201645084_3_Q_노영훈

7주차 과제 - 농어의 길이, 높이, 폭으로 무게를 예측하는 최적의 머신러닝 모델을 만드시오
; 최적의 특성 갯수를 찾으시오(그래프를 그림)

# 1. 데이터 설정.

## - 농어의 무게

```
# 예측할 무게
perch_weight = np.array(
    [5.9 , 32.0 , 40.0 , 51.5 , 70.0 , 100.0 , 78.0 , 80.0 , 85.0 , 85.0 ,
     110.0 , 115.0 , 125.0 , 130.0 , 120.0 , 120.0 , 130.0 , 135.0 , 110.0 ,
     130.0 , 150.0 , 145.0 , 150.0 , 170.0 , 225.0 , 145.0 , 188.0 , 180.0 ,
     197.0 , 218.0 , 300.0 , 260.0 , 265.0 , 250.0 , 250.0 , 300.0 , 320.0 ,
     514.0 , 556.0 , 840.0 , 685.0 , 700.0 , 700.0 , 690.0 , 900.0 , 650.0 ,
     820.0 , 850.0 , 900.0 , 1015.0 , 820.0 , 1100.0 , 1000.0 , 1100.0 ,
     1000.0 , 1000.0 ]
     )
```

설명 : 농어의 무게 데이터이다.

## - 농어의 길이, 높이, 폭

https://bit.ly/perch_csv # 농어의 길이, 높이, 폭이 들어가있는 데이터

```
df = pd.read_csv('https://bit.ly/perch_csv')
perch_full = df.to_numpy()
print (perch_full)
```

설명 : 농어의 길이, 높이, 폭 순서대로 3차원 벡터로 형성되어 데이터가 들어가 있다.

## - 데이터 셔플

```
# 데이터 셔플
from  sklearn.model_selection import  train_test_split
train_input, test_input, train_target, test_target = train_test_split(perch_full, perch_weight, ra
ndom_state=42 )
```

설명 : 훈련데이터에 넣어줄 값과 실전데이터에 넣어줄 값을 구별하기 위해서 데이터를 섞는다.

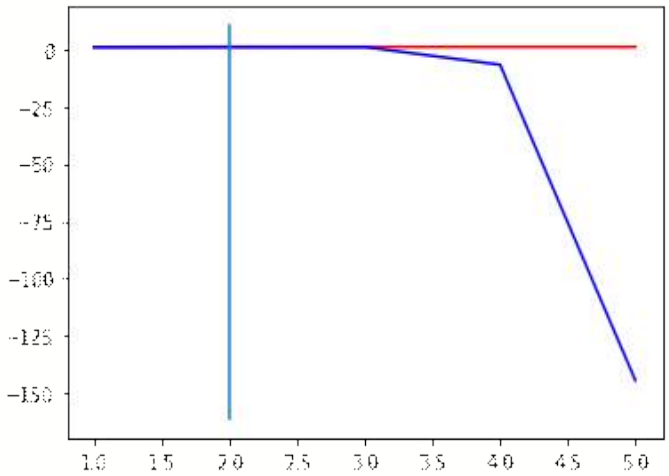
## - 훈련을 위한 라이브러리 받기

```
# 다항 특성 변환기
from  sklearn.preprocessing import  PolynomialFeatures

# 선형회귀모델 생성
from  sklearn.linear_model import  LinearRegression

# 특성 값, 최소 오차율
degree_list = [1 ,2 ,3 ,4 ,5]
min_error_rate = 1
```

설명 : 훈련을 위한 라이브러리를 받는다.

## 2. 최적의 특성 값 찾기(다항 회귀 – 규제 적용 전)

```
# 최적의 특성 값 찾기
for  degree_case in  degree_list:

  # 훈련데이터 다항 특성으로 변환.
  poly = PolynomialFeatures(degree=degree_case, include_bias=False )
  poly.fit(train_input)

  # 특성에 맞게 값 변환
  train_poly = poly.transform(train_input)
  test_poly = poly.transform(test_input)

  # 회귀모델 생성
  lr.fit(train_poly, train_target)

  # 회귀모델 점수 확인
  train_score = lr.score(train_poly,train_target)
  test_score = lr.score(test_poly,test_target)

  # 오차 확인(훈련케이스의 점수와 테스트 케이스의 점수를 뺀 뒤 절대값을 씌운다.)
  error_rate = abs(train_score - test_score)

  # 그래프를 그리기 위한 리스트에 값넣어주기
  train_score_list.append(train_score)
  test_score_list.append(test_score)

  # 오차율이 0~1 사이일때
  if  error_rate >0  and  error_rate<1 :

    # 오차율이 가장 낮은 값이 가장 최적의 모델이다.
    if  min_error_rate >error_rate:
      min_error_rate = error_rate
      min_value = degree_case

# x=min_value선을 비슷하게 그리기 위해서 지정.
x = [min_value]*10
y = [-180  + i*19  for  i in  range (1 ,11 )]

# 그래프 생성
plt.plot(degree_list, train_score_list, color='red')
plt.plot(degree_list, test_score_list, color='blue')
plt.plot(x,y)
plt.show()
```




```
최적의 특성(degree) 값 : 2
```

