_ratio** | **international\_students** | **total\_score** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **2180** | 28.2 | 47.1 | 25.5 | 55.5 | 34.2 | 9703 | 15.2 | 0.05 | 59.8 |
| **2420** | 40.7 | 29.1 | 10.4 | 63.3 | 32.4 | 7086 | 8.3 | 0.02 | 59.8 |
| **861** | 48.4 | 37.3 | 39.8 | 90.9 | 34.4 | 12338 | 4.5 | 0.18 | 57.4 |
| **993** | 18.9 | 68.7 | 16.1 | 35.0 | 34.5 | 27491 | 16.1 | 0.16 | 59.8 |
| **1909** | 69.5 | 27.5 | 69.5 | 58.2 | 78.7 | 22809 | 5.6 | 0.07 | 63.2 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **224** | 28.0 | 84.1 | 30.2 | 74.3 | 42.1 | 12938 | 15.8 | 0.33 | 47.1 |
| **919** | 22.4 | 76.9 | 21.9 | 66.8 | 30.0 | 32713 | 30.4 | 0.17 | 59.8 |
| **1766** | 61.8 | 21.1 | 56.5 | 40.0 | 75.0 | 23144 | 7.8 | 0.09 | 51.0 |
| **1851** | 18.9 | 20.5 | 28.1 | 46.2 | 60.8 | 7983 | 15.4 | 0.08 | 59.8 |
| **207** | 21.5 | 84.8 | 20.2 | 53.4 | 32.2 | 20314 | 36.5 | 0.33 | 59.8 |

2082 rows × 9 columns

In [393]:

x\_test3

Out[393]:

|  | **teaching** | **international** | **research** | **citations** | **income** | **num\_students** | **student\_staff\_ratio** | **international\_students** | **total\_score** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **723** | 16.3 | 64.2 | 18.8 | 55.1 | 31.7 | 8397 | 15.7 | 0.20 | 59.8 |
| **1374** | 23.5 | 40.6 | 18.4 | 28.7 | 52.8 | 7647 | 15.5 | 0.12 | 59.8 |
| **1477** | 23.6 | 50.0 | 17.1 | 62.6 | 48.0 | 35308 | 16.1 | 0.11 | 59.8 |
| **26** | 35.7 | 90.5 | 35.9 | 81.4 | 39.8 | 8338 | 12.7 | 0.47 | 53.6 |
| **2304** | 22.9 | 45.6 | 22.2 | 58.7 | 42.5 | 10045 | 9.5 | 0.04 | 59.8 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1930** | 20.2 | 31.3 | 15.0 | 36.2 | 40.3 | 14780 | 26.9 | 0.07 | 59.8 |
| **2008** | 31.9 | 29.6 | 14.2 | 65.3 | 41.7 | 23065 | 10.7 | 0.07 | 59.8 |
| **811** | 26.5 | 41.4 | 18.9 | 83.5 | 24.2 | 13216 | 17.4 | 0.19 | 42.4 |
| **1193** | 46.2 | 35.5 | 42.2 | 41.1 | 70.3 | 24043 | 15.8 | 0.14 | 59.8 |
| **426** | 36.4 | 54.7 | 21.9 | 81.3 | 34.0 | 18539 | 15.1 | 0.26 | 46.8 |

521 rows × 9 columns

In [410]:

*#의사결정 라이브러리*

*#머신러닝 학습데이터 -> X\_train -> y\_train(존폐업유무) / Decision Treee로 학습화시켜, 0과 1로 살 여부를 맞출 분류 및 예측*

**from** **sklearn.tree** **import** DecisionTreeClassifier

school\_clf = DecisionTreeClassifier(max\_depth=5, random\_state=13)

school\_clf.fit(x\_train3, y\_train3)

print('모델 정확도: **{}**'.format(school\_clf.score(x\_train3, y\_train3)))

*#의사결정 분류방법으로, 학습화시켜 학교가 0 or 1*

모델 정확도: 1.0

In [409]:

*#정확도 테스트*

y\_pred3 = school\_clf.predict(x\_test3) *#tree\_clf 학습 모델 객체안에 X\_test라는 내가 빼놓은 데이터를 넣어, y\_pred(살 여부를 확인함!)*

print("모델 -> 테스트데이터 정답률 계산", accuracy\_score(y\_test3, y\_pred3)\*100) *#y\_test 는 내가 빼놓은 정답 데이터, (y\_test / y\_pred) \* 100 =*

모델 -> 테스트데이터 정답률 계산 100.0

In [406]:

**from** **sklearn.tree** **import** export\_graphviz

export\_graphviz(

school\_clf,

out\_file="school.dot",

feature\_names=['teaching','international','research','citations','income','num\_students','student\_staff\_ratio','international\_students','total\_score'],

class\_names=['0', '1'],

rounded=**True**,

filled=**True**

)

