08. 선형회귀분석

1. 선형회귀분석의 이해

○ 회귀분석

- 회귀분석은 연속형 출력변수 Y와 입력변수 X의 다음의 함수 관계를 모형화하는 방법임.

$$Y = f(X) + \epsilon$$

- 여기서 ϵ은 오차항으로서 기댓값이 0이고 분산이 σ^2임.
- 회귀분석에서 X가 주어졌을 때 Y의 조건부 기댓값을 특정형태의 함수로 가정하고 자료로부터 그 함수를 추정함.

-만약 f가 선형함수라고 가정한다면 이를 선형회귀분석이라고 함.

○ 선형회귀모형

- 입력변수의 수가 1개인 경우를 단순선형회귀모형, 2개 이상인 경우를 다중선형회귀모형이라고 함.
- 입력변수가 $p \geq 2$개인 다중선형회귀모형은 다음과 같이 나타낼 수 있음.

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \boldsymbol{I}\sigma^2)$$

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i$$

- 여기서 $i = 1, 2, ..., n$에 대하여 $\boldsymbol{X}$는 i번째 행이 $(1, x_{i1}, ..., x_{ip})$인 $n \times (p+1)$ 행렬이고 $y = (y_1, ..., y_n)'$는 출력변수 벡터를 나타냄.
- 또한, $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_p)$는 자료로부터 추정해야 하는 미지의 모수이고, ϵ_i는 오차항으로 평균이 0이고 분산이 σ^2임.

- 회귀 모수 추정을 위한 다중선형회귀모형의 오차제곱합은

$$SSE = (\boldsymbol{y} - \boldsymbol{X\beta})'(\boldsymbol{y} - \boldsymbol{X\beta})$$

와 같음.

- 따라서 오차제곱합을 최소화하는 최소제곱 추정값은

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$$

와 같음.

- 여기서 j번째 회귀계수의 추정값 $\hat{\beta}_j$는 다른 변수들의 값이 고정되었을 때 x_j가 한 단위 증가할 때의 y의 증가량을 의미함. 만약 $\hat{\beta}_j$가 양수일

때, x_j의 값이 증가하면 y의 값이 증가함.

- j번째 회귀계수 β_j에 대하여 회귀계수의 유의성($H_0 : \beta_j = 0$)를 검정하기위한 검정통계량은

$$\frac{\hat{\beta}_j}{S.E(\hat{\beta}_j)}$$

으로 이는 t 분포를 따름.

- 또한, 선형회귀모형이 얼마나 적합한가를 측정하는 측도가 필요함.
- 출력변수 y_i에 대응되는 예측값을 $\hat{y}_i$라 하면 입력변수값 y_i는 다음과 같이 표현할 수 있음.

$$y_i = \hat{y}_i + e_i$$

- 즉, 관측된 출력변수의 값은 선형회귀모형으로 설명되는 부분과 선형회귀모형으로 설명되지 않는 부분으로 나눌 수 있음.
- 전체적인 출력변수의 값들의 변동의 크기를 나타내는 총제곱합, 선형회귀모형에 의해 설명되는 있는 부분의 변동의 크기를 나타내는 회귀제곱합, 마지막으로 잔차들에 의한 변동의 크기인 잔차제곱합은 다음과 같이 나타낼 수 있음.

$$SST = \boldsymbol{y}'(\boldsymbol{I} - \frac{1}{n}\boldsymbol{J})\boldsymbol{y},\ SSR = \boldsymbol{y}'(\boldsymbol{H} - \frac{1}{n}\boldsymbol{J})\boldsymbol{y},\ SST = \boldsymbol{y}'(\boldsymbol{I} - \boldsymbol{H})\boldsymbol{y}$$

여기서 $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$, $\boldsymbol{J} = \mathbf{1}\mathbf{1}'$, $\mathbf{1}$은 1로만 이루어진 벡터임.

- 이 때, $SST = SSR + SSE$가 성립하므로, 만약 주어진 자료가 선형회귀모형에 적합하다고 한다면 선형회귀모형으로 설명되는 SSR부분이 SST의 대부분을 차지하게 될 것임.
- 따라서 선형회귀모형이 얼마나 적합한가는 SST 중에서 SSR이 얼마나 차지하고 있는지를 통해 판단할 수 있고, SSR/SST를 결정계수라고 함.

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

- 그리고 회귀제곱합의 자유도는 p, 잔차제곱합의 자유도는 $n-p-1$, 전체제곱합의 자유도는 $n-1$로 선형회귀모형에 의한 제곱평균과 오차에 의한 제곱평균은 각각 다음과 같음.

$$MSR = \frac{SSR}{p},\ MSE = \frac{SSE}{n-p-1}$$

- 이러한 제곱합, 자유도, 제곱평균을 표로 정리한 것을 분산분석표라고 하고 분석분석표는 다음과 같이 나타낼 수 있음.

