제2차 빅데이터 분석 교육과정 강의교재

- 가완성본 -

ABRC 인공지능-빅데이터연구센터
AI-BIG DATA RESEARCH CENTER

선형회귀분석을 이용한 데이터 분석

1. 선형회귀분석

1.1 단일선형귀분석이란?

- 두 변수(종속변수, 독립변수)사이의 함수적 관계를 기술하는 수학적 방정식을 구하는데 사용
- 식은 독립변수의 값이 주어질 때 종속변수의 값을 추정하거나 예측하는데 사용
- 서로 영향을 주고 받는 인과관계를 갖는 두 변수 사이의 관계를 분석
- Python에서는 대표적으로 sklearn(scikit-learn) 패키지에서 Linear regression 회귀분석을 위한 함수를 제공

※ 종속변수(Dependent Variable)
: 독립변수의 특정한 값에 따른 그의 값을 예측하고자 하는 변수

※ 독립변수(Independent Variable)
: 다른 변수에 영향을 주고 그 변수의 값을 예측하려는 변수

회귀분석은 서로 영향을 주고받으면서 인과관계를 갖는 독립변수와 종속변수 사이의 관계를 분석하는데 사용합니다. 여기서 독립변수란 다른 변수에 영향을 주는 변수를 의미합니다. 종속변수란 독립변수의 값에 따라 그 결과가 변화되는 변수를 말합니다. 표 1은 회귀분석 적용분야의 예시에 대한 내용입니다.

1.1.1 회귀분석 적용분야 예시

종속변수	시리얼 수요	금 가격	주택 가격	중고차 가격
독립변수	제품의 가격	이자율	주택의 크기	차종
	5~12세 아동의 수	물가 상승률	침실의 수	배기량
	경쟁회사 제품의 가격	석유 가격	도로 접근성	연식
	광고 투자	보석용 금에 대한 수요	주택의 위치	관리상태
	금년도 연간 매출액	산업용 금에 대한 수요	주택의 상태	옵션
	과거년도 연간 매출액	상업용 금에 대한 수요		사고여부

표 1 회귀분석 적용분야 예시

회귀분석을 할 때 두 변수 사이의 관계가 중요한 것을 알았습니다. 하지만 이 관계를 어떻게 확인할까요? 바로 산포도를 통해 눈으로 쉽게 확인할 수 있습니다.

1.1.2 산포도 소개

- 회귀분석을 할 때는 먼저 두 변수 사이의 관계를 대략적으로 알아보기 위하여 산포도(Scatter Diagram)를 그린다. (산포도 = 산점도)
- 보통 X축에 독립변수, Y축에 종속변수를 설정하고 각 변수의 값을 나타내는 점을 도표에 나타낸 것
- 두 변수간의 관련성 및 예측을 위한 상관분석이나 회귀분석을 할 만한 자료인지를 미리 알 수 있음

산포도란 두 변수 사이의 관계를 대략적으로 알아보기 위해 X축에 독립변수를, 그리고 Y축에 종속변수를 설정하고 각 변수의 값을 나타내는 점을 나타낸 것입니다. 이렇게 나온 산포도를 통해 우리는 회귀분석을 할 만한 자료인지 아닌지 미리 알 수 있습니다.

1.1.2.1 산포도 예시

산포도를 그리는 법을 먼저 알아보고 갈까요? 예시는 차량가격과 판매액을 표시하고 있습니다. 여기서 차량 가격은 독립변수고 판매액은 종속변수겠지요? 앞에서 얘기했듯이 독립변수는 X축, 판매액은 Y축에 값을 설정해서 점으로 나타내면 오른쪽 산포도와 같습니다. 이제 산포도를 해석해보겠습니다. 산포도를 살펴보면 차량가격이 증가할 때 판매액도 같이 증가하는 경향이 있는 것을 확인할 수 있습니다. 표 2는 산포도 예시 데이터에 대한 내용입니다.

또한 점을 이으면 우상향 하는 하나의 선을 그릴 수 있는데 이것은 양의 상관관계가 있음을 의미합니다. 그림 1은 산포도 예시에 대한 내용입니다. 지금까진 회귀분석의 기본개념에 대해서 설명을 드렸습니다.

