

## 4 주. Regression

학번	32191197	이름	김채은
----	----------	----	-----

※ 이번 실습에 사용된 데이터셋은 공지에 있는 데이터셋 압축파일에 포함되어 있음

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

BostonHousing dataset 을 가지고 단순 선형 회귀 분석을 하고자 한다.

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

Source code :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

data=pd.read_csv('C:/Users/82104/Desktop/deeplearning/BostonHousing.csv')

x = data['lstat']
y = data['medv']
x = np.array(x).reshape(506, 1)
y = np.array(y).reshape(506, 1)

model = LinearRegression()
model.fit(x, y)

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

실행화면 캡처:

```
In [3]: runfile('C:/Users/82104/.spyder-py3/temp.py', wdir='C:/Users/
82104/.spyder-py3')
Coefficients: -0.95, Intercept: 34.554
```

Q2. 모델에서 만들어진 회귀식을 쓰시오 (medv = 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 medv 의 값을 예측하여 제시하시오.

Source code :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

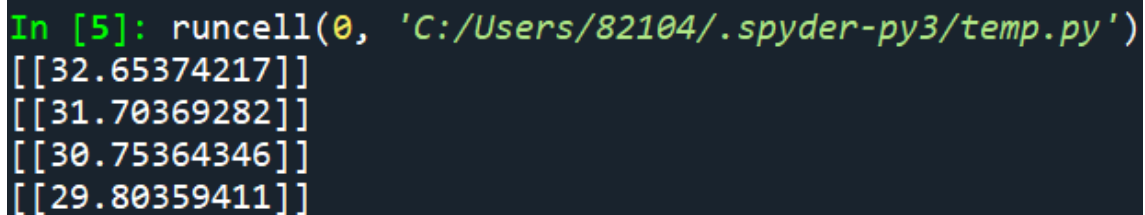
data = pd.read_csv('C:/Users/82104/Desktop/deeplearning/BostonHousing.csv')

x = data['lstat']
y = data['medv']
x = np.array(x).reshape(506, 1)
y = np.array(y).reshape(506, 1)

model = LinearRegression()
model.fit(x, y)

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

실행화면 캡처:



```
In [5]: runcell(0, 'C:/Users/82104/.spyder-py3/temp.py')
[[32.65374217]]
[[31.70369282]]
[[30.75364346]]
[[29.80359411]]
```

Q4. 데이터셋의 모든 lstat 값을 회귀식에 넣어 medv 의 값을 예측 한 뒤 mean square error 를 계산하여 제시하시오

Source code :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('C:/Users/82104/Desktop/deeplearning/BostonHousing.csv')
```

```

x = data['lstat']
y = data['medv']
x = np.array(x).reshape(506, 1)
y = np.array(y).reshape(506, 1)

model = LinearRegression()
model.fit(x, y)

pred_y = model.predict(x)
print('Mean squared error: {0:.2f}'.format(mean_squared_error(y,
pred_y)))

```

실행화면 캡처:

```

In [7]: runcell(0, 'C:/Users/82104/.spyder-py3/temp.py')
Mean squared error: 38.48

```

BostonHousing dataset 을 가지고 다중 선형 회귀 분석을 하고자 한다.

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

Source code :

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('C:/Users/82104/Desktop/deeplearning/BostonHousing.csv')

x = data[['lstat', 'ptratio', 'tax', 'rad']]
y = data['medv']

model = LinearRegression()
model.fit(x, 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 캡처:

```
In [15]: runcell(0, 'C:/Users/82104/.spyder-py3/temp.py')
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 :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('C:/Users/82104/Desktop/deeplearning/BostonHousing.csv')

x = data[['lstat', 'ptratio', 'tax', 'rad']]
y = data['medv']

model = LinearRegression()
model.fit(x, y)

print(model.predict(np.array([2.0, 14, 296, 1]).reshape(1, -1)))
print(model.predict(np.array([3.0, 15, 222, 2]).reshape(1, -1)))
print(model.predict(np.array([4.0, 15, 250, 3]).reshape(1, -1)))
```

실행하면 캡처:

```
In [23]: runcell(0, 'C:/Users/82104/.spyder-py3/temp.py')
[35.5479738]
[34.95427204]
[34.04856204]
```

Q8. 데이터셋의 모든 lstat, ptratio, tax, rad 값을 회귀식에 넣어 mdev 의 값을 예측 한 뒤 mean square error 를 계산하여 제시하시오.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('C:/Users/82104/Desktop/deeplearning/BostonHousing.csv')

x = data[['lstat', 'ptratio', 'tax', 'rad']]
y = data['medv']

model = LinearRegression()
model.fit(x, y)

pred_y = model.predict(x)

print('Mean squared error: {}'.format(mean_squared_error(y, pred_y)))
```

실행화면 캡처:

```
In [24]: runcell(0, 'C:/Users/82104/.spyder-py3/temp.py')
Mean squared error: 31.80
```

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

4 개의 변수를 가지고 모델을 만든 경우를 더 좋은 모델이라고 할 수 있다. Mean squared error 가 lstat 하나만 가지고 만든 모델보다, 4 개의 변수를 가지고 만든 모델이 더 작기 때문이다.

