09. 선형회귀분석의 가정

1. 선형회귀분석 가정의 이해

○ 선형회귀모형

- 지난 시간 학습한 선형회귀모형은 다음과 같이 표현됨.

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

그리고 여기서 최소제곱법에 의한 회귀계수 추정량은 다음과 같음.

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

- $E(y)$의 추정벡터는 $\hat{y} = X\hat{\beta}$이므로 잔차벡터는

$$e = y - \hat{y} = y - X(X'X)^{-1}X'y = [I - X(X'X)^{-1}X']y$$

이고, 이 잔차벡터의 평균과 분산은

$$E(e) = 0, \quad Var(e) = [I - X(X'X)^{-1}X']\sigma^2$$

임.

- 따라서 선형회귀모형에서 오차항에 대한 가정은

1) $E(\epsilon_i) = 0$

2) $Var(\epsilon_i) = \sigma^2$ (등분산성)

3) ϵ_i는 서로 입력이다. (입력성)

4) ϵ_i는 모든 i에 대해 정규분포를 따름. (정규성)

- 이 때 선형회귀모형에서 가정한 오차의 정규성, 입력성, 등분산성이 성립되어야 최우추정량이 되며 분산분석에서의 F 검증이 가능함.
- 이러한 오차의 가정에 대한 검토는 잔차라고 하는 y와 $\hat{y}$간의 차이를 통해서 확인할 수 있음.

○ 모형의 선형성

- 선형회귀함수가 적합한지 적합하지 않은지에 대한 문제는 자료의 산점도를 그려보거나 선형회귀모형을 적합한 후에 잔차를 각 설명변수 또는 반응변수에 대해 그려봄으로써 추측할 수 있음.
- 회귀직선의 모형이 타당하고 오차의 등분산성이 성립된다면 설명변수에 대한 잔차 산점도 또는 반응변수에 대한 잔차 산점도에서 잔차는 0을

중심으로 랜덤하게 나타나게 됨.

- 예를 들어, 산점도에서 이차곡선이나 삼차곡선의 형태가 나타난다면 가정된 선형회귀함수는 적절하지 못하다고 볼 수 있음.

○ 오차항의 정규성

- 오차항의 확률분포가 정규분포에서 많이 벗어나는 경우는 정규성을 가정한 가설검정 등의 추론을 할 수 없음.
- 정규성 가정하에서 지난시간 학습했던 부분이 모두 유효하므로 오차항에 대한 정규성 검토는 반드시 필요함.
- 오차항의 정규성을 그래프적으로 점검하는 방법으로 잔차들에 대해 분위수대분위수 그림(Q-Q plot)을 그려볼 수 있음.
- Q-Q plot을 그리는 과정은 다음과 같음.
  1) n개의 자료를 작은 것부터 크기순으로 나열함.
  2) 각 자료에 해당하는 정규점수를 계산함.
  3) i번째 순서의 자료와 i번째 순서의 정규점수를 하나의 쌍으로 2차원 공간상에 나타냄.
- Q-Q plot의 형태가 45도 기울기의 직선 근처에 있으면 정규성을 크게 벗어나지 않는다고 판단함.
- 또한, Shapiro-Wilks 검정과 같은 검정통계량을 통해서도 오차항의 정규성을 확인할 수 있음.

○ 오차항의 입력성

- 오차항의 입력성을 만족하지 않으면 앞 시점의 오차가 뒷 시점의 오차에 영향을 줌으로 입력변수가 출력변수에 미치는 영향을 제대로 파악할 수 없음 따라서 회귀모형에서 오차항의 입력성은 매우 중요한 가정임.
- 오차항의 입력성을 확인하기 위해서 연속적인 잔차의 쌍(e_{i-1},e_i)들의 산점도를 그려서 특별한 경향이 나타나지 않으면 입력이라고 판단함.
- 또한, Durbin-Watson 검정과 같은 검정통계량을 통해서도 오차항의 입력성을 확인할 수 있음.

○ 오차항의 등분산성

- 선형회귀분석에 의해 예측된 값이 어떻든지, 모든 값들에 대하여 잔차의 분산은 동일하다고 가정하고 있음.
- 오차항의 등분산성도 모형의 선형성과 마찬가지로 예측값과 잔차의 경향

선을 그렸을 때 그 형태가 수평선에 가까운, 특별한 경향을 가지지 않을 때 오차항의 등분산성을 만족한다고 판단함.