요인	제곱합	자유도	제곱평균	F
회귀	SSR	p	$MSR = \frac{SSR}{p}$	$F = \frac{MSR}{MSE}$
오차	SSE	$n-p-1$	$MSE = \frac{SSE}{n-p-1}$	
계	SST	$n-1$		

- 또한, MSR과 MSE의 비인 $\frac{MSR}{MSE}$는 자유도가 $(p, n-p-1)$인 F분포를 따르고, 이 검정통계량을 통해 모형이 유의한지를 판단하는 $H_0 : \beta_0 = \beta_1 = ... = \beta_p = 0$을 검정함.

2. sklearn을 활용한 선형회귀분석

- 선형회귀분석을 실시하기 위해 먼저 sklearn 모듈을 활용한 선형회귀분석 방법에 대해서 살펴보도록 하겠음.

- 주어진 데이터를 100% 활용하여 선형회귀분석을 실시한 후 새롭게 데이터를 모형에 적용하였을 때 예측 성능이 떨어지는 경우가 많음.
- 즉, 모델이 주어진 데이터에 너무 과적합되도록 학습한 나머지 조금이라도 벗어난 데이터에 대해 예측률이 현저하게 떨어지게 되는데, 이를 overfitting 이라고 함.
- 이러한 현상을 해소하기 위하여 모형을 만드는 Train 데이터와 검정을 하는 Test 데이터를 일정 비율로 나누어 Test 데이터로 모형을 평가하는 데 이를 교차 검증이라고 함.
- 그럼 먼저 주어진 데이터를 Train 데이터와 Test 데이터로 분할하는 방법을 알아보도록 하겠음.
- 주어진 데이터를 Train 데이터와 Test 데이터로 분할은 sklearn.model_selection 모듈의 train_test_split() 함수를 이용하여 실시할 수 있음.

```
from sklearn.model_selection import train_test_split
train 객체명, test 객체명 = train_test_split(데이터셋, test_size=비율,
random_state=번호)
```

- 주어진 데이터셋을 Train 데이터와 Test 데이터로 분류하기 위한 train_test_split() 함수는 sklearn.model_selection 모듈에서 불러올 수 있음.
- train_test_split() 함수에 먼저 분할하고자 하는 데이터셋을 입력함. 그런 후, Test 데이터의 비율을 test_size 옵션에 입력함. 이 때 입력값은 비율값이므로 0과 1사이의 값을 입력함.
- 그리고 랜덤하게 Train 데이터셋과 Test 데이터셋을 추출함에 있어 난수번호를 지정하여 데이터셋 분류의 결과가 일정하게 나타나게 하고자 할 경우 random_state 옵션에 임의의 난수번호를 지정할 수 있음.
- train_test_split() 함수를 실행하면 Train 데이터와 Test 데이터가 차례로 반환되어 출력됨. 따라서 이 데이터를 각각 할당할 Train 데이터 객체명과 Test 데이터 객체명을 지정하여 저장함.

- 그럼 분석에 사용되는 Train 데이터를 활용하여 선형회귀분석을 실시하는 방법을 알아보도록 하겠음.
- 분석에 사용되는 Train 데이터를 활용하여 선형회귀분석 실시는 sklearn.linear_model 모듈의 LinearRegression() 함수를 이용하여 실시할 수 있음.

```
from sklearn.linear_model import LinearRegression
LinearRegression().fit(입력변수, 출력변수)
```

- LinearRegression().fit() 함수에 분석에 사용되는 Train 데이터의 입력변수와 출력변수를 입력함.
- 위 명령어를 사용하면 보통의 분석과는 다르게 아무런 결과가 출력되지 않고, 분석이 실시되었다는 의미로 LinearRegression()이라는 메시지만 출력됨.
- 이는 R과 같은 다른 프로그램들과 마찬가지로 분석이 수행된 결과를 다른 객체에 할당한 후 이를 다양한 함수에 활용하여 결과를 확인할 수 있음.

- 따라서 LinearRegression().fit() 함수를 실행시킨 결과를 객체에 할당하여 다음과 같은 함수를 활용하여 선형회귀모형의 절편항, 기울기항, 결정계수 및 모형에 피팅된 모형 예측값을 확인할 수 있음.

결정계수: 회귀모형 객체.**score**(입력변수, 출력변수)
회귀절편: 회귀모형 객체.**intercept_**
기울기: 회귀모형 객체.**coef_**
모형 예측값: 회귀모형 객체.**predict**(입력변수)

- 선형회귀모형의 설명력을 확인하기 위한 결정계수는 선형회귀모형의 결과를 할당한 객체를 활용하여 score() 함수를 사용하여 확인할 수 있음. 이 때 score() 함수에 결정계수 계산을 위한 입력변수와 출력변수를 차례로 입력하면 모형의 결정계수를 확인할 수 있음. 만약 Test 데이터의 입력변수와 출력변수를 입력하면 생성된 선형회귀모형에 Test 데이터를 적용하였을 때의 설명력을 확인하여 모형을 평가할 수 있음.
- 다음으로 선형회귀모형의 절편항과 기울기항을 확인하기 위해서는 선형회귀모형의 결과를 할당한 객체를 활용하여 intercept_ 함수와 coef_ 함수를 사용하여 확인할 수 있음.
- 그리고 선형회귀모형에 의해 예측된 출력변수의 값을 확인하기 위해서는 선형회귀모형의 결과를 할당한 객체를 활용하여 predict() 함수를 사용하여 확인할 수 있음. 이 때 predict() 함수에 출력변수의 예측값을 계산하기 위한 입력변수를 입력하면 출력변수의 예측값을 확인할 수 있음. 만약 Test 데이터의 입력변수를 입력하면 생성된 선형회귀모형에 Test 데이터를 적용하였을 때의 출력변수의 예측값을 확인할 수 있음.