**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 해결하십시오

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

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

data = pd.read_csv('C:/Users/82104/Desktop/deeplearning/ucla_admit.csv')

x = data[['gre', 'gpa', 'rank']]
y = data['admit']

train_x, test_x, train_y, test_y = train_test_split(x, y,
test_size=0.3, random_state=1234)

model = LogisticRegression()
model.fit(train_x, train_y)

pred_y = model.predict(test_x)
print(pred_y)
```

**실행화면 캡처:**

[illegible]

Q11. 모델을 테스트 하여 training accuracy 와 test accuracy 를 보이시오

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

data = pd.read_csv('C:/Users/82104/Desktop/deeplearning/ucla_admit.csv')
```

```

x = data[['gre', 'gpa', 'rank']]
y = data['admit']

train_x, test_x, train_y, test_y = train_test_split(x, y,
test_size=0.3, random_state=1234)

model = LogisticRegression()
model.fit(train_x, train_y)

pred_tr_y = model.predict(train_x)
pred_y = model.predict(test_x)

training_acc = accuracy_score(train_y, pred_tr_y)
print('Accuracy : {0:3f}'.format(training_acc))

test_acc = accuracy_score(test_y, pred_y)
print('Accuracy : {0:3f}'.format(test_acc))

```

실행화면 캡처:

```

In [42]: runcell(0, 'C:/Users/82104/.spyder-py3/temp.py')
Accuracy : 0.671429
Accuracy : 0.741667

```

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

```

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

data =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/ucla_admit.csv')

```

```

x = data[['gre', 'gpa', 'rank']]
y = data['admit']

train_x, test_x, train_y, test_y = train_test_split(x, y,
test_size=0.3, random_state=1234)

model = LogisticRegression()
model.fit(train_x, train_y)

pred_y = model.predict(np.array([400, 3.5, 5]).reshape(1, -1))
print(pred_y)
pred_y = model.predict(np.array([550, 3.8, 2]).reshape(1, -1))
print(pred_y)
pred_y = model.predict(np.array([700, 4.0, 2]).reshape(1, -1))
print(pred_y)

```

실행하면 캡처:

```

In [48]: runcell(0, 'C:/Users/82104/.spyder-py3/temp.py')
[0]
[0]
[0]

```

Q13.이번에는 **gre, gpa** 만 가지고 합격 여부를 예측하는 모델을 만드시오

(train, test 를 나누되 test 의 비율은 30% 로 하고 random\_state 는 1234 로 한다)

```

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

data = pd.read_csv('C:/Users/82104/Desktop/deeplearning/ucla_admit.csv')

x = data[['gre', 'gpa']]
y = data['admit']

train_x, test_x, train_y, test_y = train_test_split(x, y,
test_size=0.3, random_state=1234)

model = LogisticRegression()
model.fit(train_x, train_y)

pred_y = model.predict(test_x)

```



```
print(pred_y)
```

실행화면 캡처:

```
In [50]: runcell(0, 'C:/Users/82104/.spyder-py3/temp.py')  
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 0 0]
```

Q14. 모델을 테스트 하여 training accuracy 와 test accuracy 를 보이시오

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

data =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/ucla_admit.csv')

x = data[['gre', 'gpa']]
y = data['admit']

train_x, test_x, train_y, test_y = train_test_split(x, y,
test_size=0.3, random_state=1234)

model = LogisticRegression()
model.fit(train_x, train_y)

training_acc = accuracy_score(train_y, pred_tr_y)
print('Accuracy : {0:3f}'.format(training_acc))

test_acc = accuracy_score(test_y, pred_y)
print('Accuracy : {0:3f}'.format(test_acc))
```

실행화면 캡처:

```
In [51]: runcell(0, 'C:/Users/82104/.spyder-py3/temp.py')
Accuracy : 0.671429
Accuracy : 0.825000
```

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

Accuracy가 크다는 것은 예측과 정답이 일치하는 확률이 높다는 의미이므로 accuracy가 높은 모델이 더 좋은 모델이다. 3 가지 변수로 모델을 만든 경우가 2 가지 변수로 모델을 만든 경우보다 accuracy가 높기 때문에 더 좋은 모델이라고 할 수 있다.