2. 선형회귀분석의 가정 확인 프로그래밍

○ 모형의 선형성

- 모형의 선형성은 선형모형에 의해 예측된 결과값과 잔차 비교를 통해서 확인할 수 있음.
- 즉, 예측값을 그래프의 x 축에, 잔차를 그래프의 y축에 플롯하였을 때, 그 경향선이 0의 값을 중심으로 직선의 형태로 나타나야 함.
- 만약 예측값과 잔차의 경향선이 0의 값을 중심으로 크게 벗어나 있다면 모형은 선형성이 없다고 판단함.

- 선형회귀분석의 가정 중 모형의 선형성을 확인하기 위해서는 산점도의 경향성을 파악하는 seaborn 모듈의 regplot() 함수를 이용하여 확인할 수 있음.

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.regplot(예측값, 잔차, line_kws = {'color':'red'})
plt.plot([예측값 최소값, 예측값 최대값], [0,0], '--', color='grey')
```

- 모형의 선형성 확인을 위한 그래프를 그리는 절차는 다음과 같음.
- seaborn 모듈의 regplot() 함수는 산점도의 경향선을 나타내는 그래픽 함수로 그래프의 x축의 값과 y축의 값을 차례로 입력함. 이 때, 모형의 선형성 확인을 위해 x축의 값에는 예측값, 그리고 y축에는 잔차를 대입함. 그리고 line_kws 옵션을 활용하여 경향선의 색을 지정할 수 있음.
- 또한, 예측값과 잔차의 경향선에 대해서 모형의 선형성을 판단하기 위한 보조선을 matplotlib.pyplot 모듈의 plot() 함수를 이용하여 그림.
- plot() 함수의 x축의 값에 회귀분석에 의해 예측된 최소값과 최대값을 리스트로 묶고, y축의 값에 [0, 0]의 값을 차례로 입력하여 보조선을 그릴 수 있음. 그 외 선의 종류, 선의 색 등을 지정할 수 있음.

○ 잔차의 정규성

- 선형회귀분석에 의해 나타난 잔차는 정규분포를 따른다는 가정을 확인하기 위해 정규성확인을 위한 그래픽 방법으로 Q-Q plot을 사용함.
- Q-Q plot은 검정하고자 하는 분포의 분위수의 값을 이용하여 확인하는 방법으로 산점도가 직선의 형태로 나타날 때 정규분포를 따른다고 판단함.

- scipy.stats 모듈의 zscore() 함수와 probplot() 함수를 이용하여 선형회귀분석의 가정 중 잔차의 정규성을 확인할 수 있음.

```
from scipy.stats import zscore, probplot
import matplotlib.pyplot as plt
import seaborn as sns
객체명 = zscore(잔차); (x, y), _ = probplot(객체명)
sns.scatterplot(x, y)
plt.plot([-3, 3], [-3, 3], '--', color='grey')
```

- 정규성 확인을 위한 Q-Q plot을 그리는 절차는 다음과 같음.
- scipy.stats 모듈의 zscore() 함수를 이용하여 표준화 잔차를 계산함. 그런 후, scipy.stats 모듈의 probplot() 함수를 이용하여 표준화 잔차값의 분위수를 계산함.
- probplot() 함수를 통해 분위수 값, 표준화 잔차값, 그 외의 값을 차례로 출력하므로 Q-Q plot을 그리기 위한 값인 분위수 값과 표준화 잔차값만 객체명을 각각 지정하여 할당함.
- 그런 후 seaborn 모듈의 scatterplot() 함수에 분위수 값과 표준화 잔차값을 대입하여 산점도를 그려 Q-Q plot을 그림.
- 또한, Q-Q plot을 통한 정규성을 판단하기 위한 보조선을 matplotlib.pyplot 모듈의 plot() 함수를 이용하여 그림.

○ 잔차의 등분산성

- 선형회귀분석에 의해 예측된 값이 어떻든지, 모든 값들에 대하여 잔차의 분산은 동일하다는 가정을 확인하기 위해 예측값에 따라서 잔차가 어떻게 달라지는지를 통하여 확인함.
- 예측값과 잔차의 경향선이 수평선에 가까울수록 잔차는 등분산성을 만족한다고 판단함.

- seaborn 모듈의 regplot() 함수를 이용하여 선형회귀분석의 가정 중 잔차의 등분산성을 확인할 수 있음.