**import** **graphviz**

**with** open("school.dot") **as** f:

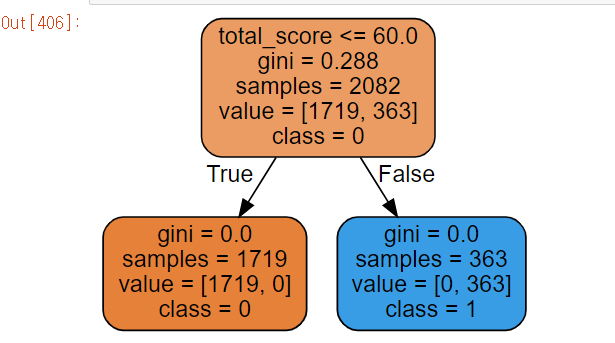
dot\_graph = f.read()

dot = graphviz.Source(dot\_graph)

dot.format = 'png'

dot.render(filename='school', directory='images/decision\_trees', cleanup=**True**)

dot

Out[406]:

total\_score <= 60.0gini = 0.288samples = 2082value = [1719, 363]class = 0gini = 0.0samples = 1719value = [1719, 0]class = 0Truegini = 0.0samples = 363value = [0, 363]class = 1False

In [408]:

*#0과1 Classification*

In [ ]:

2. 회귀분석의 개념  
관찰된 연속형 변수들에 대해 두 변수 사이의 모형을 구한뒤 적합도를 측정해 내는 분석 방법이다. 단순 선형 회귀분석을 예로 들자면, 독립변수 X 에 대하여 Y 종속변수 들 사이의 관계를 수학적 모형을 이용하여 규명하는 것으로서, 규명된 함수식을 이용하여 설명변수들의 변화로부터 종속변수의 변화를 ‘예측’ 하는 것이다.  
 쉽게 설명하자면, 회귀분석은 기존에 X(독립변수)를 통해 Y(종속변수)의 결과를 선형 모형으로 표현해놓고, 다른 미지의 Xn 변수를 대입하였을떄 Yn의 결과를 ‘예측’ 하는 것   
2-1. 회귀분석의 종류

단순선형 회귀 분석 : 단순선형 회귀는 종속변수와 독립변수 사이의 관계를 설명한 것  
다중선형 회귀 분석 : 종속적 결과변수가 2개 이상의 독립변수에 의해 예측되는 회귀분석을 다중선형 회귀 분석이라 한다. 독립변수는 연속성이거나 범주형 모두 가능함.  
  
로지스틱 회귀 분석 : 이분화된 항목에 한하여 사용하는 방법  
Cox 비례 회귀 분석 : 생존이나 그 외에, 시간에 관계된 사건들에 영향을 미치는 변수를 결정하기 위해 사용함. 결과측정에 사용된 시간은 실제 생존시간이 아니며 위험비의 개념이 사용된다.  
2-2. 회귀분석의 사용 예제  
결국, 회귀분석이란 어떤 데이터를 넣어 결과를 예측하기 위함이다. x는 데이터이고, y는 결과이다. 예를 들어, 국민 전체를 대상으로 행복지수를 구하기 위한 조건과 회귀분석모델이 존재한다고 가정하자. 판단 조건은(=벡터,데이터,피처) 삶의 만족도와 연소득 x 조건을 주고, y는 행복지수이다.

Ex) a의 연소득은 1억, 삶의만족도는 70 = 170 이라는 가정의 행복지수가 나옴.  
 b의 연소득은 8천, 삶의 만족도는 90= 170 이라는 가정의 행복지수가 나옴.  
  
위와 같은 회귀분석 모델이 있을때, 다양한 사람의 행복지수를 ‘예측’ 하기 위해선 그들의 연소득과 삶의 만족도를 조건으로 행복지수를 ‘예측’ 해 볼 수 있다.  
 회귀분석이란 결국 다량의 데이터를 주입시켜, 수학적 선형 모델로 목적을 예측할 수 있는 방법이라 할 수 있다.  
   