- 그 외 모형 평가에 사용되는 MSE, MAE, R^2을 계산하기 위해서 sklearn.metrics 모듈의 mean_squared_error()함수, mean_absolute_erroor() 함수, r2_score() 함수를 이용하여 실시할 수 있음.

from **sklearn.metrics** import **mean_squared_error, mean_absolute_error, r2_score**
MSE: **mean_squared_error**(출력변수, 출력변수 예측값)
MAE: **mean_absolute_error**(출력변수, 출력변수 예측값)
결정계수: **r2_score**(출력변수, 출력변수 예측값)

- 선형회귀모형을 평가하기 위해 얼마나 잘 예측되었는지를 확인하기 위해 사용되는 지표들인 MSE, MAE, R^2는 mean_squared_error()함수, mean_absolute_erroor() 함수, r2_score() 함수에 각각 출력변수와 출력변수의 예측값을 차례로 입력하여 확인할 수 있음.
- 만약 Test 데이터의 출력변수와 출력변수의 예측값을 차례로 입력하면 생성된 선형회귀모형에 Test 데이터를 적용하였을 때의 평가지표들을 확인할 수 있음.

3. statsmodels를 활용한 선형회귀분석

- 앞서 살펴보았던 sklearn.linear_model 모듈을 이용하여 실시한 선형회귀분석은 모형의 선형계수 및 모형의 평가 지표등은 계산할 수 있지만, 입력변수들의 유의성 검정을 위한 p-value 등은 확인할 수 없는 단점이 존재함.
- 따라서 모형의 선형계수 및 모형의 평가지표 뿐만 아니라 입력변수들의 유의성 검정을 위한 p-value를 확인하기 위해 또 다른 선형회귀분석을 위한 statsmodels 모듈을 알아볼 필요가 있음.
- statsmodels 모듈에서 선형회귀분석을 실시하기 위한 함수로는 OLS()와 ols() 함수가 존재함.

○ OLS를 이용한 회귀분석

- 그럼 먼저 OLS() 함수를 활용하여 선형회귀분석을 실시하는 방법을 알아보도록 하겠음.
- OLS() 함수의 경우 앞서 살펴본 LinearRegression() 함수와는 다르게 입력변수를 지정하면 자동으로 절편항을 입력하지 않음. 즉, 모형식에서 절편항이 제거된 상태로 분석이 이루어지기 때문에 절편항을 분석결과가 나타나게 하기 위해 입력변수 데이터에 1로만 구성된 절편항 데이터를 추가해야 함.
- 따라서 OLS() 함수를 통해 선형회귀분석을 실시하기 전 입력변수 데이터에 절편항 데이터를 추가하기 위해 statsmodels.api 모듈의 add_constant() 함수를 이용하여 실시할 수 있음.

```
from statsmodels.api import add_constant
add_constant(입력변수)
```

- add_constant() 함수에 분석에 사용되는 Train 데이터의 입력변수를 입력하면 입력변수 데이터의 0번째 열에 모두 1로 구성된 열이 추가됨.
- 이를 새로운 입력변수 객체명에 할당하여 분석에 활용하면 절편항이 추가된 선형회귀모형식을 확인할 수 있음.

- 그럼 절편항이 추가된 입력변수와 출력변수를 활용하여 선형회귀분석을 실시하는 statsmodels.api 모듈의 OLS() 함수는 다음과 같이 이용할 수 있음.

```
from statsmodels.api import OLS
객체명 = OLS(출력변수, 입력변수).fit()
모형 결과: 객체명.summary()
모형 적합값: 객체명.predict(입력변수)
```

- OLS().fit() 함수에 분석에 사용되는 Train 데이터의 출력변수와 입력변수를 입력함. 이 때, LinearRegression() 함수와는 출력변수와 입력변수 입력 순서가 다르니 주의하기 바람.
- 위 명령어를 사용하면 LinearRegression() 함수와 마찬가지로 아무런 결과가 출력되지 않고, 분석이 실시되었다는 의미의 메시지만 출력됨.
- 따라서 LinearRegression() 함수와 마찬가지로 분석이 수행된 결과를 다른 객체에 할당한 후 이를 다양한 함수에 활용하여 결과를 확인할 수 있음.
- 선형회귀모형의 각종 결과는 선형회귀모형의 결과를 할당한 객체를 활용하여 summary() 함수를 사용하여 확인할 수 있음.
- 그리고 선형회귀모형에 의해 예측된 출력변수의 값을 확인하기 위해서는 선형회귀모형의 결과를 할당한 객체를 활용하여 predict() 함수를 사용하여 확인할 수 있음. 이 때 predict() 함수에 출력변수의 예측값을 계산하기 위한 입력변수를 입력하면 출력변수의 예측값을 확인할 수 있음. 만약 Test 데이터의 입력변수를 입력하면 생성된 선형회귀모형에 Test 데이터를 적용하였을 때의 출력변수의 예측값을 확인할 수 있음.

