4주. Clustering, KNN			
학번	32152339	이름	송준영

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 numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import pandas as pd
#데이터 불러오기
df=pd.read_csv('C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_09
14/BostonHousing.csv')
df=df[['indus', 'dis', 'medv']]
#scaling
scaler=StandardScaler()
scaler.fit(df)
BH=scaler.transform(df)


BH[:5,]
```

실행화면 캡쳐:

```
In [187]: import numpy as np
     ...: import pandas as pd
     ...: from sklearn.preprocessing import StandardScaler
     ...: from sklearn.cluster import KMeans
     ...: from sklearn.preprocessing import StandardScaler
     ...: import pandas as pd
     ...: #데이터 불러오기
     ...: df=pd.read_csv('C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_0914/BostonHousing.csv')
     ...: df=df[['indus', 'dis', 'medv']]
     ...: #scaling
     ...: scaler=StandardScaler()
     ...: scaler.fit(df)
     ...: BH=scaler.transform(df)
     ...:
     ...: BH[:5,]
Out[187]:
array([[-1.2879095 ,  0.1402136 ,  0.15968566],
       [-0.59338101,  0.55715988, -0.10152429],
       [-0.59338101,  0.55715988,  1.32424667],
       [-1.30687771,  1.07773662,  1.18275795],
       [-1.30687771,  1.07773662,  1.48750288]])
```

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
kmeans = KMeans(n_clusters=5, random_state=123).fit(BH[:500,])
np.hstack((BH[:500,], kmeans.labels_.reshape(-1, 1)))[:10,]
```

실행화면 캡쳐:

```
In [180]: kmeans = KMeans(n_clusters=5, random_state=123).fit(BH[:500,])
     ...: np.hstack((BH[:500,], kmeans.labels_.reshape(-1, 1)))[:10,]
Out[180]:
array([[-1.2879095 ,  0.1402136 ,  0.15968566,  2.        ],
       [-0.59338101,  0.55715988, -0.10152429,  2.        ],
       [-0.59338101,  0.55715988,  1.32424667,  1.        ],
       [-1.30687771,  1.07773662,  1.18275795,  1.        ],
       [-1.30687771,  1.07773662,  1.48750288,  1.        ],
       [-1.30687771,  1.07773662,  0.6712218 ,  4.        ],
       [-0.47665354,  0.83924392,  0.03996443,  2.        ],
       [-0.47665354,  1.02463789,  0.49708184,  4.        ],
       [-0.47665354,  1.08719646, -0.65659542,  2.        ],
       [-0.47665354,  1.32963473, -0.39538548,  4.        ]])
```

Q3. 각 클러스터의 중심점 값을 출력 하시오

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
kmeans.cluster_centers_
```

실행화면 캡쳐:

```
In [181]: kmeans.cluster_centers_
Out[181]:
array([[ 1.17486908, -0.83795852, -0.7256677 ],
       [-1.04746396,  0.03299445,  1.61437165],
       [-0.40576204,  0.08002824, -0.12606471],
       [ 1.12397199, -1.0298077 ,  2.89477147],
       [-1.01177187,  1.71875715,  0.16656619]])
```

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

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
np.hstack((BH[500:,], kmeans.predict(BH[500:,]).reshape(-1, 1)))
```

실행화면 캡쳐:

```
In [185]: np.hstack((BH[500:,], kmeans.predict(BH[500:,]).reshape(-1, 1)))
Out[185]:
array([[-0.21109853, -0.61647899, -0.62394418,  2.        ],
       [ 0.11573841, -0.62579623, -0.01445431,  2.        ],
       [ 0.11573841, -0.71663927, -0.21036176,  2.        ],
       [ 0.11573841, -0.77368357,  0.14880191,  2.        ],
       [ 0.11573841, -0.66843684, -0.0579893 ,  2.        ],
       [ 0.11573841, -0.61324648, -1.15724782,  0.        ]])
```

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
#데이터셋 구성
df=pd.read_csv('C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_0914/BostonHousing.csv')
pd.set_option('display.max_column', 500)
df.describe
df=df[['indus', 'dis', 'medv']][:500]
#scaling 데이터셋 생성
scaler=StandardScaler()
scaler.fit(df)
df_scaled=scaler.transform(df)
#kmeans 학습 및 예측
kmeans = KMeans(n_clusters=5, random_state=123).fit(df_scaled)
#원 데이터에 예측값 병합
df['cluster']=kmeans.labels_.reshape(-1, 1)
#클러스터별 평균
df.groupby('cluster').mean()
```

실행화면 캡쳐:

```
In [196]: df=pd.read_csv('C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_0914/BostonHousing.csv')
     ...: pd.set_option('display.max_column', 500)
     ...: df.describe
     ...: df=df[['indus', 'dis', 'medv']][:500]
     ...: #scaling 데이터셋 생성
     ...: scaler=StandardScaler()
     ...: scaler.fit(df)
     ...: df_scaled=scaler.transform(df)
     ...: #kmeans 학습 및 예측
     ...: kmeans = KMeans(n_clusters=5, random_state=123).fit(df_scaled)
     ...: #원 데이터에 예측값 병합
     ...: df['cluster']=kmeans.labels_.reshape(-1, 1)
     ...: #클러스터별 평균
     ...: df.groupby('cluster').mean()
Out[196]:
             indus       dis       medv
cluster
0        19.188827  2.032289  15.865363
1         3.957910  3.864451  37.365672
2         8.355860  3.963392  21.374522
3        18.840000  1.628710  49.130000
4         4.202529  7.410669  24.063218
```

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
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
#데이터셋 구성
df=pd.read_csv('C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_09
14/PimaIndiansDiabetes.csv')
X=df.drop('diabetes',axis=1)
y=df['diabetes']
scaler=StandardScaler()
scaler.fit(X)
X=scaler.transform(X)
#데이터셋 분할
train_X,  test_X,  train_y,  test_y  =train_test_split(X,  y,
test_size=0.3,random_state=123)
#KNN 모델 생성
model =  KNeighborsClassifier(n_neighbors=5)
model.fit(train_X, train_y)
```

```
In [198]: from sklearn.neighbors import KNeighborsClassifier
     ...: from sklearn.model_selection import train_test_split
     ...: from sklearn.metrics import accuracy_score
     ...: #데이터셋 구성
     ...: df=pd.read_csv('C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_0914/PimaIndiansDiabetes.csv')
     ...: X=df.drop('diabetes',axis=1)
     ...: y=df['diabetes']
     ...: #SCALING
     ...: scaler=StandardScaler()
     ...: scaler.fit(X)
     ...: X=scaler.transform(X)
     ...: #데이터셋 분할
     ...: train_X, test_X, train_y, test_y =train_test_split(X, y, test_size=0.3,random_state=123)
     ...: #KNN 모델 생성
     ...: model =  KNeighborsClassifier(n_neighbors=5)
     ...: # 모델 훈련
     ...: model.fit(train_X, train_y)
Out[198]: KNeighborsClassifier()
```

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
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
#predict
pred_y_train=model.predict(train_X)
pred_y_test=model.predict(test_X)
# training accuracy
accuracy_score(train_y, pred_y_train)
# test accuracy
accuracy_score(test_y, pred_y_test)
# f1 score  (test set에 대해)
f1_score(test_y, pred_y_test,pos_label ='pos')
# precision (test set에 대해)
precision_score(test_y, pred_y_test,pos_label ='pos')
# recall  (test set에 대해)
recall_score(test_y, pred_y_test,pos_label ='pos')
```

실행화면 캡쳐:

```
In [236]: from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
     ...: #predict
     ...: pred_y_train=model.predict(train_X)
     ...: pred_y_test=model.predict(test_X)

In [237]: accuracy_score(train_y, pred_y_train)
Out[237]: 0.8156424581005587

In [238]: accuracy_score(test_y, pred_y_test)
Out[238]: 0.7402597402597403

In [239]: f1_score(test_y, pred_y_test,pos_label ='pos')
Out[239]: 0.625

In [240]: precision_score(test_y, pred_y_test,pos_label ='pos')
Out[240]: 0.6944444444444444

In [241]: recall_score(test_y, pred_y_test,pos_label ='pos')
Out[241]: 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
K=[]
TestAccuracy=[]
for i in range(1,11):
    model =  KNeighborsClassifier(n_neighbors=i)
    model.fit(train_X,train_y)
    pred_y = model.predict(test_X)
    K.append(i)
    TestAccuracy.append(accuracy_score(test_y,pred_y))
K_Acc=pd.DataFrame({'TestAccuracy':TestAccuracy,'K':K})

#가장 높은 test accuracy 값을 도출하는 k값
K_Acc[K_Acc['TestAccuracy'].max()==K_Acc['TestAccuracy']]
```

실행화면 캡쳐:

```
In [242]: K=[]
     ...: TestAccuracy=[]
     ...: for i in range(1,11):
     ...:     model =  KNeighborsClassifier(n_neighbors=i)
     ...:     model.fit(train_X,train_y)
     ...:     pred_y = model.predict(test_X)
     ...:     K.append(i)
     ...:     TestAccuracy.append(accuracy_score(test_y,pred_y))
     ...:
     ...: K_Acc=pd.DataFrame({'TestAccuracy':TestAccuracy,'K':K})
     ...:
     ...: #가장 높은 test accuracy 값을 도출하는 k값
     ...: K_Acc[K_Acc['TestAccuracy'].max()==K_Acc['TestAccuracy']]
Out[242]:
   TestAccuracy  K
2      0.761905  3
```

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.model_selection import KFold
df=pd.read_csv('C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_09
14/PimaIndiansDiabetes.csv')
X=df.drop('diabetes',axis=1)
y=df['diabetes']
#SCALING
scaler=StandardScaler()
scaler.fit(X)
X=scaler.transform(X)
#10fold
n_splits=10
kf = KFold(n_splits=n_splits, random_state=123, shuffle=True)
# 모델 생성
model =  KNeighborsClassifier(n_neighbors=5)
acc,i = np.zeros(n_splits),0
for train_index, test_index in kf.split(X):
    print("fold:", i)
    train_X, test_X = X[train_index], X[test_index]
    train_y, test_y =  y[train_index], y[test_index]
    model.fit(train_X, train_y)
    pred_y = model.predict(test_X)
    acc[i] = accuracy_score(test_y, pred_y)
    print('Accuracy : {0:3f}'.format(acc[i]))
    i += 1
print(f'{n_splits}fold accuracy:', acc)
print("mean accuracy :", np.mean(acc))
```

실행화면 캡쳐:

```
In [245]: from sklearn.model_selection import KFold
     ...: df=pd.read_csv('C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_0914/PimaIndiansDiabetes.csv')
     ...: X=df.drop('diabetes',axis=1)
     ...: y=df['diabetes']
     ...: #SCALING
     ...: scaler=StandardScaler()
     ...: scaler.fit(X)
     ...: X=scaler.transform(X)
     ...: #10fold
     ...: n_splits=10
     ...: kf = KFold(n_splits=n_splits, random_state=123, shuffle=True)
     ...: # 모델 생성
     ...: model =  KNeighborsClassifier(n_neighbors=5)
     ...: acc,i = np.zeros(n_splits),0
     ...: for train_index, test_index in kf.split(X):
     ...:     print("fold:", i)
     ...:     train_X, test_X = X[train_index], X[test_index]
     ...:     train_y, test_y =  y[train_index], y[test_index]
     ...:     model.fit(train_X, train_y)
     ...:     pred_y = model.predict(test_X)
     ...:     acc[i] = accuracy_score(test_y, pred_y)
     ...:     print('Accuracy : {0:3f}'.format(acc[i]))
     ...:     i += 1
     ...:
     ...: print(f'{n_splits}fold accuracy:', acc)
     ...: print("mean accuracy :", np.mean(acc))
fold: 0
Accuracy : 0.779221
fold: 1
Accuracy : 0.779221
fold: 2
Accuracy : 0.766234
fold: 3
Accuracy : 0.675325
fold: 4
Accuracy : 0.662338
fold: 5
Accuracy : 0.727273
fold: 6
Accuracy : 0.714286
fold: 7
Accuracy : 0.792208
fold: 8
Accuracy : 0.697368
fold: 9
Accuracy : 0.776316
10fold accuracy: [0.77922078 0.77922078 0.76623377 0.67532468 0.66233766 0.72727273
 0.71428571 0.79220779 0.69736842 0.77631579]
mean accuracy : 0.7369788106630212
```

