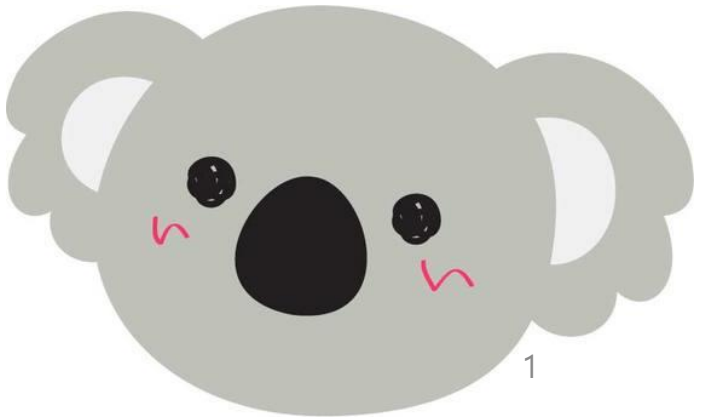
코알라 유니브 코싸인팀
발표자 : 김현솔

Week11_후반부

목차.

Stage3 사이킷런으로 선형회귀모형 구현하기
Stage4 의류 사이즈 추천하기
Challenge2 실제값과 예측값을 비교하는 DataFrame 만들기
Homework2 타이타닉 생존자 예측하기(by Linear Regression)



Stage3 사이킷런으로 선형회귀모형 구현하기

1) Data set 구성하기

Train : Test = 8:2
Train : Valid = 8:2

Train = 모형 적합용 훈련 데이터
Valid = 모형 성능 검증할 중간 데이터
Test = 모형의 최종 성능 평가용

<코드>

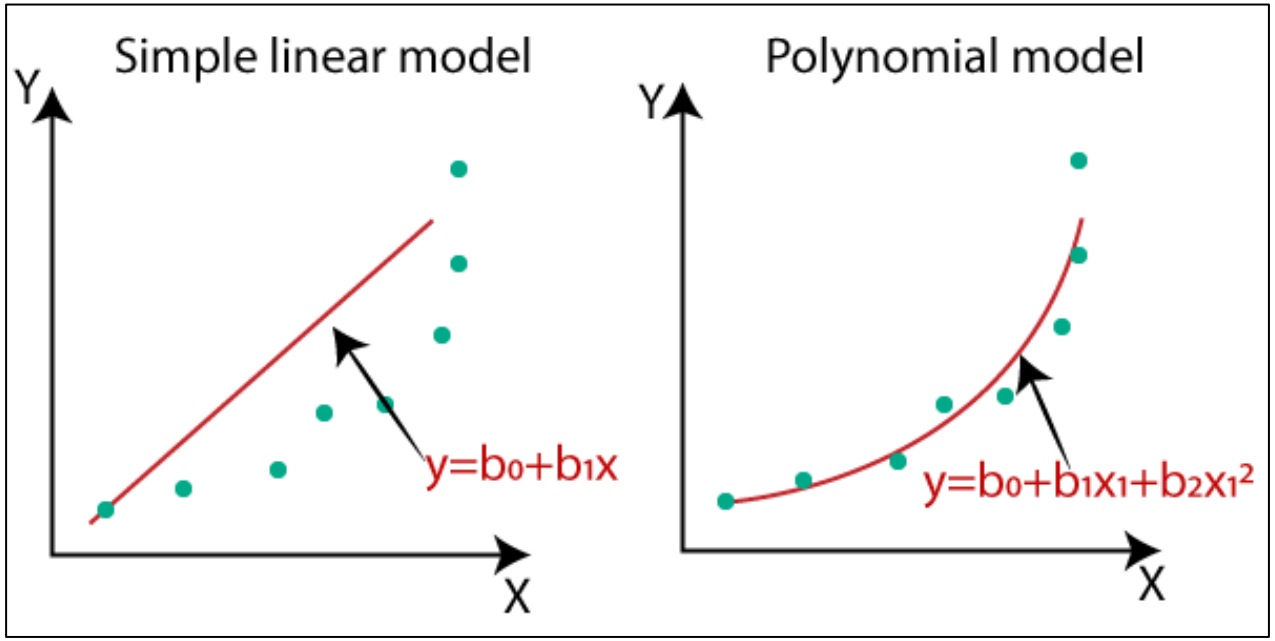
```
In [29]: from sklearn.model_selection import train_test_split
train_data = house_data.drop(['price'], axis=1) #대부분의 특징열을 살리고있음 > 특징열 바꿔가며 모형 성능 확인해보기
target_data = house_data['price']

#train : test = 8:2
x_train, x_test, y_train, y_test = train_test_split(train_data, target_data, test_size=0.2)

#train : valid = 8:2
x_train, x_valid, y_train, y_valid = train_test_split(x_train, y_train, test_size=0.2 )

print('모든데이터', train_data.shape) #모든 데이터의 개수
print('train set', x_train.shape, y_train.shape)  #y의 경우 target data(=price) 하나만 들어있어서 숫자가 생략되어 출력
print('valid set', x_valid.shape, y_valid.shape)
print('test set', x_test.shape, y_test.shape)
```

```
모든데이터 (21613, 18)
train set (13832, 18) (13832,)
valid set (3458, 18) (3458,)
test set (4323, 18) (4323,)
```

