4주. Clustering, KNN			
학번	32183164	이름	이석현

BostonHousing 데이터셋은 보스턴 지역의 지역정보 및 평균주택 가격 (medv) 정보를 담고 있다.

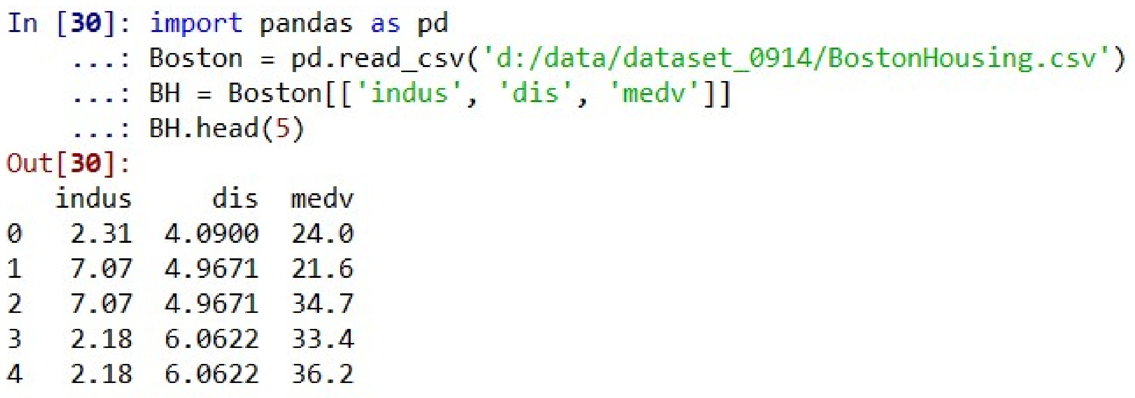
BostonHousing dataset에 대해 clustring을 실시하려고 한다.

Q1 Bostonhousing dataset에서 indus, dis, mdev 3개 변수(컬럼에 대한 데이터를 추출하고, 추출된 데이터에 대해 scaling을 하여 새로운 데이터셋 BH 를 생성하시오. (BH 의 앞 5개 행을 출력한다)

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
import pandas as pd
Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
BH = Boston[['indus', 'dis', 'medv']]
BH.head(5)
```

실행화면 캡쳐:



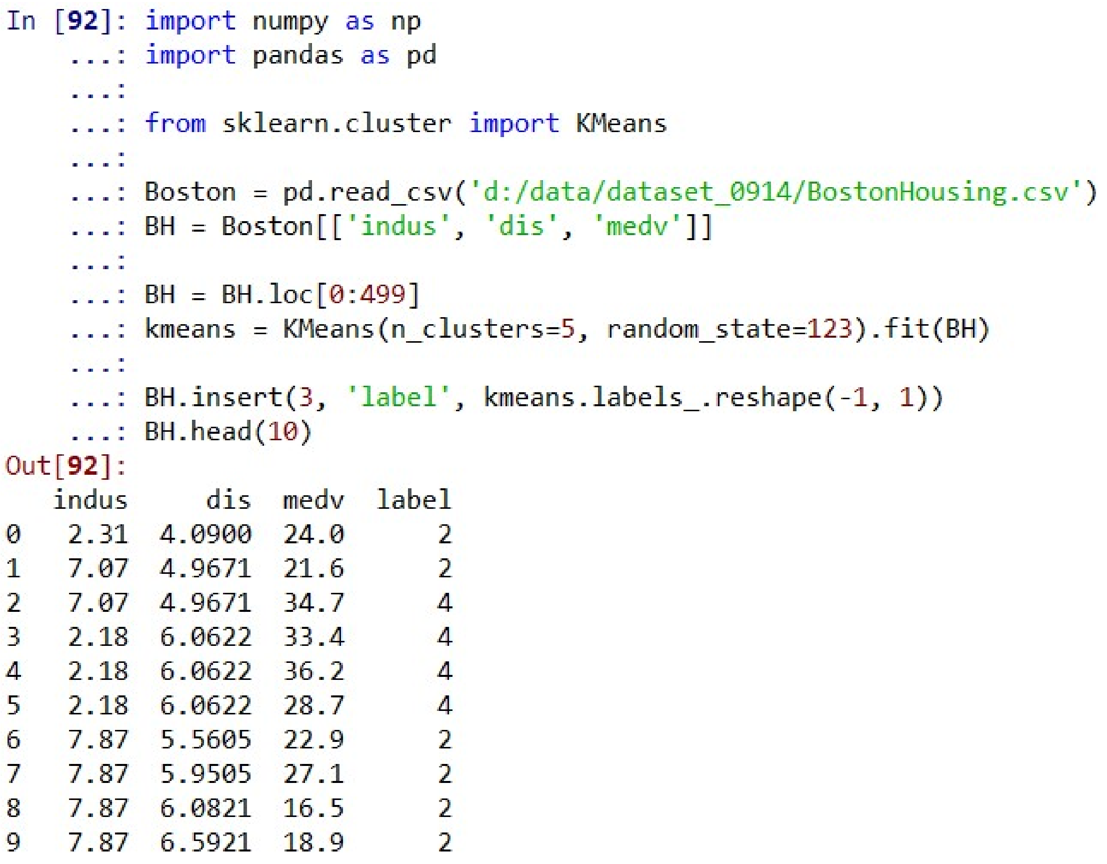
```
In [30]: import pandas as pd
    ...: Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
    ...: BH = Boston[['indus', 'dis', 'medv']]
    ...: BH.head(5)
Out[30]:
   indus     dis  medv
0   2.31  4.0900  24.0
1   7.07  4.9671  21.6
2   7.07  4.9671  34.7
3   2.18  6.0622  33.4
4   2.18  6.0622  36.2
```

Q2. BH 에 대해 KMeans 클러스터링을 실시하되 1~500행에 대해서만 실시하고, 클러스터의 개수는 5, random_state 의 값은 123 으로 하시오. 그리고 생성된 클러스터 값을 BH 에 추가하여 결과를 보이시오. (앞에서 10개의 행에 대해서만 결과를 보인다.)

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
BH = Boston[['indus', 'dis', 'medv']]
BH = BH.loc[0:499]
kmeans = KMeans(n_clusters=5, random_state=123).fit(BH)
BH.insert(3, 'label', kmeans.labels_.reshape(-1, 1))
BH.head(10)
```