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

```
df=pd.read_csv('C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_09
14/PimaIndiansDiabetes.csv')
X=df.drop('diabetes',axis=1)
y=df['diabetes']
#SCALING
scaler=StandardScaler()
scaler.fit(X)
X=scaler.transform(X)
n_splits=10
kf = KFold(n_splits=n_splits, random_state=123, shuffle=True)

k_acc = np.zeros((n_splits,2))
acc = np.zeros(n_splits)
i = 1
for k in range(1,11):
    model =  KNeighborsClassifier(n_neighbors=k)
    print('---'*20)
    print("K:",k)
    for i,(train_index, test_index) in enumerate(kf.split(X),1):

        train_X, test_X = X[train_index], X[test_index]
        train_y, test_y =  y[train_index], y[test_index]

        model.fit(train_X, train_y)

        pred_y = model.predict(test_X)

        acc[i-1] = accuracy_score(test_y, pred_y)

    print("mean accuracy :", np.mean(acc))

    k_acc[k-1,0]=k
    k_acc[k-1,1]=np.mean(acc)
print('---'*20)
#test accuracy가 가장 높은 k 출력 --> 10
print(f'best K       : {k_acc[k_acc[:,1]==k_acc[:,1].max()][:,0]}')
print(f'test Accuracy : {k_acc[k_acc[:,1]==k_acc[:,1].max()][:,1]}')
```

실행화면 캡쳐:

```
In [248]: df=pd.read_csv('C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_0914/PimaIndiansDiabetes.csv')
     ...: X=df.drop('diabetes',axis=1)
     ...: y=df['diabetes']
     ...: #SCALING
     ...: scaler=StandardScaler()
     ...: scaler.fit(X)
     ...: X=scaler.transform(X)
     ...: n_splits=10
     ...: kf = KFold(n_splits=n_splits, random_state=123, shuffle=True)
     ...:
     ...: k_acc = np.zeros((n_splits,2))
     ...: acc = np.zeros(n_splits)
     ...: i = 1
     ...: for k in range(1,11):
     ...:     model =  KNeighborsClassifier(n_neighbors=k)
     ...:     print('---'*20)
     ...:     print("K:",k)
     ...:     for i,(train_index, test_index) in enumerate(kf.split(X),1):
     ...:
     ...:         train_X, test_X = X[train_index], X[test_index]
     ...:         train_y, test_y =  y[train_index], y[test_index]
     ...:
     ...:         model.fit(train_X, train_y)
     ...:
     ...:         pred_y = model.predict(test_X)
     ...:
     ...:         acc[i-1] = accuracy_score(test_y, pred_y)
     ...:
     ...:     print("mean accuracy :", np.mean(acc))
     ...:
     ...:     k_acc[k-1,0]=k
     ...:     k_acc[k-1,1]=np.mean(acc)
     ...:
     ...: print('---'*20)
     ...: #test accuracy가 가장 높은 k 출력 --> 10
     ...: print(f'best K        : {k_acc[k_acc[:,1]==k_acc[:,1].max()][:,0]}')
     ...: print(f'test Accuracy : {k_acc[k_acc[:,1]==k_acc[:,1].max()][:,1]}')
------------------------------------------------------------
K: 1
mean accuracy : 0.721308954203691
------------------------------------------------------------
K: 2
mean accuracy : 0.7121667805878331
------------------------------------------------------------
K: 3
mean accuracy : 0.7486671223513329
------------------------------------------------------------
K: 4
mean accuracy : 0.7395762132604238
------------------------------------------------------------
K: 5
mean accuracy : 0.7369788106630212
------------------------------------------------------------
K: 6
mean accuracy : 0.7292036910457963
------------------------------------------------------------
K: 7
mean accuracy : 0.7461893369788106
------------------------------------------------------------
K: 8
mean accuracy : 0.740926179084074
------------------------------------------------------------
K: 9
mean accuracy : 0.7461722488038277
------------------------------------------------------------
K: 10
mean accuracy : 0.7487525632262473
------------------------------------------------------------
best K        : [10.]
test Accuracy : [0.74875256]
```

Q11. K-fold cross validation을 사용하는 이유를 설명하시오

단 한 번의 샘플링으로 train dataset과 valid dataset을 나눈다면(hold-out 검증) 그 결과는 신뢰받지 못한다. 샘플링을 할 때마다 결과가 조금씩 다르게 나오기 때문이다. 또한 valid dataset 성능을 향상시키는 작업을 반복하다 보면 어느새 valid dataset에 과적합 되는 경우가 생길 수도 있기 때문이다.

그렇기 때문에 우리는 보다 정확하고 신뢰할 수 있을만한 평가방법으로 분석을 해야 하는데 이에 대한 대안이 바로 K-fold cross validation이다. K-fold cross validation은 최초의 데이터셋을 k겹 분할한 후 각 겹마다의 valid 성능을 구하여 평균내기 때문에 보다 더 신뢰할 수 있고 편차가 적은 성능 척도를 관찰할 수 있다.