train_test_split은 defaul가 0.75:0.25로 데이터를 나눠주므로 값을 바꿔줘야 함!

Stage3 사이킷런으로 선형회귀모형 구현하기

* linear regression vs polynomial regression



cf. 다항식의 차수가 높을수록 모형 성능이 좋아질까? NO

2) Linear Regression 적합하기

```
#일반 선형회귀모형
from sklearn.linear_model import LinearRegression
lr = LinearRegression().fit(x_train, y_train)
print('train set score', lr.score(x_train, y_train))
print('valid set score', lr.score(x_valid, y_valid))
```

```
train set score 0.7041146430088838
valid set score 0.6875034614715918
```

3) Polynomial Regression 적합하기

```
#다항 회귀 모형
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

model = make_pipeline(PolynomialFeatures(2), LinearRegression()) #이차항 넣음
model.fit(x_train, y_train)
print('train set score', model.score(x_train, y_train))
print('valid set score', model.score(x_valid, y_valid))

#3차항이 더 복잡한 모형을 만들 수 있는대신에, 오히려 모형 성능은 떨어짐
```

```
train set score 0.8192156331978423
valid set score 0.8361642047420474
```

모형 성능이 향상!

Stage3 사이킷런으로 선형회귀모형 구현하기

참고. Score 산출

사이킷런은 연속적인 값(ex. 집 값)에 대해서는 상관계수에 기인해서 스코어링을 함 > 결정계수 이용(R-square)

$$R^2 = 1 - \frac{\Sigma(y-\hat{y})^2}{\Sigma\left(y-\bar{y}\right)^2}$$

결정계수(R^2)란?

: 모형을 설명하는 정도를 뜻한다.

- 1에 가까울 수록 모형을 완벽하게 설명함.
- 0에 가까울수록 모형을 잘 설명하지 못함.

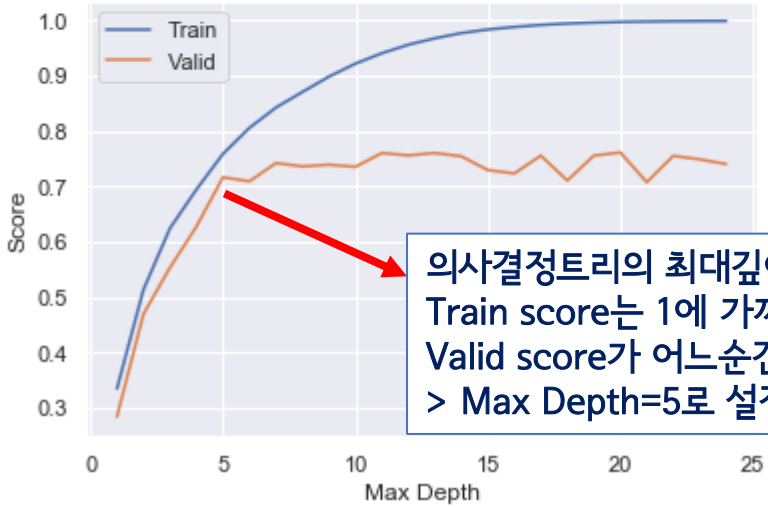
4) Decision Tree Regressor 적합하기

```
#Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor().fit(x_train, y_train)
print("train set score:", tree.score(x_train, y_train))
print("valid set score:", tree.score(x_valid, y_valid))
```

```
train set score: 0.999451587272514
valid set score: 0.7205420299755929
```

5) 적절한 max_depth 찾기!!!

```
#parameter 조율하기
train_score = []
valid_score = []
depth_range = range(1, 25)
for depth in depth_range:
    tree = DecisionTreeRegressor(max_depth=depth).fit(x_train, y_train)
    train_score.append(tree.score(x_train, y_train))
    valid_score.append(tree.score(x_valid, y_valid))

plt.plot(depth_range, train_score)
plt.plot(depth_range, valid_score)
plt.legend(['Train', 'Valid'])

plt.xlabel('Max Depth')
plt.ylabel('Score')

plt.show()
#max_depth = 5로 결정
```