차종	차량 가격 (x)	판매액
A	13	40
B	19	83
C	16	62
D	14	48
E	15	58
F	14	43

표 2 산포도 예시 데이터

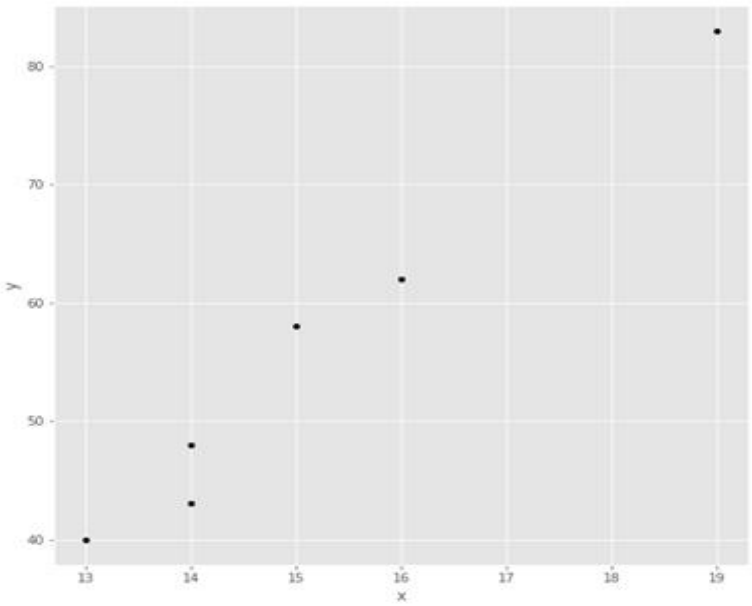


그림 1 산포도 예시

1.2 단일선형회귀모델 소개

- 종속변수 Y가 독립변수 X와 오차항(Error Term)에 어떻게 관련되어 있는가를 나타내는 방정식

※ 오차항(Error Term)

: 독립변수 X의 값이 주어질 때 종속변수 Y의 실제 값과 예측 값의 차이를 말함

※ 단일선형회귀모델 식

$$Y_i = \alpha + \beta X_i + \epsilon_i$$

- 위 식은 X라는 독립변수가 Y라는 종속변수에 주는 영향력을 식으로 나타낸 것
  - $Y\,i$: i번째 관측치에 대한 종속변수의 값
  - $X\,i$: 이미 알려진 독립변수의 i번째 값
  - α : X값이 변해도 Y의 변동에는 영향을 주지 않는 회귀 계수
  - β : X의 영향력을 크기와 부호로 나타내 주는 회귀 계수, 독립변수 X의 기울기
  - $\varepsilon\,i$: i번째 관측치에 대한 오차항

이제 회귀분석 중 하나인 단일선형회귀모델에 대해서 알아보도록 하겠습니다. 식을 보면 종속변수 Y의 값을 구하기 위해선 회귀계수 알파, 베타와 오차항을 모두 더해주면 구할 수 있습니다.

1.2.1 단일선형회귀모델의 가정

- 하나의 종속 변수와 하나의 독립변수를 분석
- 독립변수 X의 각 값에 대한 Y의 확률분포가 존재함
- Y의 확률분포의 평균은 X값이 변함에 따라 일정한 추세를 따라 움직인다.
- 종속변수와 독립변수 간에는 선형 함수 관계가 존재함

여기서 알파는 X값이 변해도 Y의 변동에는 영향을 주지 않는 회귀 계수입니다. 알파와 다르게 베타는 X의 영향력을 나타내 주는 회귀 계수로 독립변수 X의 기울기를 뜻합니다. 마지막으로 오차항을 더해주어 실제 종속변수 Y의 값이 됩니다. 이번엔 회귀 계수를 추정하는 방법을 알아보도록 하겠습니다.

1.2.2 회귀계수 추청

- 수집된 데이터(산포도)에 가장 적절한 회귀 직선을 구하는 것
- 방법으로는 최소자승법이 사용됨

회귀 계수를 추정한다는 것은 독립변수 X와 종속변수 Y에 대해 가장 적절한 회귀 직선을 구하는 것입니다. 회귀 직선을 구하기 위해선 최소자승법이 사용됩니다.

1.2.3 최소자승법

- 잔차를 자승한 값들의 합이 최소가 되도록 표본회귀식의 a와 b를 구하는 방법
- 측정값을 기초로 해서 적당한 제곱합을 만들고 그것을 최소로 하는 값을 구하여 측정결과를 처리하는 방법
- ※ 잔차(Residual)

  : 독립변수 X의 값이 주어질 때 표본회귀선의 예측 값 Y 과 실제 값 $Y\ i$ 사이에 표본오차 때문에 발생하는 차이

$$e_i = Y_i - \hat{Y}$$

최소자승법이란 잔차를 제곱한 값들의 합이 최소가 되도록 하는 표본회귀

식의 a와 b를 구하는 방법입니다. 잔차란 독립변수 X의 값이 주어질 때 표본회귀선의 예측 값과 실제 값 사이의 차이입니다. 잔차는 수식으로 소문자 e로 표시됩니다. 변수 값 위에 바 표시가 있는 것은 해당 변수의 평균을 의미합니다.

1.2.3.1 최소자승법 식

- 자료들을 표시하는 각 좌표들과 사이를 지나는 직선과의 수직적 길이(차이)를 가장 짧게 하는 방법을 찾기 위해 각 수직적 길이를 제곱하여 합한 값을 최소화 함
- 큰 폭의 오차에 대해 보다 더 큰 가중치를 부여함으로써, 독립변수 값이 동일한 평균치를 갖는 경우 가능한 한 변동 폭이 작은 표본회귀선을 도출하기 위한 것

그림 2는 회귀계수에 대한 내용입니다.