실행화면 캡쳐:



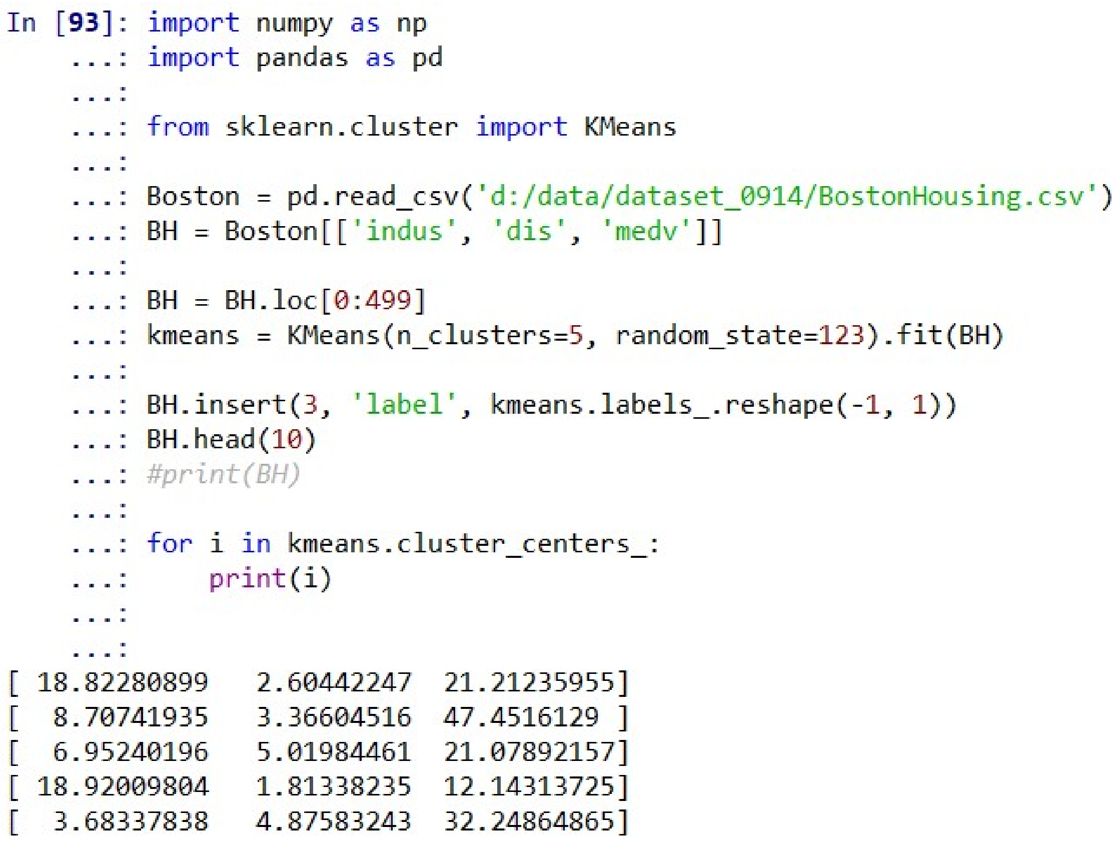
```
In [92]: import numpy as np
    ...: import pandas as pd
    ...:
    ...: from sklearn.cluster import KMeans
    ...:
    ...: Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
    ...: BH = Boston[['indus', 'dis', 'medv']]
    ...:
    ...: BH = BH.loc[0:499]
    ...: kmeans = KMeans(n_clusters=5, random_state=123).fit(BH)
    ...:
    ...: BH.insert(3, 'label', kmeans.labels_.reshape(-1, 1))
    ...: BH.head(10)
Out[92]:
   indus     dis  medv  label
0   2.31  4.0900  24.0      2
1   7.07  4.9671  21.6      2
2   7.07  4.9671  34.7      4
3   2.18  6.0622  33.4      4
4   2.18  6.0622  36.2      4
5   2.18  6.0622  28.7      4
6   7.87  5.5605  22.9      2
7   7.87  5.9505  27.1      2
8   7.87  6.0821  16.5      2
9   7.87  6.5921  18.9      2
```