```
from scipy.stats import zscore, probplot
import seaborn as sns
from numpy import pqrt, abs
sns.regplot(예측값, sqrt(abs(zscore(잔차)), line_kws = {'color':'red'})
```

- 잔차의 등분산성 확인을 위한 그래프 그리는 절차는 다음과 같음.
- 산점도의 경향선을 나타내는 그래픽 함수인 seaborn 모듈의 regplot() 함수의 x축에 예측값, y축에 표준화 잔차의 절대값의 제곱근 값을 입력하여 잔차의 등분산성을 확인하는 그래프를 그림.

○ 잔차의 입력성
- 회귀모형을 적합시킨 결과에서 Durbin-Watson값으로 확인가능함. Durbin-Watson값 확인은 statsmodels.formula.api 모듈의 ols().fit() 함수를 통해 나타난 결과에서 확인할 수 있음.

○ 상관분석
- 선형회귀모형에 사용되었던 변수들 간의 선형성을 판단하기 위해 상관분석을 종종 실시함.
- 상관분석은 상관계수를 통해 두 변수간의 상관관계가 존재하는지를 판단하는 것으로 피어슨의 상관 분석와 스피어만의 순위 상관 분석 등이 사용됨.
- 상관 분석과 히트맵을 이용하여 상관행렬을 시각적으로 알아보기 쉽게 나타내는 방법은 다음과 같음.

```
from scipy.stats import pearsonr, spearmanr
pearsonr([변수명1, 변수명2])
spearmanr([변수명1, 변수명2])

import seaborn as sns
sns.heatmap(데이터프레임.corr(method="pearson"), annot = True/False, cmap = '색상', vmin = -1, vmax = 1)
```

- 상관관계를 확인하는 corr() 함수는 상관행렬만 출력하고, 가설검정을 위한 p-value를 제공하지 않음. 이를 확인하기 위해 scipy.stats 모듈의 pearsonr(), spearmanr() 함수를 이용할 수 있음. 여기서 pearsonr() 함수는 피어슨 상관계수를 확인하고 이에 따른 p-value를, spearmanr() 함수는 스파이만의 순위상관계수를 확인하고 이에 따른 p-value를 확인할 수 있음.
- 또한, seaborn 모듈의 heatmap() 함수를 활용하여 상관행렬을 시각적으로 확인할 수 있음.
- heatmap() 함수에 먼저 그래프로 나타내고자 하는 상관행렬을 입력함. 그리고 annot 옵션에 True 입력을 통해 상관계수 값을 나타낼 수도, False 입력을 통해 상관계수값을 나타내지 않을수도 있음. 그리고 cmap 옵션에 그래프의 색상을 지정하고, 그 색상의 가장 명도가 약한 색을 나타내는 값을 vmin, 가장 명도가 진한 색을 나타내는 값을 vmax에 입력함.

- seaborn 모듈의 mpg 데이터에서, acceleration을 예측하기 위해 displacement, horsepower, weight를 사용한 회귀모형의 잔차를 분석하고자 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)
```

- 모형식으로 'acceleration ~ displacement + horsepower + weight'을 statsmodels.formula.api 모듈의 ols().fit()에 대입하여 결과를 확인할

수 있음.

- 그 결과 Durbin-Watson 값이 1.687로 나타나 잔차의 입력성을 확인할 수 있음.

```
In [5]: from statsmodels.formula.api import ols

In [6]: formula = 'acceleration ~ displacement + horsepower + weight'
   ...: result = ols(formula = formula, data = df).fit()

In [7]: result.summary()
Out[7]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           acceleration   R-squared:                       0.616
Model:                            OLS   Adj. R-squared:                  0.613
Method:                 Least Squares   F-statistic:                     207.4
Date:                Thu, 26 Aug 2021   Prob (F-statistic):           3.04e-80
Time:                        16:34:31   Log-Likelihood:                -766.01
No. Observations:                 392   AIC:                             1540.
Df Residuals:                     388   BIC:                             1556.
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       17.0729      0.484     35.276      0.000      16.121      18.024
displacement    -0.0100      0.003     -3.763      0.000      -0.015      -0.005
horsepower      -0.0835      0.005    -16.108      0.000      -0.094      -0.073
weight           0.0031      0.000     10.652      0.000       0.003       0.004
==============================================================================
Omnibus:                       48.170   Durbin-Watson:                   1.687
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               69.724
Skew:                           0.819   Prob(JB):                     7.24e-16
Kurtosis:                       4.259   Cond. No.                     1.73e+04
==============================================================================
```