표본회귀선의 회귀계수

$$b = \frac{n\sum X_i Y_i - \sum X_i \sum Y_i}{n\sum X_i^2 - (\sum X_i)^2} = \frac{\sum X_i Y_i - n\overline{XY}}{\sum X_i^2 - n\overline{X}^2}$$

$$a = \overline{Y} - b\overline{X}$$

그림 2 회귀계수

1.2.4 표본회귀계수 구하는 예

표 3은 표본회귀 예제 데이터에 대한 내용이고, 표 4는 선형회귀식 적용예에 대한 내용입니다. 3번째 열에 있는 값은 선형회귀식을 적용했을 때 예측 값이 나온 것을 확인할 수 있습니다. 2번째 열에 실제 값에서 이 값을 빼면 4번째 열과 같이 값의 차이 즉, 잔차를 구할 수 있습니다.

그림 3은 표본회귀선의 예시에 대한 내용입니다. 자 다음에는 적합도 검증이라는 것에 대해 알아보도록 하겠습니다.

차종	차량가격(x_i)	판매액(y_i)	x_i^2	$x_i y_i$
A	13	40	169	520
B	19	83	361	1577
C	16	62	256	992
D	14	48	196	672
E	15	58	225	870
F	14	43	196	602
합계	91	334	1403	5233

표 3 표본회귀 예제 데이터

$$\bar{x} = \frac{91}{6} = 15.17,\ \bar{y} = \frac{334}{6} = 55.67,\ b = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum {x_i}^2 - n\bar{x}^2} = \frac{5233 - 6(15.17)(55.67)}{1403 - 6(15.17^2)} = 7.4$$
$$a = \bar{y} - b\bar{x} = 55.67 - 7.46(15.17) = -57.5,\ \hat{y} = -57.5 + 7.46x$$

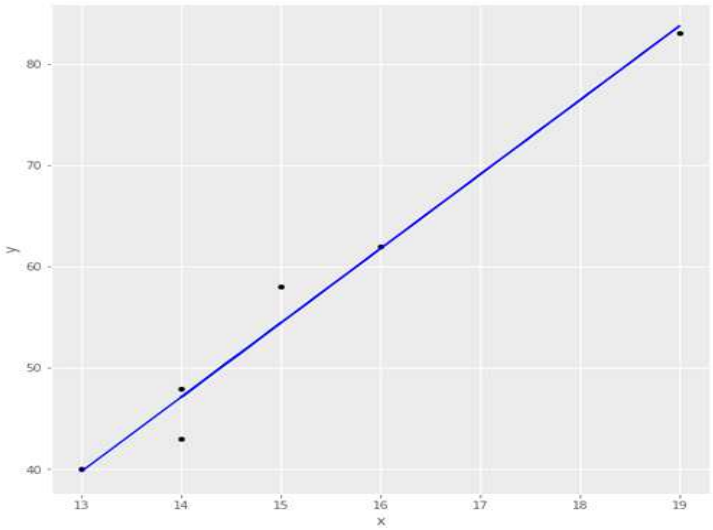


그림 3 표본회귀선 예시

차량 가격(x)	판매액(y)	$\hat{y} = -57.5 + 7.46x$	$e = y - \hat{y}$
13	40	39.48	0.52
19	83	84.24	-1.24
16	62	61.86	0.14
14	48	46.94	10.6
15	58	54.4	3.6
14	43	46.94	-3.94

표 4 선형회귀식 적용 예

$$\hat{y} = -57.5 + 7.46(20) = 91.7$$

1.3 적합도 검증

1.3.1 적합도 검증이란?

- 표본자료를 사용하여 구한 표본회귀식이 종속변수의 값을 어느 정도 정확하게 예측할 수 있는가의 정도를 검증
- 두 변수 값들이 표본회귀선 주위에 몰려 있으면 종속변수의 실제 값과 예측 값 차이인 잔차가 줄어들어 예측의 정확성이 높아짐

적합도 검증이란 표본자료를 사용하여 구한 표본회귀식이 종속변수의 값을 어느 정도 정확하게 예측할 수 있는가의 정도를 검증하는 방법입니다.

1.3.2 적합도 검증 방법

- 추정의 표준오차
- 결정계수

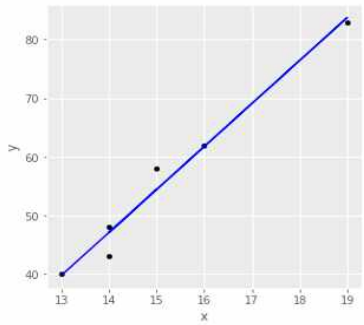
적합도를 검증하는 방법은 추정의 표준오차와 결정계수를 구하는 방법이 있습니다. 표준오차의 값이 클수록 실제 값들이 표본회귀선 주위로 널리 흩어지기 때문에 정확도가 낮고 반대로 작을수록 표본회귀선 주위에 모여들어 정확도가 높아집니다. 여기서 추정의 표준오차와 표준편차에 차이점이 어떤 차이가 있는지 궁금하실 겁니다.

