4주. Regression			
학번	32183164	이름	이석현

※ 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
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
Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
#print(Boston)
lstat = Boston['lstat']
medv  = Boston['medv']
lstat = np.array(lstat).reshape(506,1)
medv  = np.array(medv).reshape(506,1)
model =  LinearRegression()
model.fit(lstat, medv)
pred_y = model.predict(lstat)
#print(pred_y)
print('Coefficients: {0:.2f}, Intercept {1:.3f}'
     .format(model.coef_[0][0], model.intercept_[0]))
print('Mean squared error: {0:.2f}'.format(mean_squared_error(medv,
pred_y)))
print('Coefficient of determination: %.2f'% r2_score(medv, pred_y))
plt.scatter(lstat, medv,  color='black')
plt.plot(lstat, medv, color='blue', linewidth=0.1)
plt.xlabel('lstat')
plt.ylabel('medv')
plt.show()
```

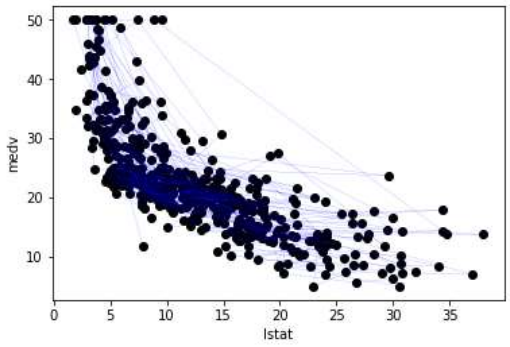
실행화면 캡쳐:

```
In [21]: print('Coefficients: {0:.2f}, Intercept {1:.3f}'.format(model.coef_[0][0],
model.intercept_[0]))
Coefficients: -0.95, Intercept 34.554

In [22]: print('Mean squared error: {0:.2f}'.format(mean_squared_error(medv, pred_y)))
Mean squared error: 38.48

In [23]: print('Coefficient of determination: %.2f'% r2_score(medv, pred_y))
Coefficient of determination: 0.54

In [24]: runfile('C:/Users/이석현/Desktop/4. Regression.py', wdir='C:/Users/이석현/Desktop')
Coefficients: -0.95, Intercept 34.554
Mean squared error: 38.48
Coefficient of determination: 0.54
```



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

medv = -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([[2.0]]))
print(model.predict([[3.0]]))
print(model.predict([[4.0]]))
print(model.predict([[5.0]]))
```

실행화면 캡쳐:

```
In [25]: print(model.predict([[2.0]]))
    ...: print(model.predict([[3.0]]))
    ...: print(model.predict([[4.0]]))
    ...: print(model.predict([[5.0]]))
[[ 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
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
Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
#print(Boston)
lstat = Boston['lstat']
medv  = Boston['medv']
lstat = np.array(lstat).reshape(506,1)
medv  = np.array(medv).reshape(506,1)
model =  LinearRegression()
model.fit(lstat, medv)
pred_y = model.predict(lstat)
#print(pred_y)
print('Coefficients: {0:.2f}, Intercept {1:.3f}'
     .format(model.coef_[0][0], model.intercept_[0]))
print('Mean squared error: {0:.2f}'.format(mean_squared_error(medv,
pred_y)))
```

실행화면 캡쳐:

```
In [27]: 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
    ...:
    ...: Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
    ...: #print(Boston)
    ...:
    ...: lstat = Boston['lstat']
    ...: medv  = Boston['medv']
    ...:
    ...: lstat = np.array(lstat).reshape(506,1)
    ...: medv  = np.array(medv).reshape(506,1)
    ...:
    ...: model =  LinearRegression()
    ...: model.fit(lstat, medv)
    ...:
    ...: pred_y = model.predict(lstat)
    ...: #print(pred_y)
    ...:
    ...: print('Coefficients: {0:.2f}, Intercept {1:.3f}'
    ...:       .format(model.coef_[0][0], model.intercept_[0]))
    ...:
    ...: # The mean squared error
    ...: print('Mean squared error: {0:.2f}'.format(mean_squared_error(medv, pred_y)))
Coefficients: -0.95, Intercept 34.554
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 :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
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

Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
#print(Boston)

ptratio = Boston['ptratio']
tax = Boston['tax']
rad = Boston['rad']
lstat = Boston['lstat']
medv  = Boston['medv']

Boston_X = Boston[['ptratio','tax','rad','lstat']]
Boston_y = Boston['medv']

model =  LinearRegression()
model.fit(Boston_X, Boston_y)
pred_y = model.predict(Boston_X)
#print(pred_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_))

print('Mean squared error: {0:.2f}'.\
     format(mean_squared_error(Boston_y, pred_y)))

print('Coefficient of determination: %.2f'
     % r2_score(Boston_y, pred_y))
```