  
  
  
  
3. 몽고 DB를 사용하는 이유  
데이터는 항사 정형화되어 나누어진 것이 아니다. 우리가 사는 현실세계는 숫자와 문자로만 이루어진 것이 아니다. 영상,문서처럼 형태와 구조화 되지 않은 것을 비정형 데이터라 한다. 비정형 데이터의 사례로는 책, 잡지, 문서의료기록, 음성정보, 영상정보와 같은 전통적ㅇ니 데이터외에 이메일, 트위터, 블로그처럼 모바일기기와 온라인상에서 생성되는 데이터들이 있다.  
 가장 대표적인 비정형데이터로는 ‘문서’ 가 있다. 문서에는 문자가 가장 많지만 숫자와 도표, 그림등이 있다. 문서정보는 정보의 관점에서 보면 유형이 불규칙하며 의미를 포함하기 어렵다. 하여, 몽고 DB를 사용하여 이러한 비정형 데이터를 분석하고 관리할 수 있다.  
 페이스북, 트위터, 인스타그램, 넷플릭스, 쇼핑몰 등에서 사진이나 메세지등을 저장하고 해당 컨텐츠에 맞게 추천해주는 것등에 있어서 NOSQL DB을 기반으로한 몽고 DB가 사용되고 있다.  
  
2-1. 몽고 사용 데이터 포맷

정답 : JSON! Key와 Value로 데이터를 구축함.  
  
몽고DB(MongoDB)는 NoSQL 중 가장 많이 쓰이는 비관계형 데이터베이스 관리시스템으로, 크로스 플랫폼 도큐먼트 지향 데이터베이스 시스템이다. ‘JSON’과 같은 동적 스키마형 문서들을 선호함에 따라 전통적인 테이블 기반 관계형 데이터베이스 구조의 사용을 삼간다. 이로써 특정한 종류의 애플리케이션을 더 쉽고 빠르게 데이터 통합을 가능케 한다. 아페로 GPL과 아파치 라이선스를 결합하여 공개된 몽고DB는 자유 오픈소스 소프트웨어다. 몽고DB는 NoSQL 데이터베이스이기 때문에 데이터 객체들이 컬렉션 내부에서 독립된 문서로 저장되는 문서 모델을 기반으로 한다.  
 NoSQL은 NoSQL은 전통적인 관계형 데이터베이스 관리 시스템(RDBMS)과는 다르게 설계된 비관계형 DBMS로, 대규모의 데이터를 유연하게 처리할 수 있는 것이 장점이다. 테이블-칼럼과 같은 스키마 없이, 분산 환경에서 단순 검색 및 추가 작업을 위한 키 값을 최적화하고, 지연(latency)과 처리율(throughput)이 우수하다. 게다가 대규모 확대가 가능한 수평적인 확장성의 특징을 가지고 있고 아파치 카산드라(Apache Cassandra), 하둡(Hadoop), 몽고DB(MongoDB) 등이 있다. RDBMS보다 덜 제한적인 일관성 모델을 이용하는 데이터의 저장 및 검색을 위한 매커니즘을 제공한다. 이러한 접근에 대한 동기에는 디자인의 단순화, 수평적 확장성, 세세한 통제를 포함한다. NoSQL은 빅데이터와 실시간 웹 애플리케이션의 상업적 이용에 널리 쓰인다.

몽고 DB의 특징 :   
Schemaless : 데이터베이스에서 말하는 스키마(Schema)는 데이터베이스를 구성하는 개체(Entity), 속성(Attribute), 관계(Relationship) 및 데이터 조작 시에 데이터값들이 갖는 제약조건 등에 관해 전반적으로 정의하는 것이다. 스키마가 존재한다는 것은 그 구조가 미리 정의되어 있어야 한다는 의미로, 이는 데이터의 급격한 변화에 대응하기가 힘들어진다는 것을 뜻한다. 몽고DB는 이러한 스키마가 사전에 정의되지 않아도 되며, 데이터베이스에 저장된 도큐먼트는 각기 다른, 다양한 필드를 가질 수 있다. 또한 각 필드는 서로 다른 데이터타입을 가질 수 있다.

MongoDB의 대표적인 특징

▶ Document-Oriented Storage

: 모든 데이터가 JSON 형태로 저장되며, 스키마가 없다.

▶ Full Index Support

: RDBMS에 뒤치지 않는 다양한 인덱싱을 제공한다.

▶ Replication & High Availability

: 데이터 복제를 통해 가용성을 향상시킬 수 있다.

▶ Auto-Sharing

: Primary key를 기반으로 여러 서버에 데이터를 나누는 scale-out이 가능하다.

▶ Querying

: key 기반의 get, put 뿐만 아니라 다양한 종류의 쿼리들을 제공한다.

▶ Fast In-Place Updates

: 고성능의 atomic operation을 지원한다.

▶ MapReduce

: 맵리듀스를 지원한다.

▶ GridFS

: 별도 스토리지 엔진을 통해 파일을 저장할 수 있다.

자유로운 Indexing

몽고DB는 가장 많이 쓰이는 MySQL에서 지원하는 지원하는 대부분의 인덱스를 지원한다. 지원하는 인덱스의 목록은 다음과 같다.