1.3.3 추정의 표준오차 (standard error of estimate)

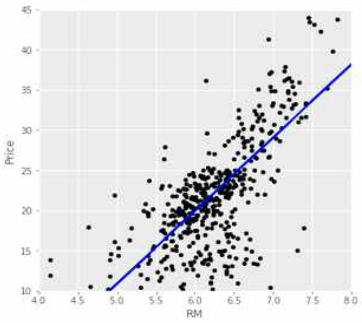
- 값이 클수록 실제 값들이 표본회귀선 주위로 널리 흩어지고 작을수록 실제 값들이 표본회귀선 주위로 모여들어 그 표본회귀선을 이용한 종속변수 값의 예측에 대한 정확도는 높아짐

- 추정의 표준오차(S_e)와 표준편차의 차이
  - 추정의 표준오차 : 표본들의 실제 값들이 표본회귀선 주위로 흩어진 변동을 측정
  - 표준편차 : 표본들의 실제 값들이 평균 주위로 흩어진 변동을 측정

그림 4는 추정의 표준오차에 대한 내용입니다.



S_e가 작은 경우



S_e가 큰 경우

그림 4 추정의 표준오차

기존에 많이 알고 있는 표준편차란 표본들의 실제 값들이 평균 주위로 흩어진 정도를 측정하는 방식이라면 추정의 표준오차는 평균 주위가 아닌 표본회귀선 주위로 흩어진 정도를 측정하는 것이 그 차이입니다. 먼저 추정의 표준오차 식에 대해 알아보도록 하겠습니다.

1.3.3.1 추정의 표준오차 식

- 예측 값과 실제 값의 차이를 잔차라고 하면 추정치의 표준오차(S_e)는 잔차들의 표준편차를 구하기 위한 식
- 실제 값이 회귀식에서 얼마나 떨어져 있는가를 나타내기 위함
- 추정 표준오차 계산 기준은 회귀직선, 절편과 기울기의 두 통계량에 의해 결정되므로 자유도는 2만큼 감소

$$S_e = \sqrt{\frac{\sum(y_i - \hat{y}_i)^2}{n-2}} = \sqrt{\frac{SSE}{n-2}} = \frac{\sum y^2 - a\sum y - b\sum xy}{n-2}$$

- SSE : 오차제곱합

추정의 표준오차는 잔차들의 표준편차를 구하기 위한 식입니다. 표준 오차 값의 계산은 오차 제곱합에 표본의 개수 n에서 2를 뺀 값으로 나누고 제곱근을 하여 구할 수 있습니다. 이때 n-2인 이유는 회귀직선, 절편과 기울기의 두 통계량에 의해 결정 되서 자유도가 2만큼 감소하기 때문입니다. 표 5는 추정의 표준오차 예시에 대한 내용입니다.

x	y	x^2	y^2	xy
13	40	169	1600	520
19	83	361	6889	1577
16	62	256	3844	992
14	48	196	2304	672
15	58	225	3364	870
14	43	196	1849	602
91	334	1403	19850	5233

표 5 추정의 표준오차 예시

$$S_e = \frac{\sum y^2 - a\sum y - b\sum xy}{n-2} = \frac{19850 - (-57.5)(334) - 7.46(5233)}{6-2} = 4.205$$

예시를 통해 계산해보면 다음과 같이 간단하게 구할 수 있는 것을 확인할 수 있습니다. 추정의 표준오차 값이 4.205로 낮은 수치를 보이는 것을 확인할 수 있습니다. 값이 낮기 때문에 회귀분석을 하기 적합하다고 판단할 수 있습니다.

자 이제 결정계수라는 것을 구하기 위해 먼저 총변동이라는 것에 대해 알아보도록 하겠습니다.

1.3.4 총변동 (total variation)

- 총제곱합(Sum of Squares Total : SST) = 회귀제곱합(Sum of Squares Regression:SSR) + 잔차제곱합(Sum of Squares Error:SSE)

$$\sum (y_i - \bar{y})^2 = \sum (\hat{y}_i - \bar{y})^2 + \sum (y_i - \hat{y}_i)^2$$

$$SST = SSR + SSE$$

- SST : 실제 값 $y\ i$ 들이 이들의 평균 y 로부터 흩어진 정도
- SSR : 예측치와 실제 값 $y\ i$ 들의 평균 y 의 차이의 제곱의 합
- SSE : 예측치와 실제 값의 차이의 제곱의 합

총변동은 회귀제곱합과 잔차제곱합을 더하여 구할 수 있습니다. 이렇게 구해진 총변동은 실제의 Y 값들이 이들의 평균으로부터 흩어진 정도를 의미합니다. 총변동은 회귀제곱합과 잔차제곱합을 더하여 구할 수 있습니다. 이렇게 구해진 총변동은 실제의 Y 값들이 이들의 평균으로부터 흩어진 정도를

의미합니다.