○ ols를 이용한 회귀분석

- 앞서 설명한 LinearRegression() 함수와 OLS() 함수는 모두 출력변수와 입력변수 데이터를 직접 입력하는 방식이었음.
- 출력변수와 입력변수를 직접 입력하는 방식이 아닌 변수이름을 활용하여 모형식을 입력하여 선형회귀분석을 실시하는 statsmodels.formula.api 모듈의 ols() 함수는 다음과 같이 이용할 수 있음.

```
from statsmodels.formula.api import ols
객체명 = ols(모형식, data=데이터셋).fit()
모형 결과: 객체명.summary()
모형 적합값: 객체명.predict(입력변수)
```

- ols().fit() 함수에 분석에 사용되는 모형식을 따옴표 안에 먼저 입력함. 이 때, 모형식 입력방법은 출력변수와 입력변수는 ~ 기호를 이용하여 구분함. 그리고 입력변수들은 + 기호를 이용하여 구분함. 즉, '출력변수 ~ 입력변수1 + 입력변수2 + ... ' 형태로 입력함.
- 그리고 data 옵션에 분석에 사용되는 데이터셋을 입력함. 이 때, 앞서 모형식에 사용되었던 변수명이 분석에 사용되는 데이터셋에 반드시 포함되어 있어야 함.
- ols() 함수도 앞선 명령어들과 마찬가지로 아무런 결과가 출력되지 않고, 분석이 실시되었다는 의미의 메시지만 출력됨.
- 따라서 분석이 수행된 결과를 다른 객체에 할당한 후 이를 다양한 함수에 활용하여 결과를 확인할 수 있음.
- 선형회귀모형의 각종 결과는 선형회귀모형의 결과를 할당한 객체를 활용하여 summary() 함수를 사용하여 확인할 수 있음.
- 그리고 선형회귀모형에 의해 예측된 출력변수의 값을 확인하기 위해서는 선형회귀모형의 결과를 할당한 객체를 활용하여 predict() 함수를 사용하여 확인할 수 있음. 이 때 predict() 함수에 출력변수의 예측값을 계산하기 위한 입력변수를 입력하면 출력변수의 예측값을 확인할 수 있음. 만약 Test 데이터의 입력변수를 입력하면 생성된 선형회귀모형에 Test 데이터를 적용하였을 때의 출력변수의 예측값을 확인할 수 있음.

- seaborn 모듈의 mpg 데이터에서, acceleration을 예측하기 위해 displacement, horsepower, weight를 사용한 선형회귀분석을 실시하

고자 함.

- info() 함수를 활용하여 자료를 확인한 결과 horsepower에서 6개의 결측값이 존재하여 dropna() 함수를 이용하여 결측값을 제거하였음.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('mpg')

In [3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64
 7   origin        398 non-null    object
 8   name          398 non-null    object
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB

In [4]: df = df.dropna(subset=['horsepower'], how='any', axis=0)
```

- 다음으로 모형을 생성하기 위한 Train 데이터와 검증을 위한 Test 데이터를 구분하기 위해 train_test_split() 함수를 사용하였음.

```
In [5]: from sklearn.model_selection import train_test_split

In [6]: train, test = train_test_split(df, test_size=0.3, random_state=1)

In [7]: X_train=train[['displacement', 'horsepower', 'weight']]
   ...: y_train=train['acceleration']
   ...: X_test=test[['displacement', 'horsepower', 'weight']]
   ...: y_test=test['acceleration']
```

- 이 때 Train 데이터는 train 객체에 할당하고, Test 데이터는 test 객체에 할당하였음. 그런 후 X_train과 X_test 객체에 train 객체와 test 객체의 입력변수를 선택하여 각각 할당하였고, y_train과 y_test 객체에 train 객체와 test 객체의 출력변수를 선택하여 각각 할당하였음.

- Train 데이터의 입력변수 자료인 X_train과 Train 데이터의 출력변수 자료인 y_train을 각각 LinearRegression().fit() 함수에 대입하여 선형회귀분석을 실시하였음.

```
In [8]: from sklearn.linear_model import LinearRegression

In [9]: lr = LinearRegression().fit(X_train, y_train)
```

- LinearRegression().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이

를 lr이라는 객체에 할당하였고, 이를 활용하여 결정계수, 절편, 기울기, 출력변수의 예측값을 각각 계산하였음.

