4주. Regression			
학번	32152339	이름	송준영

※ 이번 실습에 사용된 데이터셋은 공지에 있는 데이터셋 압축파일에 포함되어 있음

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

BostonHousing dataset을 가지고 단순 선형 회귀 분석을 하고자 한다.

Q1 lstat (소득분위가 하위인 사람들의 비율) 로 mdev (주택가격)을 예측하는 단순 선형회귀 모델을 만드시오 (tain, test 나누지 않음). 모델의 내용을 보이시오

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
#필요 라이브러리, 함수 불러오기
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split
#데이터 불러오기
df=pd.read_csv("C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_09
14/BostonHousing.csv")
#데이터셋 가공
X=np.array(df['lstat']).reshape(-1,1)
y=np.array(df['medv']).reshape(-1,1)
#모델 학습
model=LinearRegression()
model.fit(X,y)
print(model)
p r i n t ( ' c o e f f i c i e n t s : { 0 : . 2 f } , I n t e r c e p t
{1:.3f}'.format(model.coef_[0][0],model.intercept_[0]))
```

실행화면 캡쳐:

```
   ....
   ...: #데이터 불러오기
   ...: df=pd.read_csv("C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_0914/BostonHousing.csv")
   ...: #데이터셋 가공
   ...: X=np.array(df['lstat']).reshape(-1,1)
   ...: y=np.array(df['medv']).reshape(-1,1)
   ...: #모델 학습
   ...: model=LinearRegression()
   ...: model.fit(X,y)
   ...: pred_y=model.predict(X)
   ...: print(model)
   ...:
   ...: print('coefficients:{0:.2f},Intercept {1:.3f}'.format(model.coef_[0][0],model.intercept_[0]))
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
coefficients:-0.95,Intercept 34.554
```

Q2. 모델에서 만들어진 회귀식을 쓰시오 (mdev = W x lstat + b 의 형태)

mdev = (-0.95) * lstat + 34.554

Q3. 회귀식을 이용하여 lstat 의 값이 각각 2.0, 3.0, 4.0, 5.0 일 때 mdev 의 값을 예측하여 제시하시오.

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
print(model.predict([[2]]))
print(model.predict([[3]]))
print(model.predict([[4]]))
print(model.predict([[5]]))
```

실행화면 캡쳐:

```
In [18]: print(model.predict([[2]]))
    ...: print(model.predict([[3]]))
    ...: print(model.predict([[4]]))
    ...: print(model.predict([[5]]))
[[32.65374217]]
[[31.70369282]]
[[30.75364346]]
[[29.80359411]]
```

Q4. 데이터셋의 모든 lstat 값을 회귀식에 넣어 mdev 의 값을 예측 한 뒤 mean square error를 계산하여 제시하시오

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
pred_y=model.predict(X)
print('Mean squared error: {0:.2f}'.format(mean_squared_error(y,pred_y)))
```

실행화면 캡쳐:

```
In [20]: pred_y=model.predict(X)
    ...: print('Mean squared error: {0:.2f}'.format(mean_squared_error(y,pred_y)))
Mean squared error: 38.48
```

BostonHousing dataset을 가지고 다중 선형 회귀 분석을 하고자 한다.

Q5. **lstat** (소득분위가 하위인 사람들의 비율), **ptratio**(초등교사비율), **tax**(세금), **rad**(고속도로접근성)로 **mdev** (주택가격)을 예측하는 단순 선형회귀 모델을 만드시오 (tain, test 나누지 않음). 모델의 내용을 보이시오

Source code :

```
df_X=df[['lstat','ptratio','tax','rad']]
df_y=df['medv']

model=LinearRegression()
model.fit(df_X,df_y)

print('coefficients:{0:.2f},{1:.2f},{2:.2f},{3:.2f}        Intercept
{4:.3f}'.

format(model.coef_[0],model.coef_[1],model.coef_[2],model.coef_[3],
model.intercept_))
```

실행화면 캡쳐:

```
In [26]: df_X=df[['lstat','ptratio','tax','rad']]
    ...: df_y=df['medv']
    ...:
    ...: mode=LinearRegression()
    ...: model.fit(df_X,df_y)
    ...:
    ...: print('coefficients:{0:.2f},{1:.2f},{2:.2f},{3:.2f} Intercept {4:.3f}'.
    ...:
format(model.coef_[0],model.coef_[1],model.coef_[2],model.coef_[3],model.intercept_))
coefficients:-0.81,-1.23,-0.02,0.33 Intercept 58.546
```

Q6. 모델에서 만들어진 회귀식을 쓰시오

mdev = (-0.81)*lstat + (-1.23)*ptratio + (-0.02)*tax + (-0.33)*rad + 58.546

Q7. lstat, ptratio, tax, rad 의 값이 다음과 같을 때 mdev 의 예측값을 보이시오.

lstat	ptratio	tax	rad
2.0	14	296	1
3.0	15	222	2
4.0	15	250	3

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
test_x=np.array([[2.0,14,296,1],
                 [3.0,15,222,2],
                 [4.0,15,250,3]])
model.predict(test_x)
```

실행화면 캡쳐:

```
In [33]: test_x=np.array([[2.0,14,296,1],
    ...:                  [3.0,15,222,2],
    ...:                  [4.0,15,250,3]])
    ...: model.predict(test_x)
Out[33]: array([35.5479738 , 34.95427204, 34.04856204])
```

Q8. 데이터셋의 모든 lstat, ptratio, tax, rad 값을 회귀식에 넣어 mdev 의 값을 예측한 뒤 mean square error를 계산하여 제시하시오.

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
pred_y=model.predict(df[['lstat','ptratio','tax','rad']])
print('Mean squared error: {0:.2f}'.format(mean_squared_error(df['medv'],pred_y)))
```

실행화면 캡쳐:

```
In [8]: pred_y=model.predict(df[['lstat','ptratio','tax','rad']])
   ...: print('Mean squared error: {0:.2f}'.format(mean_squared_error(df['medv'],pred_y)))
Mean squared error: 31.80
```

Q9. lstat 하나만 가지고 모델을 만든 경우와 4개 변수를 가지고 모델을 만든 경우 어느쪽이 더 좋은 모델이라고 할수 있는가? 그 이유는?

4개 변수를 가지고 모델을 만든 경우이다.

- 1개 변수를 가지고 만든 모델의 MSE : 38.48
- 4개 변수를 가지고 만든 모델의 MSE : 31.8

4개 변수를 가지고 만든 모델의 MSE가 더 낮다. 이는 성능이 더 좋다는 것을 의미하므로 더 좋은 모델이라고 할 수 있다.

ucla_admit.csv 파일은 미국 UCLA 의 대학원 입학에 대한 정보를 담고 있다. 컬럼(변수)에 대한 설명은 다음과 같다.

admit : 합격여부 (1:합격, 0:불합격)

gre : GRE 점수

gpa : GPA 점수

rank : 성적 석차

이 데이터셋에 대해 다음의 문제를 해결하시오

Q10. **gre**, **gpa**, **rank**를 가지고 합격여부를 예측하는 logistic regression 모델을 만드시오. (train, test를 나누되 test 의 비율은 30% 로 하고 random_state 는 1234 로 한다)

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
#데이터 불러오기
df=pd.read_csv("C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_0914/ucla_admit.csv ")
df_X=df[[ 'gre',  'gpa', 'rank']]
df_y=df.admit
#데이터 분할
train_X,test_X,train_y,test_y=train_test_split(df_X,df_y,test_size=0.3,random_state=1234)

#학습
model=LogisticRegression()
model.fit(train_X,train_y)
```

실행화면 캡쳐:

```
In [12]: df=pd.read_csv("C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_0914/ucla_admit.csv ")
    ...: df_X=df[[ 'gre',  'gpa', 'rank']]
    ...: df_y=df.admit
    ...: #데이터 분할
    ...:
train_X,test_X,train_y,test_y=train_test_split(df_X,df_y,test_size=0.3,random_state=1234)
    ...:
    ...: #학습
    ...: model=LogisticRegression()
    ...: model.fit(train_X,train_y)
C:\Users\ATIV\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
Out[12]:
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

Q11. 모델을 테스트 하여 training accuracy 와 test accuracy를 보이시오

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
from sklearn.metrics import accuracy_score

#train
pred_y=model.predict(train_X)
acc=accuracy_score(train_y,pred_y)
print('training Accuracy:{0:3f}'.format(acc))

#test
pred_y=model.predict(test_X)
acc=accuracy_score(test_y,pred_y)
print('test Accuracy:{0:3f}'.format(acc))
```

