| 4주. Regression | | | |
|---|---|---|---|
| 학번 | 32171373 | 이름 | 노병우 |

※ 이번 실습에 사용된 데이터셋은 공지에 있는 데이터셋 압축파일에 포함되어 있음

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

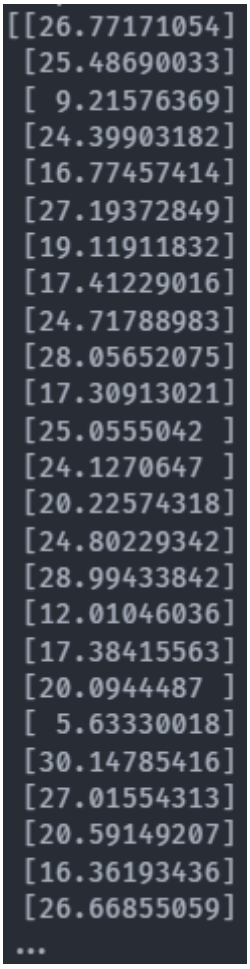
**BostonHousing dataset을 가지고 단순 선형 회귀 분석을 하고자 한다.**

Q1 lstat (소득분위가 하위인 사람들의 비율) 로 medv (주택가격)을 예측하는 단순 선형 회귀 모델을 만드시오 (tain:test = 7:3, random_state는 1234). 모델의 내용을 보이시오

Source code :

```
# Package import 는 생략
house = pd.read_csv('data/BostonHousing.csv')

lstat = house["lstat"]
medv = house["medv"]

lstat = np.array(lstat).reshape(506,1)
medv = np.array(medv).reshape(506,1)

# 데이터셋 train:test = 7:3, random_state = 1234
train_X, test_X, train_y, test_y = train_test_split(lstat,
medv, test_size=0.3, random_state=1234)

model = LinearRegression()
model.fit(train_X, train_y)
pred_y = model.predict(test_X)
```

**실행화면 캡쳐:**




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

Medv = -0.94 * lstat + 34.321

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

Source code :

```
q3_test_X = [[2.0], [3.0], [4.0]]
print(model.predict(q3_test_X))
```

**실행화면 캡쳐:**

```
[[32.44550746]
 [31.50768979]
 [30.56987212]]
```

Q4. 모델에 대해 rooted mean square error (RMSE)와 R2score를 보이시오

Source code :

```
print('Mean squared error:
{0:.2f}'.format(mean_squared_error(test_y, pred_y)))
print('Coefficient of determination: %.2f'%
r2_score(test_y, pred_y))
```

**실행화면 캡쳐:**

```
Mean squared error: 40.44
Coefficient of determination: 0.56
```

**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 = 7:3, random_state는 1234)). 모델의 내용을 보이시오

Source code :

```
house_X = house[['lstat','ptratio','tax', 'rad']]
house_y = house['mdev']

house_X = np.array(house_X).reshape(102,3)
house_y = np.array(house_y).reshape(102,1)
# 데이터셋 train:test = 7:3, random_state = 1234
train_X, test_X, train_y, test_y =
train_test_split(house_X, house_y,
test_size=0.3,random_state=1234)
```

```
model = LinearRegression()
model.fit(train_X, train_y)
pred_y = model.predict(test_X)
```

**실행화면 캡쳐:**

```
[[26.66526457]
 [20.71363008]
 [ 9.88846515]
 [22.97641169]
 [16.40414453]
 [29.44814266]
 [18.79770607]
 [16.95385445]
 [26.56863507]
 [30.00361065]
 [16.86493079]
 [23.38993174]
 [23.79129591]
 [14.62163105]
 [23.96626866]
 [27.11309458]
 [12.29748804]
 [17.97892188]
 [19.26586971]
 [ 6.80038881]
 [28.73873892]
 [24.58748964]
 [19.69432009]
 [21.27169903]
 [26.60229641]
...
 [29.82000525]
 [ 6.76805294]
 [18.40969878]
 [24.82533123]]
```

Q6. 모델에서 만들어진 회귀식을 쓰시오

Medv = -0.81 * X + 59.262

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 :

```
q7_test_X = np.array([[2.0, 14, 296, 1],[3.0, 15, 222,
2],[4.0, 15, 250, 3]]).reshape(3,4)
print(model.predict(q7_test_X))
```

**실행화면 캡쳐:**

```
[[35.49774089]
 [34.90871561]
 [34.01764254]]
```

Q8. 모델에 대해 rooted mean square error (RMSE)와 R2score를 보이시오

```
print('Mean squared error:
{0:.2f}'.format(mean_squared_error(test_y, pred_y)))
print('Coefficient of determination: %.2f'%
r2_score(test_y, pred_y))
```

**실행화면 캡쳐:**

```
Mean squared error: 34.49
Coefficient of determination: 0.63
```

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

후자가 더 좋은 모델이라고 할 수 있다. 이와 같이 생각한 이유는 Medv의 값이 Lstat 값 하나로 결정되는 종속변수가 아니기 때문이다. Ptratio, Tax, Rad와 같은 변수 또한 Medv의 값에 영향을 주기 때문이다. 간단히 말해, Tax(세금 과세 비율)과 Rad(접근 가능한 고속도로 지수)는 집 값에 대해 충분한 영향을 미칠 수 있다. 같은 lstat이라도 Rad, Tax 지수에 따라 다를 수 있기 때문에 lstat만 고려한 모델 보다 많은 종속 변수를 고려한 모델이 현실에 가깝게 반영한 모델이라고 할 수 있기 때문이다.