```
In [10]: r_square = lr.score(X_train, y_train)

In [11]: print('R^2: ', r_square)
R^2:  0.6228883243680499

In [12]: print('y절편: ', lr.intercept_)
y절편:  17.194626087262485

In [13]: print('기울기: ', lr.coef_)
기울기:  [-0.00963822 -0.09122627  0.00328734]

In [14]: y_train_pred = lr.predict(X_train)
```

- score() 함수를 이용하여 결정계수를 계산한 결과 0.62288로 나타났고, intercept_ 함수를 이용하여 절편을 계산한 결과 17.1946으로 나타났음. 그리고 coef_ 함수를 이용하여 입력변수인 displacement, horsepower, weight들의 기울기를 계산한 결과 각각 -0.0096, -0.0912, 0.0032로 나타났음.

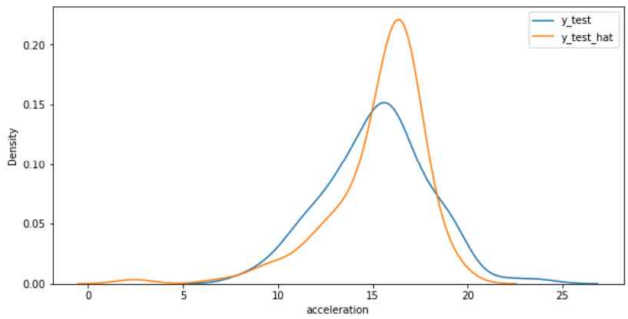
- predict() 함수에 Train 데이터의 입력변수 데이터인 X_train을 대입하여 Train 데이터의 출력변수 예측값을 y_train_pred 객체에 할당하여 선형회귀모형을 평가하는 지표인 MSE, MAE 등을 계산하였음.

```
In [15]: from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

In [16]: mean_squared_error(y_train, y_train_pred)
Out[16]: 2.8722873823218

In [17]: mean_absolute_error(y_train, y_train_pred)
Out[17]: 1.3112447619938132

In [18]: r2_score(y_train, y_train_pred)
Out[18]: 0.6228883243680499
```

- 그 결과 모형 생성에 사용된 Train 데이터의 MSE는 2.8722, MAE는 1.3112, 앞서 score() 함수를 이용해서 확인한 결과였던 결정계수는 0.6228로 나타났음.

- 그리고 Train 데이터의 출력변수의 값과 예측값을 seaborn 모듈의 kdeplot() 함수를 이용한 커널밀도함수를 통해 그림을 그려본 결과를 통해 얼마나 출력변수의 값과 예측값이 일치하는지도 평가할 수 있음.

```
In [19]: import matplotlib.pyplot as plt
    ...: import seaborn as sns

In [20]: plt.figure(figsize=(10, 5))
    ...: sns.kdeplot(y_train)
    ...: sns.kdeplot(y_train_pred)
    ...: plt.legend(labels=['y_train','y_train_hat'], loc='best')
Out[20]: <matplotlib.legend.Legend at 0x180c8b59408>
```



- predict() 함수에 Test 데이터의 입력변수 데이터인 X_test를 대입하여 Test 데이터의 출력변수 예측값을 y_test_pred 객체에 할당하여 선형회귀모형을 평가하는 지표인 MSE, MAE 등을 계산하였음.

```
In [21]: y_test_pred = lr.predict(X_test)

In [22]: from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

In [23]: mean_squared_error(y_test, y_test_pred)
Out[23]: 3.0940803551294307

In [24]: mean_absolute_error(y_test, y_test_pred)
Out[24]: 1.2886178072796097

In [25]: r2_score(y_test, y_test_pred)
Out[25]: 0.5791980313385208
```

- 그 결과 모형 검증에 사용된 Test 데이터의 MSE는 3.0940, MAE는 1.2886, 결정계수는 0.5791로 나타났음. 이는 Train 데이터의 결과와 큰 차이가 없는 것을 확인할 수 있음.

- 다음으로 Train 데이터의 입력변수 자료인 X_train과 Train 데이터의 출력변수 자료인 y_train을 각각 OLS() 함수에 대입하여 선형회귀분석을 실시하고자 함.
- OLS() 함수는 앞서 설명하였듯이 절편항을 기본적으로 제공하지 않기 때문에 상수항을 추가한 입력변수 데이터를 생성해야 함. 따라서 이를 위해 statsmodels.api 모듈의 add_constant() 함수를 사용하여 입력변수 데이터에 절편항을 추가하였음.

```
In [35]: from statsmodels.api import add_constant

In [36]: X_train_ad = add_constant(X_train)

In [37]: print(X_train_ad)
     const  displacement  horsepower  weight
335    1.0         122.0        88.0    2500
397    1.0         119.0        82.0    2720
394    1.0          97.0        52.0    2130
29     1.0          97.0        88.0    2130
275    1.0         163.0       125.0    3140
..     ...           ...         ...     ...
205    1.0          97.0        75.0    2155
257    1.0         232.0        90.0    3210
73     1.0         307.0       130.0    4098
237    1.0          98.0        63.0    2051
38     1.0         350.0       165.0    4209

[274 rows x 4 columns]
```

- 그 결과 입력변수 데이터 0번째 열에 const라는 1로만 구성된 변수가 추가된 것을 확인할 수 있음.

- 절편항이 추가된 입력변수 데이터와 출력변수 데이터를 활용하여 statsmodels.api 모듈의 OLS().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 result라는 객체에 할당하였고, 이를 summary() 함수를 적용하여 선형회귀분석 결과를 확인하였음.