의사결정트리의 최대깊이가 깊어질수록
Train score는 1에 가까워지지만
Valid score가 어느순간 증가하지 않음
> Max Depth=5로 설정

Stage4 의류 사이즈 추천하기

1) Data 불러오기

```
import pandas as pd
df = pd.read_csv('data/korean_players.csv')
df
```

	이름	적합사이즈	나이	출장수	키	몸무게
0	조현우	275	27	12	189	75
1	김영권	285	29	69	186	81
2	김민재	290	22	19	190	88
3	이청용	265	30	89	180	70
4	손흥민	260	26	79	183	77
5	이재성	275	26	43	180	70
6	백승호	280	22	0	180	68
7	이강인	270	18	0	173	63
8	이승우	260	21	10	170	63
9	황의조	275	26	25	184	73

2) 요약통계량 살펴보기

```
df.describe()
```

	적합사이즈	나이	출장수	키	몸무게
count	10.000000	10.000000	10.000000	10.000000	10.000000
mean	273.500000	24.700000	34.600000	181.500000	72.800000
std	10.013879	3.802046	33.383962	6.398785	7.828722
min	260.000000	18.000000	0.000000	170.000000	63.000000
25%	266.250000	22.000000	10.500000	180.000000	68.500000
50%	275.000000	26.000000	22.000000	181.500000	71.500000
75%	278.750000	26.750000	62.500000	185.500000	76.500000
max	290.000000	30.000000	89.000000	190.000000	88.000000

3) Train data 만들기

```
x_train=df.drop(['적합사이즈', '이름'], axis=1)
y_train=df['적합사이즈']
x_train
# y_train
```

	나이	출장수	키	몸무게
0	27	12	189	75
1	29	69	186	81
2	22	19	190	88
3	30	89	180	70
4	26	79	183	77
5	26	43	180	70
6	22	0	180	68
7	18	0	173	63
8	21	10	170	63
9	26	25	184	73

Stage4 의류 사이즈 추천하기

4) 모형 적합(단순선형회귀)

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression().fit(x_train, y_train)
print('train set accuracy:', lr.score(x_train, y_train))

#오히려 출장수 제거했더니 정확도 0.55로 낮아짐
```

```
train set accuracy: 0.6550739388896691
```

> 65.51%의 설명력을 보임

5) Test data 예측해보기

5-1) Csv 파일 불러와서 예측하기

```
#test data 예측해보기
df_test = pd.read_csv('data/test_players.csv')
df_test
```

	이름	나이	출장수	키	몸무게
0	황인범	22	14	177	67
1	김신욱	31	51	196	93

```
x_test = df_test.drop(['이름'], axis=1)
prediction = lr.predict(x_test)
prediction
```

```
array([270.2212304 , 295.31336894])
```

* 보완 해야 하는 점
1. 다량의 데이터 확보
2. 사전탐색, 분석작업필요
3. 평가 기준에 대한 고민
4. 더 많은 특징을 가진 데이터 확보

5-2) 직접 값 입력해서 예측해보기

```
# 리스트 형식으로 데이터 넣어서 예측해보기
lr.predict([[15, 0, 160, 50], [22, 10, 180, 72]])
```

```
array([251.43870005, 276.08265283])
```

Challenge2 실제값과 예측값을 비교하는 DataFrame 만들기

Challenge 2

```
comparison = pd.DataFrame(y_train) #적합사이즈 집어넣기

y_test = lr.predict(x_train) #나의 예측값
comparison['나의 예측'] = y_test

comparison
```

	적합사이즈	나의 예측
0	275	281.829639
1	285	277.882881
2	290	289.474580
3	265	263.176528
4	260	268.559816
5	275	270.052410
6	280	273.917114
7	270	265.904555
8	260	267.476237
9	275	276.726240

Homework2 타이타닉 생존자 예측하기(by Linear Regression)

단순선형회귀모형(week11)

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

model = make_pipeline(PolynomialFeatures(2), LinearRegression()).fit(x_train, y_train)

print("train set score: {:.2f}".format(model.score(x_train, y_train)))
```

```
train set score: 0.39
```

성능이 그닥 좋지 않음.

의사결정나무(week10)

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier()
tree.fit(x_train, y_train)

print('training set accuracy:', tree.score(x_train, y_train))
```

```
training set accuracy: 0.8838383838383838
```

의사결정나무가 훨씬 좋은 모형성능을 보임.