▶ Single Field Indexes

: 가장 기본적인 인덱스 타입

▶ Compound Indexes

: RDBMS에서 많이 쓰이는 복합 인덱스

▶ Multikey Indexes

: Array에 매칭되는 값이 하나라도 있으면 인덱스에 추가하는 멀티키 인덱스

▶ Geospatial Indexes and Queries

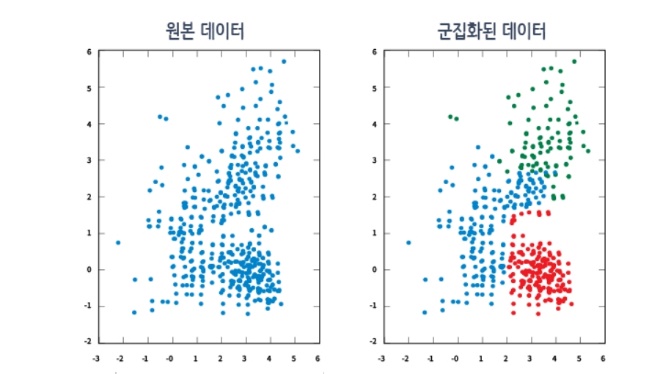
: 위치 기반 인덱스와 쿼리

▶ Text Indexes

: String 컨텐츠에도 인텍싱이 가능

▶ Hashed Index

: BTree 인덱스가 아닌 Hash 타입의 인덱스도 사용 가능  
  
2-2.몽고 DB 사용 사례  
 페이스북, 트위터, 인스타그램, 넷플릭스, 쇼핑몰 등에서 사진이나 메세지등을 저장하고 해당 컨텐츠에 맞게 추천해주는 것등에 있어서 NOSQL DB을 기반으로한 몽고 DB가 사용되고 있다.

3. 비지도 학습 클러스터링  
 비지도 학습과 비지도 학습의 가장 큰 차이점은 우선적으로, ‘정답=Label’이 있느냐 없느냐의 차이이다. 지도학습의 경우 데이터를 통해서 Label을 맞추도록 하는 방식으로 머신러닝을 구현한다.   
 하지만, 비지도 학습은 문제는 알려주되, 정답은 알려주지 않는다. 여러 문제를 직접 학습함으로써 해당 데이터의 패턴, 특성 및 구조를 스스로 파악하여 이를 통해 새로운 데이터에서 일정한 규칙을 찾게 되는 것이다.  
   
비지도 학습은 구체적인 결과에 대해 사전 지식은 없지만, 해당 결과 데이터를 통해 유의미한 지식을 얻고자 할 때 사용된다. 사람도 제대로 알 수 없는 본질적인 문제나 데이터에 숨겨진 특징이나 구조를 연구할때 사용된다.  
  
 y 없이 x만 이용해서 학습하는 것으로 정답이 없는 문제를 푸는 것으로 ‘학습’이 맞게 됐는지 확인할 길은 없지만, 대체적으로 모든 데이터가 레이블(정답) 이 없는 형태로 있기 때문에, 정답이 없는 경우에 사용한다.  
  
  


데이터가 분산화 되었을때, 레이블이 없다 하여도 데이터간 거리에 따라 군집을 나눌 수 있다. 이렇게 x만 가지고 군집을 학습하는 것을 군집화라고 한다.  
  
군집화는 레이블이 없는 학습 데이터들의 특징(Feature)을 분석하여 서로 동일하거나 유사한 특징을 가진 데이터끼리 그룹화함으로서 레이블이 없는 학습 데이터를 Clustering(군집화) 로 분류한다. 그리고, 새로운 데이터가 입력되면 지도 학습의 분류 모델처럼 학습한 군집을 가지고 해당 데이터가 어느 군집에 속하는지를 분석하는 것이다.  
  
**Clustering 타당성 평가**   
비지도 학습에 사용되는 데이터에는 Label(레이블)이 없으므로, 지도학습처럼 단순정확도를 지표로 그 정확도를 평가할 수는 없다.  
 군집을 만든 결과가 얼마만큼 타당한지는 군집간의 거리, 군집의 지름, 군집의 분산도 등을 종합적으로 고려하여 평가할 수 있다.  
 따라서, 일반적으로 군집간 분산(inter-cluster variance)이 최대가 되고, 군집 내 분산이 최소가 될 때 최적의 군집 모양과 개수로 판단할 수 있다.  
 **군집화 활용 분야**  
군집화는 의학 분야등 특정 질병에 대한 공간 군집 분석을 통해 질병의 분포면적과 확산 경로 등을 파악하는 역학조사에 활용되며, 홍보 분야에서는 고객을 세분화 할때 사용되곤 한다.