```
In [38]: from statsmodels.api import OLS

In [39]: result = OLS(y_train, X_train_ad).fit()

In [40]: result.summary()
Out[40]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           acceleration   R-squared:                       0.623
Model:                            OLS   Adj. R-squared:                  0.619
Method:                 Least Squares   F-statistic:                     148.7
Date:                Fri, 27 Aug 2021   Prob (F-statistic):           6.93e-57
Time:                        17:15:50   Log-Likelihood:                -533.34
No. Observations:                 274   AIC:                             1075.
Df Residuals:                     270   BIC:                             1089.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         17.1946      0.581     29.596      0.000      16.051      18.338
displacement  -0.0096      0.003     -3.044      0.003      -0.016      -0.003
horsepower    -0.0912      0.006    -14.969      0.000      -0.103      -0.079
weight         0.0033      0.000      8.984      0.000       0.003       0.004
==============================================================================
Omnibus:                       38.552   Durbin-Watson:                   1.857
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               52.410
Skew:                           0.924   Prob(JB):                     4.16e-12
Kurtosis:                       4.084   Cond. No.                     1.74e+04
==============================================================================
```

- 그 결과 LinearRegression() 함수의 결과와 같은 절편항, 기울기항, 결정계수 등등을 확인할 수 있음. 거기에 추가하여 조정된 결정계수, AIC, BIC와 같은 통계량도 확인할 수 있음.
- 그리고 특히 LinearRegression() 함수에서 확인할 수 없었던 계수들의 유의성을 확인할 수 있음. displacement, horsepower, weight 변수들의 p-value는 각각 0.003, 0.000, 0.000으로 나타나 모두 출력변수에 유의한 영향을 끼친다는 것을 확인할 수 있음.

- 다음으로 Train 데이터인 train 데이터를 ols() 함수에 대입하여 선형회귀분석을 실시하고자 함.
- train 데이터를 입력변수 데이터셋과 출력변수 데이터셋으로 구분하지 않고 모형식을 활용하여 statsmodels.formula.api 모듈의 ols().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 result라는 객체에 할당하였고, 이를 summary() 함수를 적용하여 선형회귀분석 결과를 확인하였음.

```
In [49]: from statsmodels.formula.api import ols

In [50]: formula = 'acceleration ~ displacement + horsepower + weight'

In [51]: result = ols(formula = formula, data = train).fit()

In [52]: result.summary()
Out[52]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           acceleration   R-squared:                       0.623
Model:                            OLS   Adj. R-squared:                  0.619
Method:                 Least Squares   F-statistic:                     148.7
Date:                Fri, 27 Aug 2021   Prob (F-statistic):           6.93e-57
Time:                        17:16:32   Log-Likelihood:                -533.34
No. Observations:                 274   AIC:                             1075.
Df Residuals:                     270   BIC:                             1089.
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       17.1946      0.581     29.596      0.000      16.051      18.338
displacement    -0.0096      0.003     -3.044      0.003      -0.016      -0.003
horsepower      -0.0912      0.006    -14.969      0.000      -0.103      -0.079
weight           0.0033      0.000      8.984      0.000       0.003       0.004
==============================================================================
Omnibus:                       38.552   Durbin-Watson:                   1.857
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               52.410
Skew:                           0.924   Prob(JB):                     4.16e-12
Kurtosis:                       4.084   Cond. No.                     1.74e+04
==============================================================================
```

- 그 결과 앞서 실시한 OLS() 함수와 같은 형태의 결과를 얻을 수 있음.

4. 선형회귀분석의 실습

○ seaborn 모듈의 penguins 데이터에서, body_mass_g을 예측하기 위해 bill_length_mm와 bill_depth_mm를 사용한 선형회귀분석을 실시하고자 함.

- info() 함수를 활용하여 자료를 확인한 결과 body_mass_g, bill_length_mm, bill_depth_mm에서 2개의 결측값이 존재하여 dropna() 함수를 이용하여 결측값을 제거하였음.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('penguins')

In [3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB

In [4]: df = df.dropna(subset=['bill_length_mm', 'bill_depth_mm', 'body_mass_g'], how='any', axis=0)
```

- 다음으로 모형을 생성하기 위한 Train 데이터와 검증을 위한 Test 데이터를 구분하기 위해 train_test_split() 함수를 사용하였음.

```
In [5]: from sklearn.model_selection import train_test_split

In [6]: train, test = train_test_split(df, test_size=0.3, random_state=1)

In [7]: X_train=train[['bill_length_mm', 'bill_depth_mm']]
   ...: y_train=train['body_mass_g']
   ...: X_test=test[['bill_length_mm', 'bill_depth_mm']]
   ...: y_test=test['body_mass_g']
```

- 이 때 Train 데이터는 train 객체에 할당하고, Test 데이터는 test 객체에 할당하였음. 그런 후 X_train과 X_test 객체에 train 객체와 test 객체의 입력변수를 선택하여 각각 할당하였고, y_train과 y_test 객체에 train 객체와 test 객체의 출력변수를 선택하여 각각 할당하였음.