1.3.5 결정계수 (Coefficient of Determination)

$$R^2 = \frac{\text{설명되는 변동}}{\text{총변동}} = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} = \frac{a\sum y_i + b\sum x_i y_i - n\bar{y}^2}{\sum {y_i}^2 - n\bar{y}^2}$$

- 결정계수는 0부터 1까지의 값을 가짐
- 표본회귀선이 모든 자료에 완전히 적합하면 SSE=0, R 2 =1이 됨
- R 2 의 값이 1에 가까울수록 표본회귀선으로 종속변수의 실제 값 y_i를 예측하는데 정확성이 더 높음

- ※ 총편차(Total Deviation) = 설명된 편차 + 설명 안 된 편차

$$(y_i - \bar{y}) = (\hat{y_i} - \bar{y}) + (y_i - \hat{y_i})$$

결정계수는 0부터 1까지의 값을 가지는데 1에 가까울수록 실제 값 Y를 예측하는데 정확성이 더 높은 것을 의미합니다. 결정계수를 예시를 통해 계산해보도록 하겠습니다. 표 6은 결정계수 예시에 대한 내용입니다.

x	y	$\hat{y}$	$y-\hat{y}$	$(y-\hat{y})^2$	$y-\bar{y}$	$(y-\bar{y})^2$
13	40	39.48	0.52	0.2704	-15.67	245.44
19	83	84.24	-1.24	1.5376	27.33	747.11
16	62	61.86	0.14	0.0196	6.33	40.11
14	48	46.94	1.06	1.1236	-7.67	58.78
15	58	54.4	3.6	12.96	2.33	5.44
14	43	46.94	-3.94	15.5236	-12.67	160.44
91	334	333.86	0.14	31.4348	-0.02	1257.33
SEE=31.4348 SST=1257.33						

표 6 결정계수 예시

- $SSR = SST - SSE = 1257.33 - 31.4348 = 1225.895$
- $R^2 = \frac{SST}{SST} = \frac{1225.895}{1257.33} = 0.975$
- 차량가격이 판매액 변동 97.5%를 결정하고 다른 요인들이 나머지 2.5%의 영향을 미침

앞에서 학습한 수식을 통해 아래와 같이 각 값을 계산할 수 있습니다. 최종 결과로 결정계수 값이 0.975가 나온 것을 확인할 수 있습니다. 이 값이 의미하는 것은 예시로 다룬 차량가격이 판매액 변동의 97.5%를 결정하고 나머지는 2.5% 영향을 미침을 의미합니다. 적합도 검증을 통해 수치가 매우 높은 것을 확인했기 때문에 독립변수를 교체하지 않아도 됨을 알 수 있습니다. 다음에는 MSE 즉 평균제곱오차라는 것에 대해 알아보겠습니다.

1.4 성능평가

1.4.1 잔차 (Residuals)

- 회귀분석 모델의 예측 값 $\hat{y}$와 실제 값 y_i사이 차이

$$e_i = y_i - \hat{y}$$

1.4.2 Mean Squared Error (MSE)

- 평균제곱오차
- 회귀선과 모델 예측 값 사이의 오차를 사용
- 오차를 제곱한 값들의 평균

$$\frac{\sum_{i=1}^{n}(\hat{y}-y_i)^2}{n}$$

MSE는 오차를 제곱한 값들의 평균 값을 의미합니다.

1.4.3 Root Mean Squared Error (RMSE)

- MSE에서 구한 값에 루트를 적용한 값

$$\sqrt{\frac{\sum_{i=}^{n}(\hat{y}-y_i)^2}{n}}$$

마지막으로 RMSE라고 하는 것이 있는데 이는 MSE 값에 루트를 씌어서 구하게 됩니다. 이 성능 평가 방식들은 회귀분석에서 대표적으로 쓰이는 방식들입니다. 자 이제 학습한 내용을 실습하는 시간을 갖도록 하겠습니다. 실

습이 진행되는 순서는 간단한 데이터를 생성해 단일선형회귀분석을 하는 것과 실제 데이터를 사용해 단일 선형회귀분석을 진행하도록 하겠습니다.

1.5 단일선형회귀분석 실습 - Basic 1

1.5.1 Python package 로드 및 matplotlib 출력 옵션설정

우선 사용할 패키지들을 가져오도록 하겠습니다. 첫 번째 줄은 우리가 단일선형회귀분석을 하기 위해서 사용할 sklearn에 패키지 중 linear_model 모듈을 가져오는 코드입니다. linear_model은 회귀분석을 할 수 있게 도와주는 모듈입니다.

두 번째 줄은 numpy 패키지를 np라는 이름으로 가져옵니다. Numpy는 앞에서 배운 것처럼 행렬, 벡터 등의 수학 계산을 위한 자료구조와 계산 함수를 제공하는 패키지입니다. 세 번째 줄은 Pandas 패키지를 pd라는 이름으로 가져옵니다.Pandas패키지는 CSV파일 또는 데이터베이스에서 데이터를 읽고 쓸 수 있고 또한 데이터를 쉽게 조작해 새로운 컬럼을 추가할 수 있는 패키지입니다.