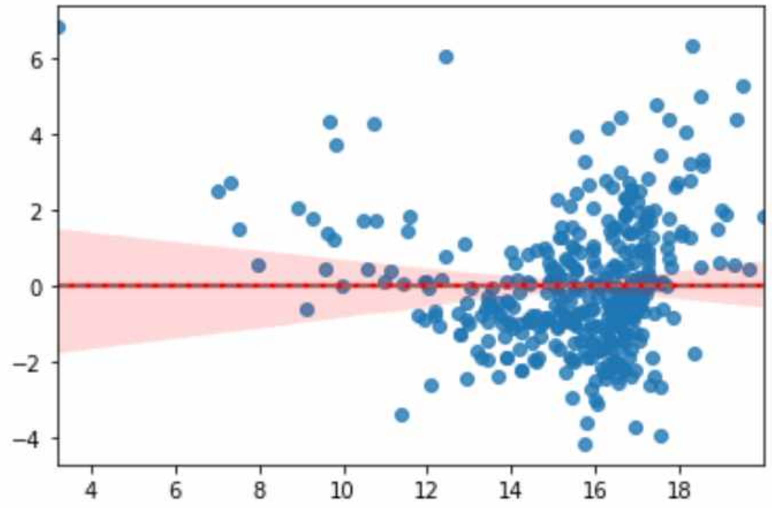
- predict() 함수에 displacement, horsepower, weight 자료를 대입하여 회귀모형에 적용하였을 때 예측값을 확인할 수 있음.
- 이 값을 출력변수인 acceleration 값과의 차이를 통해 잔차값인 residual을 저장할 수 있음.

```
In [8]: import matplotlib.pyplot as plt
   ...: import seaborn as sns

In [9]: mpg_pred = result.predict(df[['displacement', 'horsepower', 'weight']])
   ...: residual = df['acceleration'] - mpg_pred

In [10]: print(residual.head())
0   -1.896614
1    0.377654
2   -0.906892
3   -0.038026
4   -2.442524
dtype: float64
```

- 선형모형에 의한 예측값과 잔차 값을 regplot() 함수에 대입하여 모형의 선형성을 확인하는 그래프를 확인할 수 있음.
- 그 결과 예측값과 잔차값의 산점도 경향선이 0을 중심으로 수평선의 형태로 나타나는 것을 확인할 수 있음. 따라서 모형의 선형성 가정을 만족

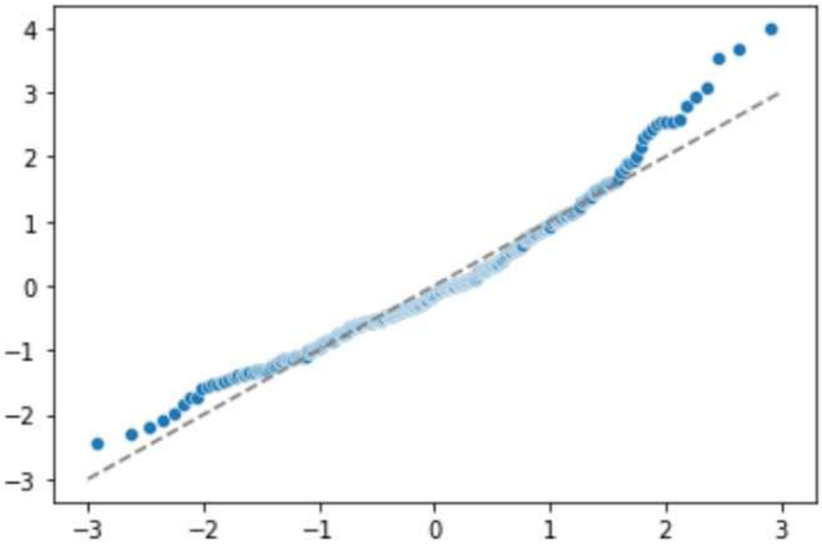
하는 것을 알 수 있음.

```
In [11]: sns.regplot(mpg_pred, residual, line_kws={'color':'red'})
    ...: plt.plot([mpg_pred.min(),mpg_pred.max()],[0,0], '--', color='grey')
```