- Train 데이터의 입력변수 자료인 X_train과 Train 데이터의 출력변수 자료인 y_train을 각각 LinearRegression().fit() 함수에 대입하여 선형회귀분석을 실시하였음.

```
In [8]: from sklearn.linear_model import LinearRegression

In [9]: lr = LinearRegression().fit(X_train, y_train)
```

- LinearRegression().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 lr이라는 객체에 할당하였고, 이를 활용하여 결정계수, 절편, 기울기, 출력변수의 예측값을 각각 계산하였음.

```
In [10]: r_square = lr.score(X_train, y_train)

In [11]: print('R^2: ', r_square)
R^2:  0.4693159195080788

In [12]: print('y절편: ', lr.intercept_)
y절편:  3315.418745221548

In [13]: print('기울기: ', lr.coef_)
기울기:  [  72.79158574 -135.63472974]

In [14]: y_train_pred = lr.predict(X_train)
```

- score() 함수를 이용하여 결정계수를 계산한 결과 0.46931로 나타났고, intercept_ 함수를 이용하여 절편을 계산한 결과 3315.4187로 나타났음. 그리고 coef_ 함수를 이용하여 입력변수인 bill_length_mm와 bill_depth_mm들의 기울기를 계산한 결과 각각 72.7915, -135.6347로 나타났음.

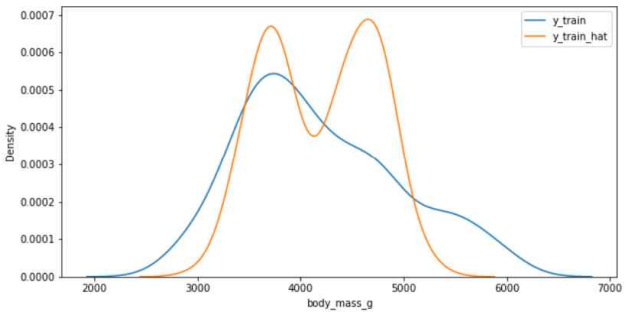
- predict() 함수에 Train 데이터의 입력변수 데이터인 X_train을 대입하여 Train 데이터의 출력변수 예측값을 y_train_pred 객체에 할당하여 선형회귀모형을 평가하는 지표인 MSE, MAE 등을 계산하였음.

```
In [15]: from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

In [16]: mean_squared_error(y_train, y_train_pred)
Out[16]: 313519.1726750238

In [17]: mean_absolute_error(y_train, y_train_pred)
Out[17]: 454.76011830778106

In [18]: r2_score(y_train, y_train_pred)
Out[18]: 0.4693159195080788
```

- 그 결과 모형 생성에 사용된 Train 데이터의 MSE는 313519.1726, MAE는 454.76011, 앞서 score() 함수를 이용해서 확인한 결과였던 결정계수는 0.46931로 나타났음.

- 그리고 Train 데이터의 출력변수의 값과 예측값을 seaborn 모듈의 kdeplot() 함수를 이용한 커널밀도함수를 통해 그림을 그려본 결과를 통해 얼마나 출력변수의 값과 예측값이 일치하는지도 평가할 수 있음.

```
In [19]: import matplotlib.pyplot as plt
    ...: import seaborn as sns

In [20]: plt.figure(figsize=(10, 5))
    ...: sns.kdeplot(y_train)
    ...: sns.kdeplot(y_train_pred)
    ...: plt.legend(labels=['y_train','y_train_hat'], loc='best')
Out[20]: <matplotlib.legend.Legend at 0x180c8b59408>
```



- predict() 함수에 Test 데이터의 입력변수 데이터인 X_test를 대입하여 Test 데이터의 출력변수 예측값을 y_test_pred 객체에 할당하여 선형회귀모형을 평가하는 지표인 MSE, MAE 등을 계산하였음.

```
In [21]: y_test_pred = lr.predict(X_test)

In [22]: from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

In [23]: mean_squared_error(y_test, y_test_pred)
Out[23]: 401534.3858332877

In [24]: mean_absolute_error(y_test, y_test_pred)
Out[24]: 530.8484192226334

In [25]: r2_score(y_test, y_test_pred)
Out[25]: 0.46955642576593093
```

- 그 결과 모형 검증에 사용된 Test 데이터의 MSE는 401534.3858, MAE는 530.8484, 결정계수는 0.46955로 나타났음. 이는 출력변수의 단위에 영향을 받아 큰 차이가 나는 것처럼 보이지만 결정계수값을 통해 살펴보면 Train 데이터의 결과와 큰 차이가 없는 것을 확인할 수 있음.