네 번째 줄은 Matplotlib 패키지를 가져옵니다. Matplotlib은 막대그래프, 히스토그램, 파이차트, 산점도 등 다양한 그림을 그리는데 사용되는 파이썬의 시각화 패키지입니다. 다섯 번째 줄은 matplotlib의 서브패키지인 pyplot을 “plt”라는 이름으로 가져오는 코드입니다.

여섯 번째 줄은 “%matplotlib inline”이라는 명령어를 호출합니다. 이는 matplotlib의 결과를 Ipython notebook 안에서 출력하기 위해 사용하는 명령어입니다. 그리고 일곱 번째 줄은 matplotlib 패키지에서 제공하는 스타일 중 하나인 “ggplot”을 지정해서 사용하는 명령어입니다. 그림 5는 python package 가져오기와 matplotlib 출력 옵션 설정에 대한 내용입니다.

```
1 from sklearn import linear_model
2 import numpy as np
3 import pandas as pd
4 import matplotlib
5 import matplotlib.pyplot as plt
6 %matplotlib inline
7 matplotlib.style.use('ggplot')
```

그림 5 Python package 로드 및 matplotlib 출력 옵션 설정

1.5.2 데이터 생성

다음은 사용할 데이터를 만들도록 하겠습니다. 첫 번째 줄은 data 변수 안에 dictionary형태에 x, y key이름을 가진 데이터를 생성합니다. 세 번째 줄은 data변수 안에 저장된 데이터를 pandas.DataFrame 함수를 통해 2차원의 수정 가능한 테이블 형태의 구조로 변경 후 data 변수에 저장합니다. 해당 data변수를 출력하면 x, y 컬럼 명을 가진 데이터프레임이 생성된 것을 확인할 수 있습니다. 그림 6은 데이터 생성에 대한 내용입니다.

```
1 data = {'x' : [13, 19, 16, 14, 15, 14],
2         'y' : [40, 83, 62, 48, 58, 43]}
3 data = pd.DataFrame(data)
4 data
```

	x	y
0	13	40
1	19	83
2	16	62
3	14	48
4	15	58
5	14	43

그림 6 데이터 생성

다음으로 산점도로 데이터를 표현하도록 하겠습니다. Pyplot.plot 함수를 사용하여 선이나 마커를 플롯할 수 있는데 첫 번째 줄 plot앞에 data변수를 입력했습니다. 이유는 plot 앞에 데이터프레임 형식에 변수를 놓으면 컬럼 명만 입력해서 쉽게 데이터를 가져올 수 있기 때문입니다. 자 이제 plot함수 안에 있는 파라미터들에 대해서 알아보도록 하겠습니다.

Kind 파라미터는 기본 선 이외의 여러 가지 플롯 스타일을 입력해서 사용 가능 하도록 하는 파라미터입니다. Kind 파라미터에 Scatter를 입력한 것은 산점도를 그리기 위해서 사용했습니다. 다음으로 x, y 파라미터가 있습니다. X, y 파라미터는 한 열을 다른 열에 대해 플롯하기 위해 사용합니다. X는 x축을 의미하고 y는 y축을 의미합니다. 따라서 여기서 우리는 생성한 데이터 x, y를 알맞은 파라미터에 사용했습니다.

1.5.3 산점도로 표현

다음은 figsize 파라미터가 있습니다. figsize는 플롯의 크기를 조정할 때 사용하는 파라미터입니다. 안에 숫자를 입력해서 크고 작게 변경 가능합니다. 마지막 파라미터로 color가 있습니다. 해당 코드를 실행하면 산점도를 출력할 수 있습니다. 그림 7은 산점도 표현에 대한 내용입니다.

```
1 data.plot(kind="scatter", x='x', y='y', figsize=(5,5), color="black")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xcbdca90>
```