실행화면 캡쳐:

```
In [44]: runfile('C:/Users/이석현/Desktop/4. Regression.py', wdir='C:/Users/이석현/Desktop')
Coefficients: -1.23,-0.02,0.33,-0.81 Intercept 58.546
Mean squared error: 31.80
Coefficient of determination: 0.62
```

Q6. 모델에서 만들어진 회귀식을 쓰시오

medv = -1.23*ptratio -0.02*tax +0.33*rad -0.81*lstat + 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

```
In [45]: print(model.predict([[14,296,1,2.0]]))
[ 35.5479738]

In [46]: print(model.predict([[15,222,2,3.0]]))
[ 34.95427204]

In [47]: print(model.predict([[15,250,3,4.0]]))
[ 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
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

Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
#print(Boston)

ptratio = Boston['ptratio']
tax = Boston['tax']
rad = Boston['rad']
lstat = Boston['lstat']
medv  = Boston['medv']

Boston_X = Boston[['ptratio','tax','rad','lstat']]
Boston_y = Boston['medv']

model =  LinearRegression()
model.fit(Boston_X, Boston_y)
pred_y = model.predict(Boston_X)
#print(pred_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_))

print('Mean squared error: {0:.2f}'.\
     format(mean_squared_error(Boston_y, pred_y)))
```

실행화면 캡쳐:

```
In [48]: 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
    ...:
    ...: Boston = pd.read_csv('d:/data/dataset_0914/BostonHousing.csv')
    ...: #print(Boston)
    ...:
    ...: ptratio = Boston['ptratio']
    ...: tax = Boston['tax']
    ...: rad = Boston['rad']
    ...: lstat = Boston['lstat']
    ...: medv  = Boston['medv']
    ...:
    ...: Boston_X = Boston[['ptratio','tax','rad','lstat']]
    ...: Boston_y = Boston['medv']
    ...:
    ...: model =  LinearRegression()
    ...: model.fit(Boston_X, Boston_y)
    ...: pred_y = model.predict(Boston_X)
    ...: #print(pred_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_))
    ...:
    ...: print('Mean squared error: {0:.2f}'.\
    ...:       format(mean_squared_error(Boston_y, pred_y)))
Coefficients: -1.23,-0.02,0.33,-0.81 Intercept 58.546
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?

변수의 개수가 많을수록 모델이 더 많은 부분을 설명할 수 있게 되고 보다 정확할 수 있기 때문에 4개의 독립변수를 가지고 다중회귀분석한 모델이 일반적으로 더 좋은 모델이라고 할 수 있을 것 같습니다. 하지만 어느 모델이 더 좋은지 판단하기 위해서는 사용되는 변수가 분석에 적절한 변수인지 고려해주어야 한다고 생각합니다.

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
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

ucla_admit = pd.read_csv('d:/data/dataset_0914/ucla_admit.csv')
ucla_admit_X = ucla_admit[['gre','gpa','rank']]
ucla_admit_y = ucla_admit['admit']
train_X, test_X, train_y, test_y = \
   train_test_split(ucla_admit_X, ucla_admit_y, test_size=0.3,\
                    random_state=1234)
model =  LogisticRegression()
model.fit(train_X, train_y)
pred_y = model.predict(test_X)
```

실행화면 캡쳐:

```
In [68]: import matplotlib.pyplot as plt
    ...: import numpy as np
    ...: import pandas as pd
    ...: from sklearn import datasets
    ...: from sklearn.linear_model import LogisticRegression
    ...: from sklearn.model_selection import train_test_split
    ...: from sklearn.metrics import accuracy_score
    ...:
    ...: ucla_admit = pd.read_csv('d:/data/dataset_0914/ucla_admit.csv')
    ...: #print(ucla_admit)
    ...:
    ...: ucla_admit_X = ucla_admit[['gre','gpa','rank']]
    ...: ucla_admit_y = ucla_admit['admit']
    ...:
    ...: train_X, test_X, train_y, test_y = \
    ...:     train_test_split(ucla_admit_X, ucla_admit_y, test_size=0.3,\
    ...:                      random_state=1234)
    ...:
    ...: # Define learning model
    ...: model =  LogisticRegression()
    ...:
    ...: # Train the model using the training sets
    ...: model.fit(train_X, train_y)
    ...:
    ...:
    ...:
    ...:
    ...: # Make predictions using the testing set
    ...: pred_y = model.predict(test_X)
```