### - 최적의 특성 값으로 무게 예측.

```
poly = PolynomialFeatures(degree=2 , include_bias=False )
poly.fit(train_input)

# 특성에 맞게 값 변경
train_poly = poly.transform(train_input)
test_poly = poly.transform(test_input)

# 회귀모델 생성
lr.fit(train_poly, train_target)
predict_test = poly.transform([[26 ,7 ,4 ]])
lr.predict(predict_test)
```

결과 : array([211.3607989])
설명 : 최적의 특성 값으로 무게 예측

## 3. 최적의 특성 값 찾기(다항 회귀 – 규제 적용(릿지 회귀))

### - 다항 특성 생성.

```
poly = PolynomialFeatures(degree=5 , include_bias=False )
poly.fit(train_input)
train_poly = poly.transform(train_input)
test_poly = poly.transform(test_input)
```

설명 : 데이터를 최악의 특성값 5로 지정한 다항 특성으로 변환한다.

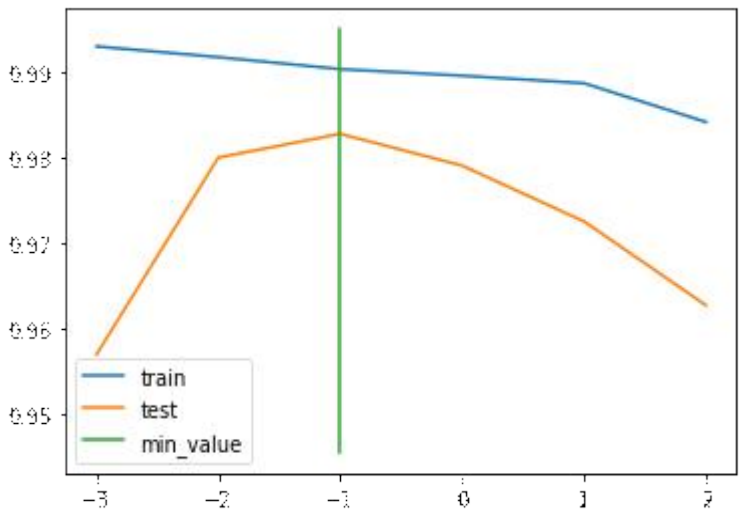
### - 특성의 스케일 조정

```
# 규제 선형모델(ridge)을 위한 특성의 스케일 조정
from  sklearn.preprocessing import  StandardScaler
ss = StandardScaler()
ss.fit(train_poly)
train_scaled = ss.transform(train_poly)
test_scaled = ss.transform(test_poly)
```

설명 : 규제 선형모델을 위해서 특성의 스케일을 조정한다.

## - 최적의 규제 강도 값 찾기 (다항 특성의 특성값이 5일 때)

```
# 규제 강도
alpha_list = [0.001 , 0.01 , 0.1 , 1 , 10 , 100 ]
min_error_rate = 1
min_value = 0
for  alpha in  alpha_list:

    # 릿지 모델을 만듭니다
    ridge = Ridge(alpha=alpha)

    # 릿지 모델을 훈련합니다
    ridge.fit(train_scaled, train_target)

    # 훈련 점수와 테스트 점수를 저장합니다
    train_score = ridge.score(train_scaled, train_target)
    test_score = ridge.score(test_scaled, test_target)
    train_score_ridge_list.append(train_score)
    test_score_ridge_list.append(test_score)

    # 오차율
    error_rate = abs (train_score - test_score)

    # 오차가 가장 적게나오는 값을 찾습니다.
    if  min_error_rate > error_rate:
      min_error_rate = error_rate
      min_value = alpha

# X = min_value를 그리기 위한 그래프 생성.
X =[np.log10(min_value)]*10
Y =[i/i*0.94 +0.0055 *i for  i in  range (1 ,11 )]

# 최적값 그래프 생성.
plt.plot(np.log10(alpha_list), train_score_ridge_list,label="train")
plt.plot(np.log10(alpha_list), test_score_ridge_list,label="test")
plt.plot(X,Y,label="min_value")
plt.legend()
plt.show()
```




```
print(min_value)
```

result : 0.1

설명 : 다항 특성의 특성값이 5일 때 Ridge 회귀를 적용한 경우 최적의 규제 강도는 0.1입니다.