추가적으로, lstat만을 변수로한 모델의 r2 점수는 0.56 인데 반해 4개의 변수로한 모델의 r2 점수는 0.63으로 보다 높은 정확도를 보임을 알 수 있다. 이 또한, 후자가 더 좋은 모델임에 대한 근거가 될 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, rank를 가지고 합격여부를 예측하는 logistic regression 모델을 만드시오. (tain:test = 7:3, random_state는 1234).

```
from sklearn.linear_model import LogisticRegression
ucla = pd.read_csv("data/ucla_admit.csv")
ucla_X = ucla[["gre", "gpa", "rank"]]
ucla_y = ucla["admit"]
```

```
train_X, test_X, train_y, test_y =
train_test_split(ucla_X, ucla_y, test_size=0.3,
random_state=1234) # train : test = 7 : 3

model = LogisticRegression()
model.fit(train_X, train_y)
pred_y = model.predict(test_X)
print(pred_y)
```

**실행화면 캡쳐:**

```
[0 0 1 0 0 0 0 0 0 0 1 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 0 0 0 0 1 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 1 0 0 0 1 1 1 1 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1
 1 0 0 0 0 0 0 0 0]
```

Q11. 모델을 테스트 하여 training accuracy 와 test accuracy를 보이시오

```
pred_y = model.predict(test_X)
acc = accuracy_score(test_y, pred_y)
print('Test Accuracy : {0:3f}'.format(acc))
pred_y = model.predict(train_X)
acc = accuracy_score(train_y, pred_y)
print('Train Accuracy : {0:3f}'.format(acc))
```

**실행화면 캡쳐:**

```
Test Accuracy : 0.741667
Train Accuracy : 0.671429
```

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 |

```
q12_test_X = np.array([[400, 3.5, 5], [550, 3.8, 2],
[700, 4.0, 2]]).reshape(3,3)
print(model.predict(q12_test_X))
```

**실행화면 캡쳐:**

```
[0 0 0]
```

Q13.이번에는 **gre, gpa만 가지고 합격 여부를 예측하는 모델을 만드시오**
(tain:test = 7:3, random_state는 1234).

```
ucla_X = ucla[["gre","gpa"]]
ucla_y = ucla["admit"]
train_X, test_X, train_y, test_y =
train_test_split(ucla_X, ucla_y, test_size=0.3,
random_state=1234) # train : test = 7 : 3
model.fit(train_X, train_y)
pred_y = model.predict(test_X)
print(pred_y)
```

**실행화면 캡쳐:**

`

```
[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 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 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 0 0 0 0 0 0 0]
```

Q14. 모델을 테스트 하여 training accuracy 와 test accuracy를 보이시오

```
pred_y = model.predict(test_X)
acc = accuracy_score(test_y, pred_y)
print('Test Accuracy : {0:3f}'.format(acc))
pred_y = model.predict(train_X)
acc = accuracy_score(train_y, pred_y)
print('Train Accuracy : {0:3f}'.format(acc))
```

**실행화면 캡쳐:**

```
Test Accuracy : 0.825000
Train Accuracy : 0.625000
```

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

들어가기 앞서, 3가지 변수(gre, gpa, rank)모델 2가지 변수(gre, gpa)모델의 정확도를 비교해보자. 3가지 변수의 Accuracy는 0.741, 2가지 변수의 Accuracy는 0.825 이다. 놀랍게도 이전 housing 데이터의 학습모델과 달리 더 적은 변수를 이용한 학습 모델의 정확도가 더 높게 나왔다. Rank가 Admit 예측에 필요한 적절한 변수인지에 대한 논의가 필요할 것으로 보인다. 이에 대한 개인적인 의견은 이렇다. Rank는 학교 석차로 입학전 각 학교에 대한 석차로 볼 수 있다. 물론, 높은 석차를 가진 학생이 입학할 가능성이 높다고 할 순 있지만, 학교별 수준 차이가 심할 경우 편차가 클 것으로 생각된다. 때문에, 학교 내 석차를 변수로서 사용하기에는 적절하지 않다.

이러한 근거로 3가지 변수로 모델을 만든 경우보다 Rank를 제외한 2가지 변수로 모델을 만드는 경우가 더 좋은 모델이라고 생각한다.