Q3. 각 클러스터의 중심점 값을 출력 하시오

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
BH = Boston[['indus', 'dis', 'medv']]
BH = BH.loc[0:499]
kmeans = KMeans(n_clusters=5, random_state=123).fit(BH)
BH.insert(3, 'label', kmeans.labels_.reshape(-1, 1))
BH.head(10)
for i in kmeans.cluster_centers_:
    print(i)
```

실행화면 캡쳐:



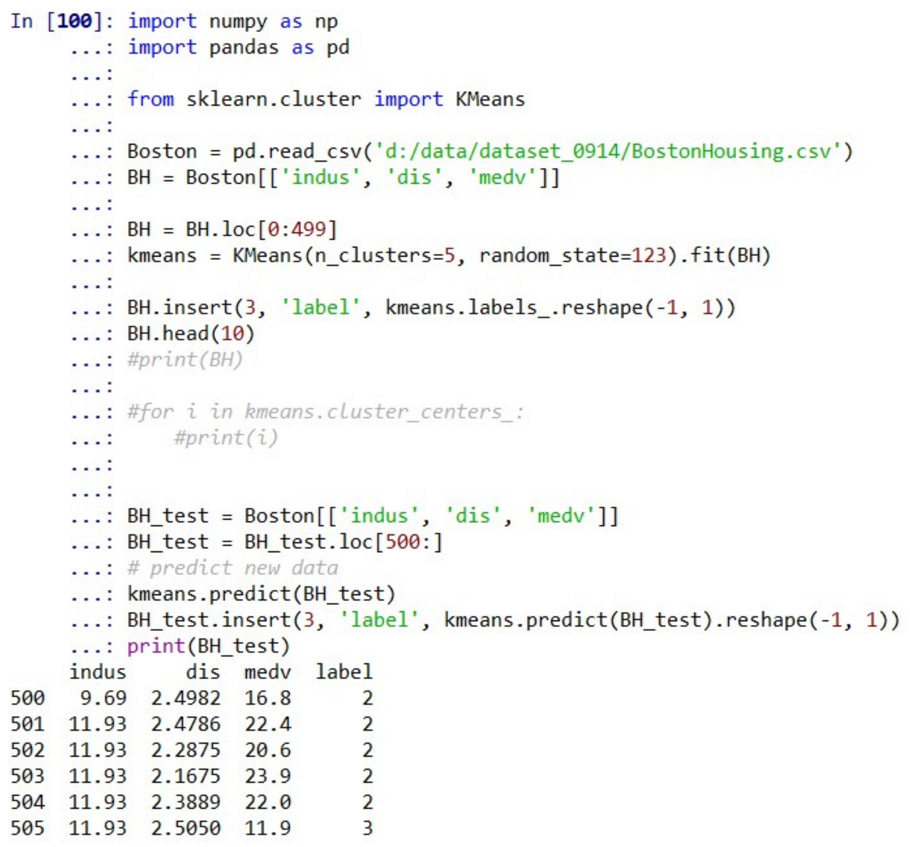
```
In [93]: import numpy as np
    ...: import pandas as pd
    ...:
    ...: from sklearn.cluster import KMeans
    ...:
    ...: Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
    ...: BH = Boston[['indus', 'dis', 'medv']]
    ...:
    ...: BH = BH.loc[0:499]
    ...: kmeans = KMeans(n_clusters=5, random_state=123).fit(BH)
    ...:
    ...: BH.insert(3, 'label', kmeans.labels_.reshape(-1, 1))
    ...: BH.head(10)
    ...: #print(BH)
    ...:
    ...: for i in kmeans.cluster_centers_:
    ...:     print(i)
    ...:
    ...:
[ 18.82280899   2.60442247  21.21235955]
[  8.70741935   3.36604516  47.4516129 ]
[  6.95240196   5.01984461  21.07892157]
[ 18.92009804   1.81338235  12.14313725]
[  3.68337838   4.87583243  32.24864865]
```

Q4. BH데이터에서 501행 이후에 대해 클러스터를 예측하여 보이시오 (데이터 + 클러스터 값을 함께 보임)

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
BH = Boston[['indus', 'dis', 'medv']]
BH = BH.loc[0:499]
kmeans = KMeans(n_clusters=5, random_state=123).fit(BH)
BH.insert(3, 'label', kmeans.labels_.reshape(-1, 1))
BH.head(10)
BH_test = Boston[['indus', 'dis', 'medv']]
BH_test = BH_test.loc[500:]
kmeans.predict(BH_test)
BH_test.insert(3, 'label', kmeans.predict(BH_test).reshape(-1, 1))
print(BH_test)
```

실행화면 캡쳐:



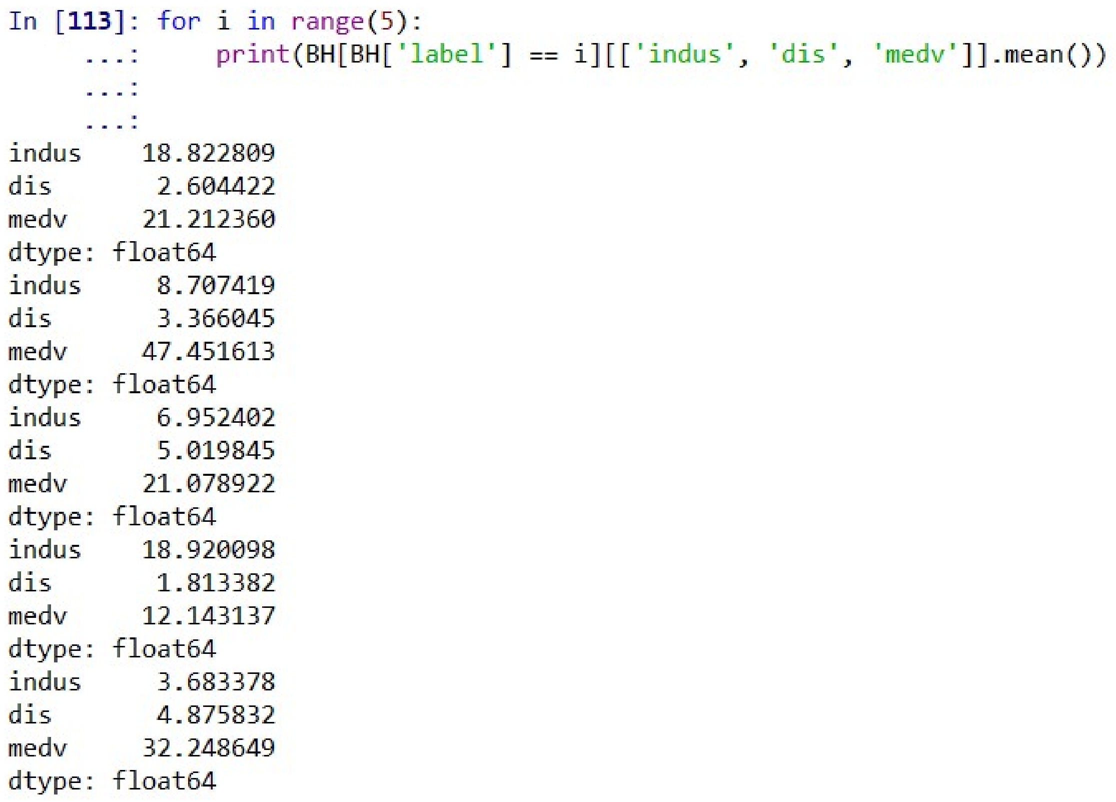
```
In [100]: import numpy as np
     ...: import pandas as pd
     ...:
     ...: from sklearn.cluster import KMeans
     ...:
     ...: Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
     ...: BH = Boston[['indus', 'dis', 'medv']]
     ...:
     ...: BH = BH.loc[0:499]
     ...: kmeans = KMeans(n_clusters=5, random_state=123).fit(BH)
     ...:
     ...: BH.insert(3, 'label', kmeans.labels_.reshape(-1, 1))
     ...: BH.head(10)
     ...: #print(BH)
     ...:
     ...: #for i in kmeans.cluster_centers_:
     ...:     #print(i)
     ...:
     ...:
     ...: BH_test = Boston[['indus', 'dis', 'medv']]
     ...: BH_test = BH_test.loc[500:]
     ...: # predict new data
     ...: kmeans.predict(BH_test)
     ...: BH_test.insert(3, 'label', kmeans.predict(BH_test).reshape(-1, 1))
     ...: print(BH_test)
     indus     dis  medv  label
500   9.69  2.4982  16.8      2
501  11.93  2.4786  22.4      2
502  11.93  2.2875  20.6      2
503  11.93  2.1675  23.9      2
504  11.93  2.3889  22.0      2
505  11.93  2.5050  11.9      3
```

Q5. (2점) 각 클러스터별로 ndus, dis, mdev 의 평균값을 구하되 scaling 이전의 값으로 계산하여 보이시오. (1~500행을 대상으로 계산한다)

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
for i in range(5):
   print(BH[BH['label'] == i][['indus', 'dis', 'medv']].mean())
```

실행화면 캡쳐:



```
In [113]: for i in range(5):
     ...:     print(BH[BH['label'] == i][['indus', 'dis', 'medv']].mean())
     ...:
     ...:
indus    18.822809
dis       2.604422
medv     21.212360
dtype: float64
indus     8.707419
dis       3.366045
medv     47.451613
dtype: float64
indus     6.952402
dis       5.019845
medv     21.078922
dtype: float64
indus    18.920098
dis       1.813382
medv     12.143137
dtype: float64
indus     3.683378
dis       4.875832
medv     32.248649
dtype: float64
```

PimaIndiansDiabetes dataset을 가지고 Classification 을 하고자 한다. (마지막의 diabetes 컬럼이 class label 임)

Q6. 데이터셋을 scaling 한 후 (diabetes 컬럼 제외) train/test set 으로 나누시오.
(test set을 30% 로 한다. random_state 는 123)
KNN 으로 분류 모델을 만드시오 (K=5)

Source code :

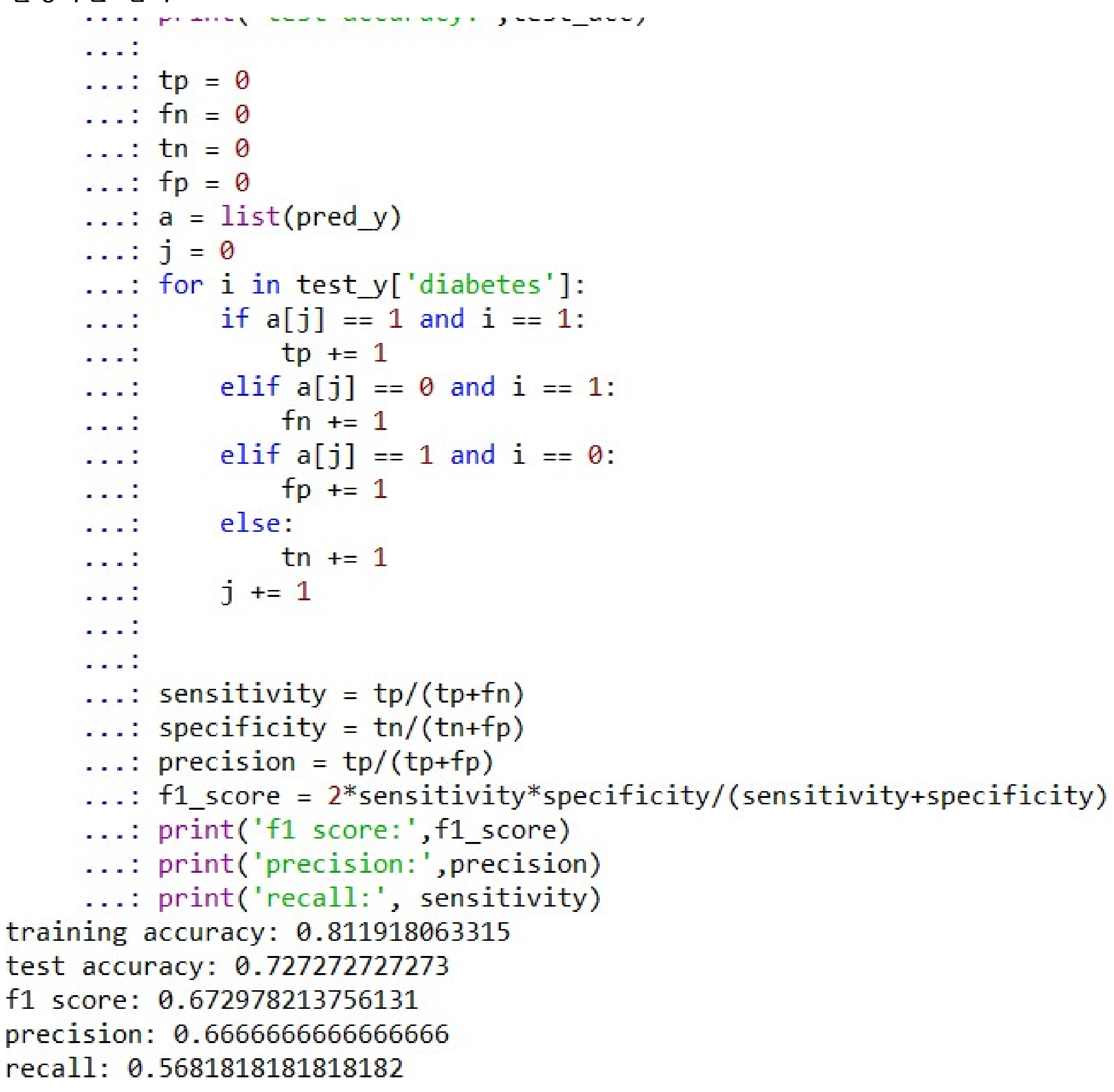
```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
import numpy as np
import pandas as pd

from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

pid = pd.read_csv('d:/data/dataset_0914/PimaIndiansDiabetes.csv')
pid_x = pid[['pregnant','glucose','pressure', 'triceps', 'insulin',
'mass', 'pedigree', 'age']]
pid_y = pid[['diabetes']]
pid_y.head(50)
pid_y.loc[pid_y['diabetes'] == 'pos'] = 1
pid_y.loc[pid_y['diabetes'] == 'neg'] = 0
scaler = StandardScaler()
scaler.fit(pid_x)
pid_x_scaled = scaler.transform(pid_x)
train_X, test_X, train_y, test_y  = train_test_split(pid_x, pid_y,
test_size=0.3, random_state=123)
train_y = train_y.astype('int')
test_y = test_y.astype('int')
model =  KNeighborsClassifier(n_neighbors=5)
model.fit(train_X, train_y)
```

Q7. 다음의 모델 성능 평가값을 보이시오
- training accuracy
- test accuracy
- f1 score (test set에 대해)
- precision (test set에 대해)
- recall   (test set에 대해)

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
pred_y = model.predict(train_X)
training_acc = accuracy_score(train_y, pred_y)
print('training accuracy:',training_acc)

pred_y = model.predict(test_X)
test_acc = accuracy_score(test_y, pred_y)
print('test accuracy:',test_acc)

tp = 0
fn = 0
tn = 0
fp = 0
a = list(pred_y)
j = 0
for i in test_y['diabetes']:
    if a[j] == 1 and i == 1:
        tp += 1
    elif a[j] == 0 and i == 1:
        fn += 1
    elif a[j] == 1 and i == 0:
        fp += 1
    else:
        tn += 1
    j += 1

sensitivity = tp/(tp+fn)
specificity = tn/(tn+fp)
precision = tp/(tp+fp)
f1_score = 2*sensitivity*specificity/(sensitivity+specificity)
print('f1 score:',f1_score)
print('precision:',precision)
print('recall:', sensitivity)
```

실행화면 캡처:



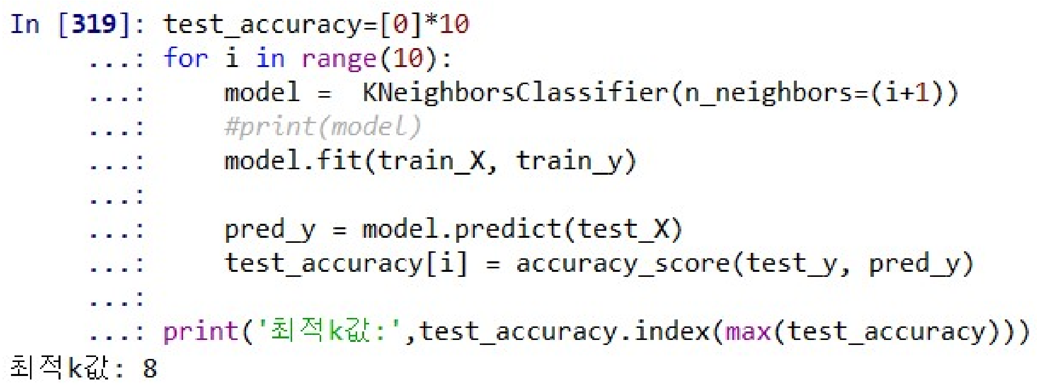
```
   ...: print('test accuracy:',test_acc)
   ...:
   ...: tp = 0
   ...: fn = 0
   ...: tn = 0
   ...: fp = 0
   ...: a = list(pred_y)
   ...: j = 0
   ...: for i in test_y['diabetes']:
   ...:     if a[j] == 1 and i == 1:
   ...:         tp += 1
   ...:     elif a[j] == 0 and i == 1:
   ...:         fn += 1
   ...:     elif a[j] == 1 and i == 0:
   ...:         fp += 1
   ...:     else:
   ...:         tn += 1
   ...:     j += 1
   ...:
   ...:
   ...: sensitivity = tp/(tp+fn)
   ...: specificity = tn/(tn+fp)
   ...: precision = tp/(tp+fp)
   ...: f1_score = 2*sensitivity*specificity/(sensitivity+specificity)
   ...: print('f1 score:',f1_score)
   ...: print('precision:',precision)
   ...: print('recall:', sensitivity)
training accuracy: 0.811918063315
test accuracy: 0.727272727273
f1 score: 0.672978213756131
precision: 0.6666666666666666
recall: 0.5681818181818182
```

Q8. (2점) K 값을 1~10 으로 바꾸어 가면서 테스트하여 가장 높은 test accuracy 값을 도출하는 K값을 찾으시오

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
test_accuracy=[0]*10
for i in range(10):
    model =  KNeighborsClassifier(n_neighbors=(i+1))
    model.fit(train_X, train_y)

    pred_y = model.predict(test_X)
    test_accuracy[i] = accuracy_score(test_y, pred_y)
print('최적k값:',test_accuracy.index(max(test_accuracy)))
```

실행화면 캡쳐:



```
In [319]: test_accuracy=[0]*10
     ...: for i in range(10):
     ...:     model =  KNeighborsClassifier(n_neighbors=(i+1))
     ...:     #print(model)
     ...:     model.fit(train_X, train_y)
     ...: 
     ...:     pred_y = model.predict(test_X)
     ...:     test_accuracy[i] = accuracy_score(test_y, pred_y)
     ...: 
     ...: print('최적k값:',test_accuracy.index(max(test_accuracy)))
최적k값: 8
```

Q9. **PimaIndiansDiabetes** 데이터셋에 대해 KNN (K=5) 으로 분류모델을 만들되 10-fold cross validation 으로 성능을 평가하시오

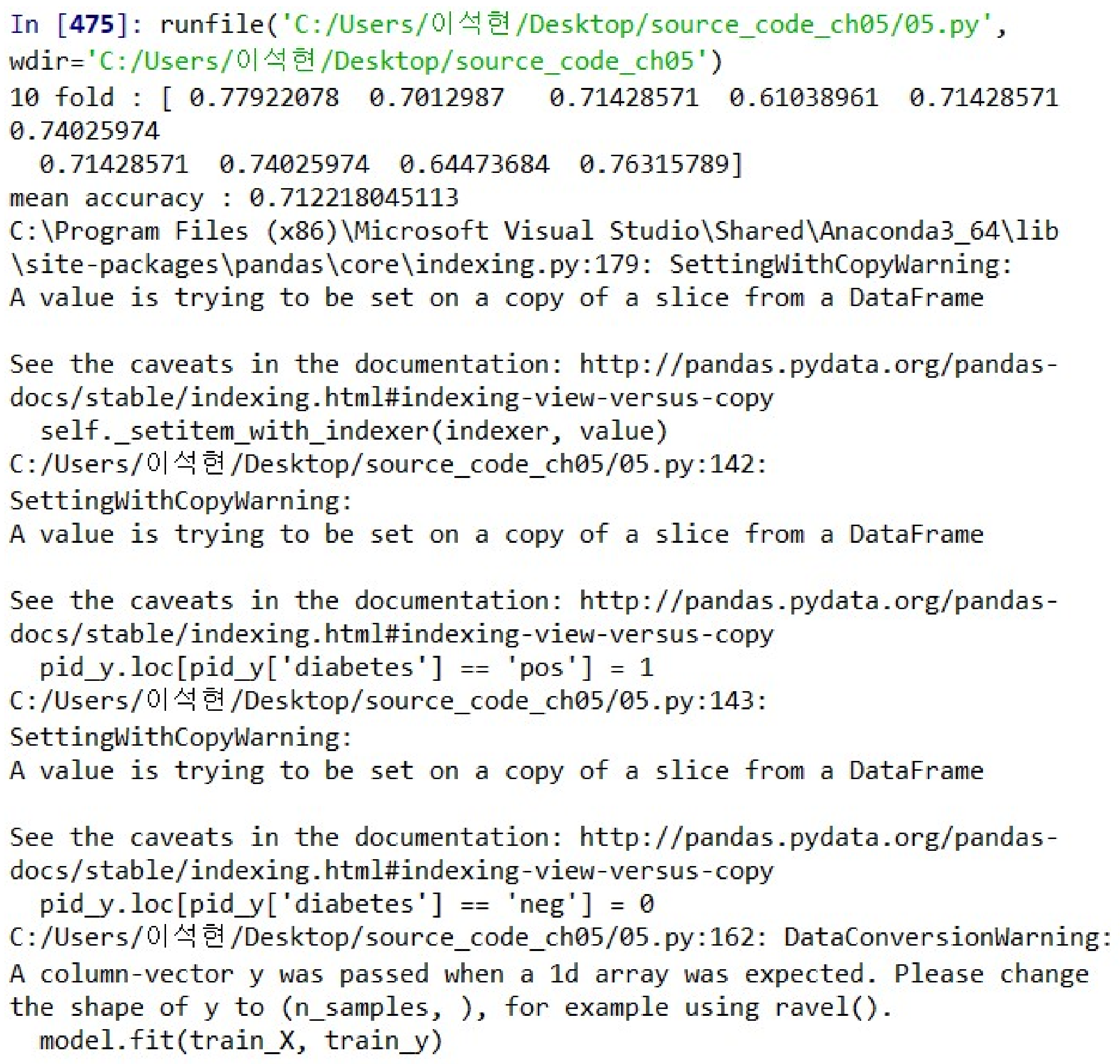
* random_state 는 123

- 각 fold 별 accuracy를 보이시오
- 전체 평균 accuracy를 보이시오

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd

pid = pd.read_csv('d:/data/dataset_0914/PimaIndiansDiabetes.csv')
pid_x = pid[['pregnant','glucose','pressure', 'triceps', 'insulin',
'mass', 'pedigree', 'age']]
pid_y = pid[['diabetes']]
pid_y.loc[pid_y['diabetes'] == 'pos'] = 1
pid_y.loc[pid_y['diabetes'] == 'neg'] = 0
scaler = StandardScaler()
scaler.fit(pid_x)
pid_x_scaled = scaler.transform(pid_x)

kf = KFold(n_splits=10, random_state=123, shuffle=True)
model =  KNeighborsClassifier(n_neighbors=5)
acc = np.zeros((10,))
i = 0
for train_index, test_index in kf.split(pid_x_scaled):
   train_X,    test_X    =    pid_x.iloc[train_index].astype('int'),
pid_x.iloc[test_index].astype('int')
   train_y,   test_y   =     pid_y.iloc[train_index].astype('int'),
pid_y.iloc[test_index].astype('int')
   model.fit(train_X, train_y)
   pred_y = model.predict(test_X)
   acc[i] = accuracy_score(test_y, pred_y)
   i += 1
print("10 fold :", acc)
print("mean accuracy :", np.mean(acc))
```