Q11. 모델을 테스트 하여 training accuracy 와 test accuracy를 보이시오

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
pred_y = model.predict(test_X)
test_acc = accuracy_score(test_y, pred_y)
print('Test Accuracy : {0:3f}'.format(test_acc))

pred_y = model.predict(train_X)
training_acc = accuracy_score(train_y, pred_y)
print('Training Accuracy : {0:3f}'.format(training_acc))
```

실행화면 캡쳐:

```
In [78]: pred_y = model.predict(test_X)
    ...: #print(pred_y)
    ...:
    ...:
    ...: test_acc = accuracy_score(test_y, pred_y)
    ...: print('Test Accuracy : {0:3f}'.format(test_acc))
    ...:
    ...: pred_y = model.predict(train_X)
    ...: training_acc = accuracy_score(train_y, pred_y)
    ...: print('Training Accuracy : {0:3f}'.format(training_acc))
Test Accuracy : 0.750000
Training Accuracy : 0.667857
```

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
print(model.predict([[400,3.5,5]]))
print(model.predict([[550,3.8,2]]))
print(model.predict([[700,4.0,2]]))
```

실행화면 캡쳐:

```
In [71]: print(model.predict([[400,3.5,5]]))
    ...: print(model.predict([[550,3.8,2]]))
    ...: print(model.predict([[700,4.0,2]]))
[0]
[0]
[0]
```

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
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

ucla_admit = pd.read_csv('d:/data/dataset_0914/ucla_admit.csv')
#print(ucla_admit)
ucla_admit_X = ucla_admit[['gre','gpa']]
ucla_admit_y = ucla_admit['admit']
train_X, test_X, train_y, test_y = \
   train_test_split(ucla_admit_X, ucla_admit_y, test_size=0.3,\
                    random_state=1234)
model =  LogisticRegression()
model.fit(train_X, train_y)
pred_y = model.predict(test_X)
```

실행화면 캡쳐:

```
In [1]: import matplotlib.pyplot as plt
   ...: import numpy as np
   ...: import pandas as pd
   ...: from sklearn import datasets
   ...: from sklearn.linear_model import LogisticRegression
   ...: from sklearn.model_selection import train_test_split
   ...: from sklearn.metrics import accuracy_score
   ...:
   ...: ucla_admit = pd.read_csv('d:/data/dataset_0914/ucla_admit.csv')
   ...: #print(ucla_admit)
   ...:
   ...: ucla_admit_X = ucla_admit[['gre','gpa']]
   ...: ucla_admit_y = ucla_admit['admit']
   ...:
   ...: train_X, test_X, train_y, test_y = \
   ...:     train_test_split(ucla_admit_X, ucla_admit_y, test_size=0.3,\
   ...:                      random_state=1234)
   ...:
   ...: # Define learning model
   ...: model =  LogisticRegression()
   ...:
   ...: # Train the model using the training sets
   ...: model.fit(train_X, train_y)
   ...:
   ...:
   ...:
   ...:
   ...: # Make predictions using the testing set
   ...: pred_y = model.predict(test_X)
```

Q14. 모델을 테스트 하여 training accuracy 와 test accuracy를 보이시오

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
pred_y = model.predict(test_X)
test_acc = accuracy_score(test_y, pred_y)
print('Test Accuracy : {0:3f}'.format(test_acc))
pred_y = model.predict(train_X)
training_acc = accuracy_score(train_y, pred_y)
print('Training Accuracy : {0:3f}'.format(training_acc))
```

실행화면 캡쳐:

```
In [76]: pred_y = model.predict(test_X)
    ...: #print(pred_y)
    ...:
    ...: test_acc = accuracy_score(test_y, pred_y)
    ...: print('Test Accuracy : {0:3f}'.format(test_acc))
    ...:
    ...: pred_y = model.predict(train_X)
    ...: training_acc = accuracy_score(train_y, pred_y)
    ...: print('Training Accuracy : {0:3f}'.format(training_acc))
Test Accuracy : 0.816667
Training Accuracy : 0.625000
```

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 모델이라고 생각합니다. 3가지 변수로 모델을 만들었을 때가 2가지 변수로 모델을 만들었을 때 보다 Training Accuracy는 더 높지만, Test Accuracy는 2가지 변수로 모델을 만들었을 때 더 높았기 때문에 2가지 변수로 만들어진 모델이 보다 정확하다고 생각합니다.