- 선형모형에 의한 잔차값을 zscore() 함수에 대입하여 표준화 잔차값을 얻고, 이를 probplot() 함수에 대입하여 분위수 값과 표준화 잔차값을 차례로 확인할 수 있음.
- 그 값들을 scatterplot() 함수에 대입하여 잔차의 정규성을 확인하는 Q-Q plot을 확인할 수 있음.
- 그 결과 Q-Q plot의 그림이 기준선에서 약간 벗어난 형태임을 확인할 수 있음. 따라서 잔차의 정규성 가정을 만족하지 않는 것을 알 수 있음.

```
In [12]: from scipy.stats import zscore, probplot

In [13]: sr = zscore(residual)
    ...: (x, y), _ = probplot(sr)

In [14]: import seaborn as sns
    ...: import matplotlib.pyplot as plt

In [15]: sns.scatterplot(x,y)
    ...: plt.plot([-3,3], [-3,3], '--', color='grey')
```

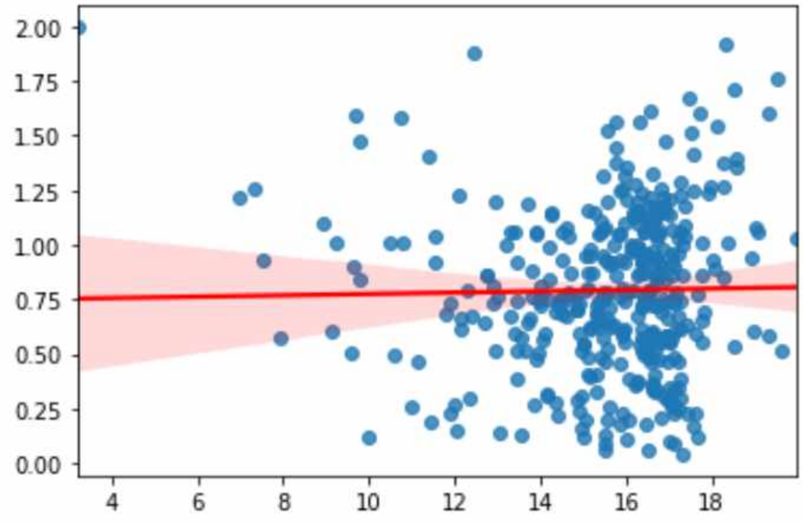


- 실제로 scipy.stats 모듈의 shapiro 함수를 통해 잔차의 정규성을 확인한 결과 p-value값이 0.0000으로 나타나 정규성을 띄지 않는다는 것을 알 수 있음.

```
In [16]: from scipy.stats import shapiro

In [17]: shapiro(residual)
Out[17]: ShapiroResult(statistic=0.96277916431427, pvalue=2.024077083717657e-08)
```

- 선형모형에 의한 예측값과 표준화 잔차값의 절대값의 제곱근값을 regplot() 함수에 대입하여 모형의 등분산성을 확인하는 그래프를 확인할 수 있음.
- 그 결과 경향선이 수평선의 형태로 나타나는 것을 확인할 수 있음. 따라서 잔차의 등분산성 가정을 만족하는 것을 알 수 있음.

```
In [18]: import seaborn as sns
    ...: from numpy import sqrt, abs

In [19]: sns.regplot(mpg_pred, sqrt(abs(sr)), line_kws={'color':'red'})
```



- 선형모형에 사용되는 변수들의 상관관계를 확인하기 위하여 corr() 함수와 seaborn 모듈의 heatmap() 함수를 이용하여 변수들간의 관계를 확인할 수 있음.
- 그 결과 출력변수인 acceleration와 입력변수인 displacement, horsepower, weight의 상관계수는 각각 -0.54, -0.69, -0.42인 것을 확인할 수 있음.
- 또한, heatmap() 함수를 통해 1에 가까운 상관계수값을 가질수록 진한 녹색의 값을 가지는 것을 확인할 수 있고, -1에 가까운 상관계수값을 가질수록 흰색에 가까운 색을 가지는 것을 확인할 수 있음.

```
In [22]: import seaborn as sns
    ...: import matplotlib.pyplot as plt

In [23]: plt.rcParams["figure.figsize"] = (8,5)
    ...: sns.heatmap(df[['acceleration', 'displacement', 'horsepower',  'weight']].corr(method='pearson'),
    ...:             annot = True, cmap = 'Greens', vmin = -1, vmax=1)
```