실행화면 캡쳐:

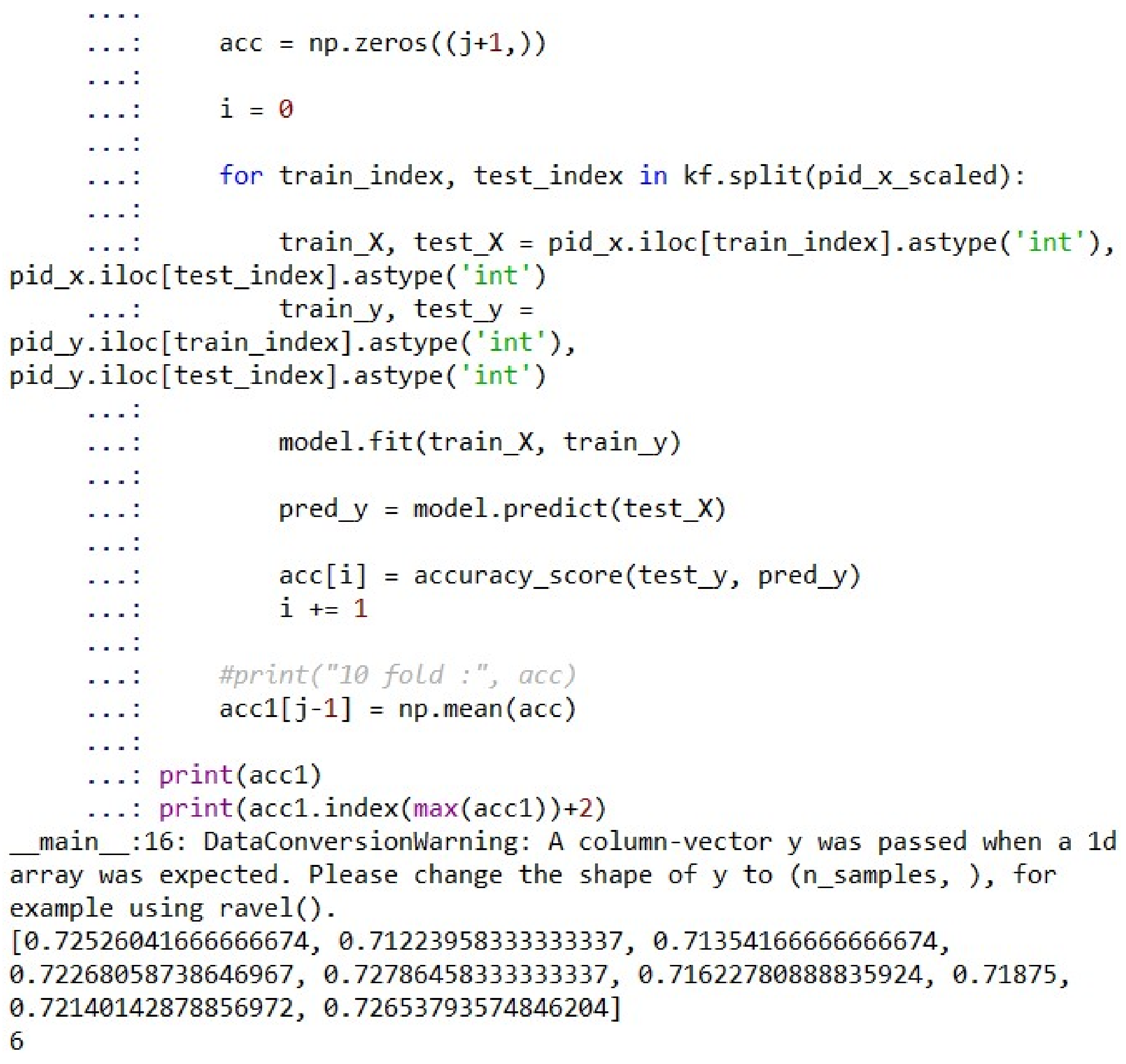


```
In [475]: runfile('C:/Users/이석현/Desktop/source_code_ch05/05.py',
wdir='C:/Users/이석현/Desktop/source_code_ch05')
10 fold : [ 0.77922078  0.7012987   0.71428571  0.61038961  0.71428571
0.74025974
  0.71428571  0.74025974  0.64473684  0.76315789]
mean accuracy : 0.712218045113
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib
\site-packages\pandas\core\indexing.py:179: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
C:/Users/이석현/Desktop/source_code_ch05/05.py:142:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  pid_y.loc[pid_y['diabetes'] == 'pos'] = 1
C:/Users/이석현/Desktop/source_code_ch05/05.py:143:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  pid_y.loc[pid_y['diabetes'] == 'neg'] = 0
C:/Users/이석현/Desktop/source_code_ch05/05.py:162: 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().
  model.fit(train_X, train_y)
```