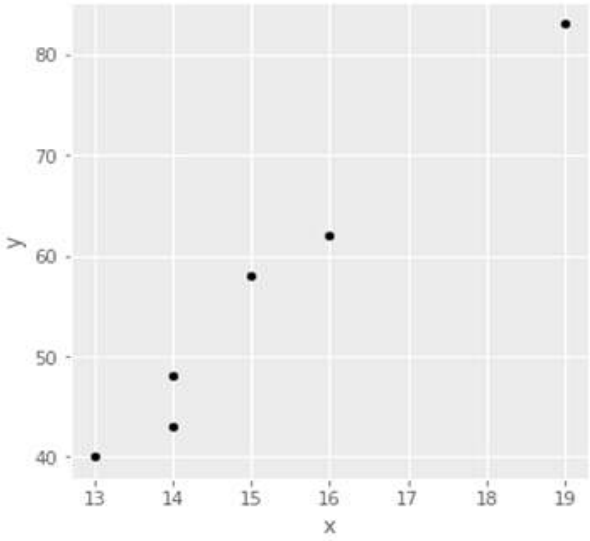


그림 7 산점도 표현

1.5.4 데이터 학습

다음은 선형회귀모델을 만들어 데이터를 학습시켜보도록 하겠습니다. 첫 번째 줄은 linear_model. LinearRegression 함수를 통해 선형회귀모델을 만들어 linear_regression 변수 안에 저장합니다. 두 번째 줄은 linear_regression.fit 함수를 이용해 모델을 학습하게 하는 함수입니다. 학습을 시키기 위해선 X, y 값을 입력해야 하는데 X값은 꼭 2차원 형태로 입력해야 하고 y값은 기존형태로 입력하면 됩니다. X값엔 data변수 안에 있는 “x”데이터 y값엔 “y”데이터를 입력했습니다.

X는 독립변수, y는 종속변수를 입력하면 됩니다. 세 번째 줄은 linear_regression.predict함수를 통해 학습된 선형회귀모델에 “x”값을 입력 값으로 해서 y값을 예측합니다. 예측된 y값은 prediction 변수에 저장됩니다. 네 번째 줄은 linear_regression.intercept_를 통해 선형회귀모델의 a계수를 출력합

니다. Intercept_는 sklearn 패키지 linear_model 모듈에서 제공해주는 함수입니다. 다섯 번째 줄은 linear_regression.coef_를 통해 선형회귀모델의 b계수를 출력합니다. coef_는 sklearn 패키지 linear_model 모듈에서 제공해주는 함수입니다. 그림 8은 데이터 학습시키는 내용입니다.

```
1 linear_regression = linear_model.LinearRegression()
2 linear_regression.fit(X = pd.DataFrame(data["x"]), y = data["y"])
3 prediction = linear_regression.predict(X = pd.DataFrame(data["x"]))
4 print('a value = ', linear_regression.intercept_)
5 print('b balue =', linear_regression.coef_)
```

```
a value =  -55.4817518248
b balue = [ 7.32846715]
```

그림 8 데이터 학습

1.5.5 적합도 검증

다음은 잔차를 구하도록 하겠습니다. 첫 번째 줄은 잔차를 구하는 공식인 실제 값 “y”에서 prediction에 저장된 예측 값 y를 빼주어 residuals 변수에 저장합니다. 두 번째 줄은 pandas.DataFrame.decribe함수를 통해 다양한 요약 통계를 생성합니다. 숫자 형의 경우 전체 개수, 평균, 표준편차, 최댓값, 최솟값, 백분위 수를 보여줍니다. 그림 9는 적합도 검증 중 잔차에 대한 내용입니다.

```
1 residuals = data["y"] - prediction
2 residuals.describe()
```

```
count    6.000000e+00
mean     5.921189e-15
std      2.491445e+00
min     -4.116788e+00
25%     -5.164234e-01
50%      2.189781e-01
75%      7.189781e-01
max      3.554745e+00
Name: y, dtype: float64
```

그림 9 적합도 검증 - 잔차

다음은 적합도 검증 방법 중 결정계수를 구하도록 하겠습니다. 첫 번째 줄에서 residuals에 저장된 잔차 값을 제곱한 값을 numpy.sum함수를 이용해 더해준 후 SSE 변수에 저장합니다. SSE 변수는 결정계수 값을 구하기 위해 사용됩니다. 두 번째 줄에서 “y”에 저장된 실제 y값에서 numpy.mean 함수를 이

용해 “y”를 평균한 값을 빼고 제곱한 값을 numpy.sum함수를 이용해 더해준 후 SST변수에 저장합니다. SST 변수는 결정계수 값을 구하기 위해 사용됩니다. 출력 된 결정계수 97.5%로 결과를 통해 x값이 y값에 많은 영향을 주는 것을 확인했습니다. 그림 10은 적합도 검증 중 결정계수에 대한 내용입니다.

```
1 SSE = (residuals**2).sum()
2 SST = ((data["y"]-data["y"].mean())**2).sum()
3 R_squared = 1 - (SSE/SST)
4 print('R_squared = ', R_squared)
```

```
R_squared =  0.9753156179610034
```

그림 10 적합도 검증 - 결정계수

1.5.6 예측하여 플롯으로 표현

다음은 예측한 값을 이용해 산포도에 선형회귀선을 그리도록 하겠습니다. 첫 번째 줄에선 plot함수를 통해 산점도를 그렸습니다. 앞에서 설명했던 부분이므로 넘어가도록 하겠습니다. 네 번째 줄은 산점도 위에 선형회귀선을 그립니다. Plot 함수 안에 파라미터 x, y를 입력하지 않으면 앞에 변수가 x축, 뒤에 변수가 y축으로 기본 세팅 됩니다. 출력된 결과를 보면 구하고자 했던 선형회귀선이 그려진 것을 확인할 수 있습니다. 그림 11은 선형회귀선과 산점도 표현에 대한 내용입니다.

```
data.plot(kind="scatter",x="x",y="y",figsize=(5,5),color="black")

# Plot regression line
plt.plot(data["x"],prediction,color="blue")
```

```
[<matplotlib.lines.Line2D at 0xfc915f8>]
```