2. 선형회귀분석의 가정 실습

○ seaborn 모듈의 penguins 데이터에서, body_mass_g을 예측하기 위해 bill_length_mm와 bill_depth_mm를 사용한 회귀모형의 잔차를 분석하고자 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)
```

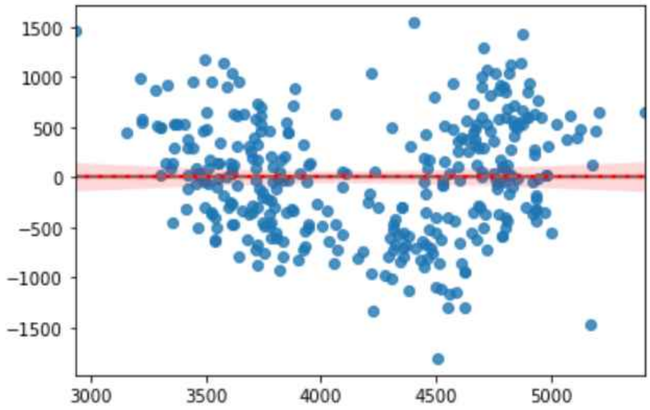
- 모형식으로 'body_mass_g ~ bill_length_mm + bill_depth_mm'를 statsmodels.formula.api 모듈의 ols().fit()에 대입하여 결과를 확인할 수 있음.
- 그 결과 Durbin-Watson 값이 1.806으로 나타나 잔차의 입력성을 확인

할 수 있음.

```
In [5]: from statsmodels.formula.api import ols

In [6]: formula = 'body_mass_g ~ bill_length_mm + bill_depth_mm'
   ...: result = ols(formula = formula, data = df).fit()

In [7]: result.summary()
Out[7]:
<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:                Thu, 26 Aug 2021   Prob (F-statistic):           1.40e-47
Time:                        17:24:16   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.
==============================================================================
```

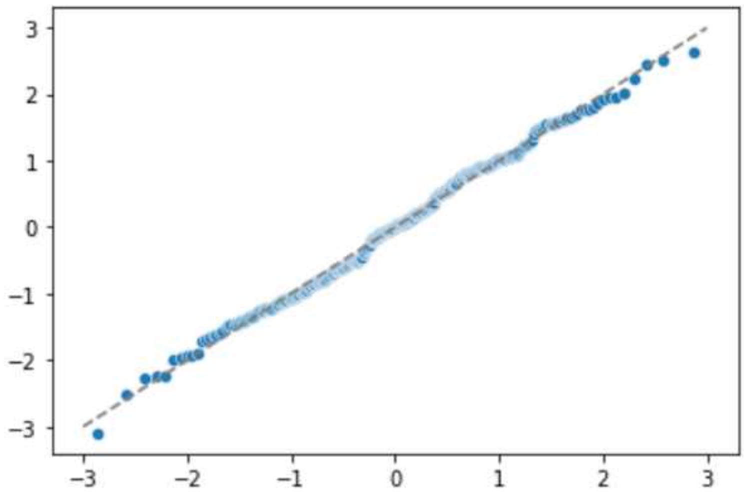
- predict() 함수에 bill_length_mm, bill_depth_mm 자료를 대입하여 회귀모형에 적용하였을 때 예측값을 확인할 수 있음.
- 이 값을 출력변수인 body_mass_g 값과의 차이를 통해 잔차값인 residual을 저장할 수 있음.
- 선형모형에 의한 예측값과 잔차 값을 regplot() 함수에 대입하여 모형의 선형성을 확인하는 그래프를 확인할 수 있음.
- 그 결과 예측값과 잔차값의 산점도 경향선이 0을 중심으로 수평선의 형태로 나타나는 것을 확인할 수 있음. 따라서 모형의 선형성 가정을 만족하는 것을 알 수 있음.

```
In [8]: peng_pred = result.predict(df[['bill_length_mm', 'bill_depth_mm']])
   ...: residual = df['body_mass_g'] - peng_pred

In [9]: import matplotlib.pyplot as plt
   ...: import seaborn as sns

In [10]: sns.regplot(peng_pred, residual, line_kws={'color':'red'})
    ...: plt.plot([peng_pred.min(),peng_pred.max()],[0,0], '--', color='grey')
```



- 선형모형에 의한 잔차값을 zscore() 함수에 대입하여 표준화 잔차값을 얻고, 이를 probplot() 함수에 대입하여 분위수 값과 표준화 잔차값을 차례로 확인할 수 있음.
- 그 값들을 scatterplot() 함수에 대입하여 잔차의 정규성을 확인하는 Q-Q plot을 확인할 수 있음.
- 그 결과 Q-Q plot의 그림이 기준선에서 거의 일치한 형태임을 확인할 수 있음. 따라서 잔차의 정규성 가정을 만족하는 것을 알 수 있음.