실행화면 캡쳐:

```
In [14]: from sklearn.metrics import accuracy_score
    ...:
    ...: #train
    ...: pred_y=model.predict(train_X)
    ...: acc=accuracy_score(train_y,pred_y)
    ...: print('training Accuracy:{0:3f}'.format(acc))
    ...:
    ...: #test
    ...: pred_y=model.predict(test_X)
    ...: acc=accuracy_score(test_y,pred_y)
    ...: print('test Accuracy:{0:3f}'.format(acc))
training Accuracy:0.667857
test Accuracy:0.750000
```

Q12. gre, gpa, rank 가 다음과 같을 때 합격 여부를 예측하여 보이시오

gre	gpa	rank
400	3.5	5
550	3.8	2
700	4.0	2

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
test_x=np.array([[400,3.5,5],
                 [550,3.8,2],
                 [700,4.0,2]])
model.predict(test_x)
```

실행화면 캡쳐:

```
In [16]: test_x=np.array([[400,3.5,5],
    ...:                  [550,3.8,2],
    ...:                  [700,4.0,2]])
    ...: model.predict(test_x)
Out[16]: array([0, 0, 0], dtype=int64)
```

Q13.이번에는 gre, gpa만 가지고 합격 여부를 예측하는 모델을 만드시오
(train, test를 나누되 test 의 비율은 30% 로 하고 random_state 는 1234 로 한다)

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
df=pd.read_csv("C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_09
14/ucla_admit.csv ")
df_X=df[[ 'gre',  'gpa']]
df_y=df.admit
#데이터 분할
train_X,test_X,train_y,test_y=train_test_split(df_X,df_y,test_size=
0.3,random_state=1234)

#학습
model=LogisticRegression()
model.fit(train_X,train_y)
```

실행화면 캡쳐:

```
In [17]: df=pd.read_csv("C:/Users/ATIV/Desktop/Deeplearning_Cloud/dataset_0914/ucla_admit.csv ")
    ...: df_X=df[[ 'gre',  'gpa']]
    ...: df_y=df.admit
    ...: #데이터 분할
    ...:
train_X,test_X,train_y,test_y=train_test_split(df_X,df_y,test_size=0.3,random_state=1234)
    ...:
    ...: #학습
    ...: model=LogisticRegression()
    ...: model.fit(train_X,train_y)
C:\Users\ATIV\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
Out[17]:
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

Q14. 모델을 테스트 하여 training accuracy 와 test accuracy를 보이시오

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
from sklearn.metrics import accuracy_score

#train
pred_y=model.predict(train_X)
acc=accuracy_score(train_y,pred_y)
print('training Accuracy:{0:3f}'.format(acc))

#test
pred_y=model.predict(test_X)
acc=accuracy_score(test_y,pred_y)
print('test Accuracy:{0:3f}'.format(acc))
```

실행화면 캡쳐:

```
In [18]: from sklearn.metrics import accuracy_score
    ...:
    ...: #train
    ...: pred_y=model.predict(train_X)
    ...: acc=accuracy_score(train_y,pred_y)
    ...: print('training Accuracy:{0:3f}'.format(acc))
    ...:
    ...: #test
    ...: pred_y=model.predict(test_X)
    ...: acc=accuracy_score(test_y,pred_y)
    ...: print('test Accuracy:{0:3f}'.format(acc))
training Accuracy:0.625000
test Accuracy:0.816667
```

Q15. 3가지 변수로 모델을 만든 경우와 2가지 변수로 모델을 만든 경우를 비교하여 어떤 모델이 더 좋은 모델인지 자신의 의견을 제시하시오

2가지 변수로 모델을 만든 경우가 더 좋은 모델이다.

- 3가지 변수로 모델을 만든 경우의 test Accuracy : 0.75
- 2가지 변수로 모델을 만든 경우의 test Accuracy : 0.82

비록 training accuracy는 3가지 변수로 만든 모델이 더 높으나, test accuracy가 높은 모델이 일반화가 더 잘된 모델이라는 의미이므로 2가지변수로 만든 모델이 더 좋은 모델이라고 할 수 있다.