- 다음으로 Train 데이터의 입력변수 자료인 X_train과 Train 데이터의 출력변수 자료인 y_train을 각각 OLS() 함수에 대입하여 선형회귀분석을 실시하고자 함.
- OLS() 함수는 앞서 설명하였듯이 절편항을 기본적으로 제공하지 않기 때문에 상수항을 추가한 입력변수 데이터를 생성해야 함. 따라서 이를 위해 statsmodels.api 모듈의 add_constant() 함수를 사용하여 입력변수 데이터에 절편항을 추가하였음.

```
In [26]: from statsmodels.api import add_constant

In [27]: X_train_ad = add_constant(X_train)

In [28]: print(X_train_ad)
     const  bill_length_mm  bill_depth_mm
232    1.0            45.5           13.7
274    1.0            46.5           14.4
324    1.0            47.3           13.8
18     1.0            34.4           18.4
257    1.0            44.4           17.3
..     ...             ...            ...
204    1.0            45.7           17.3
256    1.0            42.6           13.7
73     1.0            45.8           18.9
236    1.0            42.0           13.5
38     1.0            37.6           19.3

[239 rows x 3 columns]
```

- 그 결과 입력변수 데이터 0번째 열에 const라는 1로만 구성된 변수가 추가된 것을 확인할 수 있음.

- 절편항이 추가된 입력변수 데이터와 출력변수 데이터를 활용하여 statsmodels.api 모듈의 OLS().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 result라는 객체에 할당하였고, 이를 summary() 함수를 적용하여 선형회귀분석 결과를 확인하였음.

```
In [29]: from statsmodels.api import OLS

In [30]: result = OLS(y_train, X_train_ad).fit()

In [31]: result.summary()
Out[31]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            body_mass_g   R-squared:                       0.469
Model:                            OLS   Adj. R-squared:                  0.465
Method:                 Least Squares   F-statistic:                     104.4
Date:                Fri, 27 Aug 2021   Prob (F-statistic):           3.39e-33
Time:                        17:47:37   Log-Likelihood:                -1851.5
No. Observations:                 239   AIC:                             3709.
Df Residuals:                     236   BIC:                             3719.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3315.4187    480.805      6.896      0.000    2368.201    4262.637
bill_length_mm    72.7916      6.790     10.721      0.000      59.416      86.168
bill_depth_mm   -135.6347     18.559     -7.308      0.000    -172.198     -99.072
==============================================================================
Omnibus:                        1.120   Durbin-Watson:                   1.785
Prob(Omnibus):                  0.571   Jarque-Bera (JB):                1.228
Skew:                           0.141   Prob(JB):                        0.541
Kurtosis:                       2.791   Cond. No.                         627.
==============================================================================
```

- 그 결과 LinearRegression() 함수의 결과와 같은 절편항, 기울기항, 결정계수 등등을 확인할 수 있음. 거기에 추가하여 조정된 결정계수, AIC, BIC와 같은 통계량도 확인할 수 있음.
- 그리고 특히 LinearRegression() 함수에서 확인할 수 없었던 계수들의 유의성을 확인할 수 있음. bill_length_mm, bill_depth_mm 변수들의 p-value는 각각 0.000, 0.000으로 나타나 모두 출력변수에 유의한 영향을 끼친다는 것을 확인할 수 있음.

- 다음으로 Train 데이터인 train 데이터를 ols() 함수에 대입하여 선형회귀분석을 실시하고자 함.
- train 데이터를 입력변수 데이터셋과 출력변수 데이터셋으로 구분하지 않고 모형식을 활용하여 statsmodels.formula.api 모듈의 ols().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 result라는 객체에 할당하였고, 이를 summary() 함수를 적용하여 선형회귀분석 결과를 확인하였음.

```
In [32]: from statsmodels.formula.api import ols

In [33]: formula = 'body_mass_g ~ bill_length_mm + bill_depth_mm'

In [34]: result = ols(formula = formula, data = df).fit()

In [35]: result.summary()
Out[35]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            body_mass_g   R-squared:                       0.471
Model:                            OLS   Adj. R-squared:                  0.468
Method:                 Least Squares   F-statistic:                     150.8
Date:                Fri, 27 Aug 2021   Prob (F-statistic):           1.40e-47
Time:                        17:47:48   Log-Likelihood:                -2662.9
No. Observations:                 342   AIC:                             5332.
Df Residuals:                     339   BIC:                             5343.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      3343.1359    429.912      7.776      0.000    2497.504    4188.768
bill_length_mm   75.2808      5.971     12.608      0.000      63.537      87.025
bill_depth_mm  -142.7226     16.507     -8.646      0.000    -175.191    -110.254
==============================================================================
Omnibus:                        2.548   Durbin-Watson:                   1.806
Prob(Omnibus):                  0.280   Jarque-Bera (JB):                2.019
Skew:                           0.001   Prob(JB):                        0.364
Kurtosis:                       2.624   Cond. No.                         645.
==============================================================================
```

- 그 결과 앞서 실시한 OLS() 함수와 같은 형태의 결과를 얻을 수 있음.