Q10. (3점) K 값을 1~10 으로 바꾸어 가면서 테스트하여 가장 높은 test accuracy 값을 도출하는 K값을 찾으시오. 단 10-fold cross validation 으로 각 K 의 accuracy를 평가한다.
* random_state 는 123

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd

pid = pd.read_csv('d:/data/dataset_0914/PimaIndiansDiabetes.csv')
pid_x = pid[['pregnant','glucose','pressure', 'triceps', 'insulin',
'mass', 'pedigree', 'age']]
pid_y = pid[['diabetes']]
pid_y.loc[pid_y['diabetes'] == 'pos'] = 1
pid_y.loc[pid_y['diabetes'] == 'neg'] = 0
scaler = StandardScaler()
scaler.fit(pid_x)
pid_x_scaled = scaler.transform(pid_x)
acc1=[0]*9
for j in range(1,10):
   kf = KFold(n_splits=j+1, random_state=123)
   model =  KNeighborsClassifier(n_neighbors=5)
   acc = np.zeros((j+1,))
   i = 0
   for train_index, test_index in kf.split(pid_x_scaled):
      train_X,   test_X   =   pid_x.iloc[train_index].astype('int'),
pid_x.iloc[test_index].astype('int')
      train_y,  test_y  =    pid_y.iloc[train_index].astype('int'),
pid_y.iloc[test_index].astype('int')
      model.fit(train_X, train_y)
      pred_y = model.predict(test_X)
      acc[i] = accuracy_score(test_y, pred_y)
      i += 1
   #print("10 fold :", acc)
   acc1[j-1] = np.mean(acc)
print(acc1)
print(acc1.index(max(acc1))+2)
```

실행화면 캡처:



```
     ...:
     ...:     acc = np.zeros((j+1,))
     ...:
     ...:     i = 0
     ...:
     ...:     for train_index, test_index in kf.split(pid_x_scaled):
     ...:
     ...:         train_X, test_X = pid_x.iloc[train_index].astype('int'),
pid_x.iloc[test_index].astype('int')
     ...:         train_y, test_y =
pid_y.iloc[train_index].astype('int'),
pid_y.iloc[test_index].astype('int')
     ...:
     ...:         model.fit(train_X, train_y)
     ...:
     ...:         pred_y = model.predict(test_X)
     ...:
     ...:         acc[i] = accuracy_score(test_y, pred_y)
     ...:         i += 1
     ...:
     ...:     #print("10 fold :", acc)
     ...:     acc1[j-1] = np.mean(acc)
     ...:
     ...: print(acc1)
     ...: print(acc1.index(max(acc1))+2)
__main__:16: 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().
[0.7252604166666674, 0.7122395833333337, 0.7135416666666674,
0.72268058738646967, 0.7278645833333337, 0.7162278088835924, 0.71875,
0.72140142878856972, 0.72653793574846204]
6
```

Q11. K-fold cross validation을 사용하는 이유를 설명하시오

training 데이터셋과 test 데이터셋이 무작위로 나뉘어지므로 test accuracy의 값이 경우에 따라 차이가 크게 날 수 있다. 따라서 K-fold cross validation을 사용하여 데이터셋을 k등분한 후 각 fold의 정확도들의 평균을 모델의 정확도로 계산해준다.

주어진 데이터셋으로 모델을 개발할 때 미래의 정확도를 추정하기위해 사용한다. 또한 최종모델 개발 시 hyper parameter 튜닝에 사용되며 전처리 할대 feature selection에 사용된다.