```
In [11]: from scipy.stats import zscore, probplot

In [12]: sr = zscore(residual)
    ...: (x, y), _ = probplot(sr)

In [13]: import seaborn as sns
    ...: import matplotlib.pyplot as plt

In [14]: sns.scatterplot(x,y)
    ...: plt.plot([-3,3], [-3,3], '--', color='grey')
```



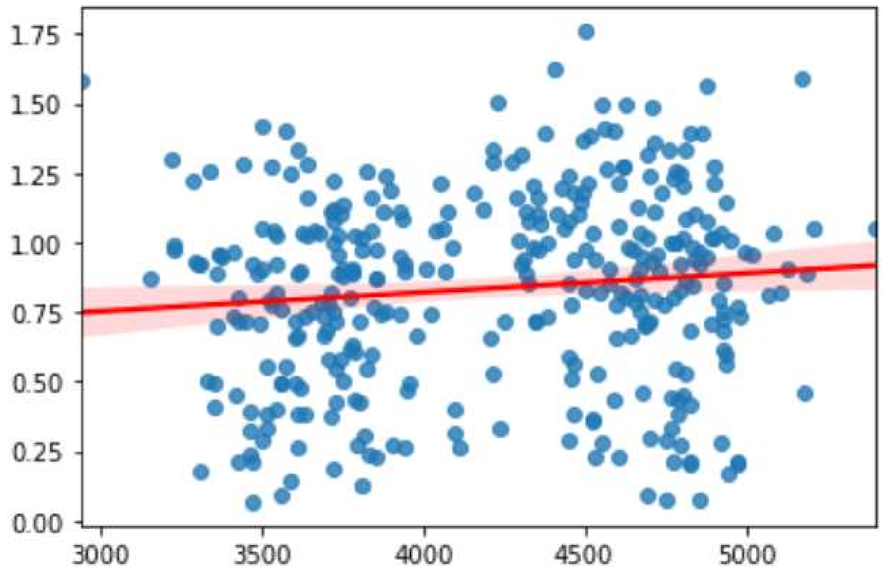
- 실제로 scipy.stats 모듈의 shapiro 함수를 통해 잔차의 정규성을 확인

한 결과 p-value값이 0.25897로 나타나 정규성을 띄는 것을 알 수 있음.

```
In [15]: from scipy.stats import shapiro
    ...: shapiro(residual)
Out[15]: ShapiroResult(statistic=0.9945068359375, pvalue=0.2589775621891022)
```

- 선형모형에 의한 예측값과 표준화 잔차값의 절대값의 제곱근값을 regplot() 함수에 대입하여 모형의 등분산성을 확인하는 그래프를 확인할 수 있음.
- 그 결과 경향선이 수평선의 형태로 나타나는 것을 확인할 수 있음. 따라서 잔차의 등분산성 가정을 만족하는 것을 알 수 있음.

```
In [16]: import seaborn as sns
    ...: from numpy import sqrt, abs

In [17]: sns.regplot(peng_pred, sqrt(abs(sr)), line_kws={'color':'red'})
```



- 선형모형에 사용되는 변수들의 상관관계를 확인하기 위하여 corr() 함수와 seaborn 모듈의 heatmap() 함수를 이용하여 변수들간의 관계를 확인할 수 있음.
- 그 결과 출력변수인 body_mass_g과 입력변수인 bill_length_mm와 bill_depth_mm의 상관계수는 각각 0.6, -0.47인 것을 확인할 수 있음.
- 또한, heatmap() 함수를 통해 1에 가까운 상관계수값을 가질수록 진한 녹색의 값을 가지는 것을 확인할 수 있고, -1에 가까운 상관계수값을 가

질수록 흰색에 가까운 색을 가지는 것을 확인할 수 있음.

```
In [18]: import seaborn as sns
    ...: import matplotlib.pyplot as plt

In [19]: plt.rcParams["figure.figsize"] = (8,5)
    ...: sns.heatmap(df[['bill_length_mm', 'bill_depth_mm', 'body_mass_g']].corr(method='pearson'),
    ...:             annot = True, cmap = 'Greens', vmin = -1, vmax=1)
```