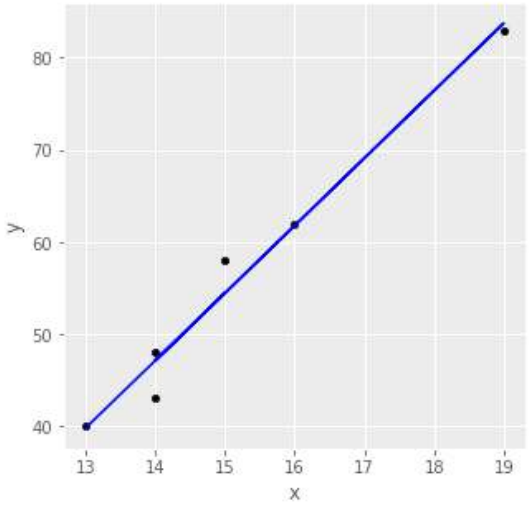


그림 11 선형회귀선 및 산점도 표현

1.5.7 성능 평가

다음은 생성된 회귀분석모델에 대한 성능평가를 진행하도록 하겠습니다. 첫 번째 줄은 sklearn 패키지에서 제공하는 MSE를 구하기 위한 모듈 mean_squared_error를 가져옵니다. 두 번째 줄은 sklearn.metrics.score 함수로 예측한 결과 값과 정확한 결과 값을 비교해서 성능을 평가합니다. Score함수는 X, y값을 입력해서 성능을 평가합니다.

파라미터를 살펴보면 독립변수 “x”를 2차원 DataFrame형태로 교체후 X값로 지정해주고 종속변수 “y”값은 y값으로 지정해서 학습한 모델을 통해 성능평가합니다. 세 번째 줄은 sklearn 패키지에서 제공하는 mean_squared_error모듈을 이용해 평균제곱오차 값을 구합니다.

해당 모듈에 파라미터 값은 학습한 모델을 통해 나온 예측 값 prediction 변수와 실제 값이 저장된 “y”값을 입력합니다. 네 번째 줄은 RMSE 값을 구합니다. 구하는 방법은 세 번째 줄에서 구한 평균제곱오차 값에 루트를 씌워주었습니다. 루트는 **0.5 수식으로 구하였습니다. 네 번째 줄은 RMSE 값을 구합니다. 구하는 방법은 세 번째 줄에서 구한 평균제곱오차 값에 루트를 씌워주었습니다. 루트는 **0.5 수식으로 구하였습니다. 이제 결과를 확인해 보도록 하겠습니다. 보시면 아시겠지만 앞에서 구한 결정계수와 score함수를 통

해 나온 결과가 동일한 것을 확인할 수 있습니다. RMSE 결과를 통해 오차도 작은 것을 확인했습니다. 그림 12는 성능평가에 대한 내용입니다.

```
from sklearn.metrics import mean_squared_error
print('score = ', linear_regression.score(X = pd.DataFrame(data["x"]), y = data["y"]))
print('Mean_Squared_Error = ', mean_squared_error(prediction, data['y']))
print('RMSE = ', mean_squared_error(prediction, data['y'])**0.5)
```

```
score =  0.975315617961
Mean_Squared_Error =  5.17274939173
RMSE =  2.27436791037
```

그림 12 성능평가

지금까지 임의의 데이터를 사용해서 간단히 실습을 했습니다. 다음에는 실제 데이터를 가지고 실습을 해보도록 하겠습니다.

1.6 단일선형회귀분석 실습 - Basic 2

1.6.1 Boston dataset 로드

첫 번째 줄은 sklearn 패키지에서 제공하는 open dataset을 가져오기 위해 용하는 모듈 datasets입니다. 두 번째 줄은 datasets 모듈을 통해 보스턴 집 가격 데이터를 가져와 boston_house_prices 변수에 저장합니다. 세 번째 줄은 로드한 보스턴 전체 데이터에 key값을 출력합니다. 네 번째 줄은 보스턴 전체 데이터 중 data에 대한 전체 행, 열 길이를 출력합니다. 다섯 번째 줄은 보스턴 데이터에 사용하는 컬럼 이름을 출력합니다. 그림 13은 boston dataset을 로드하는 내용입니다.

```
from sklearn import datasets
boston_house_prices = datasets.load_boston()
print(boston_house_prices.keys())
print(boston_house_prices.data.shape)
print(boston_house_prices.feature_names)
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR'])
(506, 13)
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

그림 13 boston dataset 로드

1.6.2 Boston dataset 정보 확인

다음은 보스턴 데이터 셋 정보를 출력해 보도록 하겠습니다. Sklearn에서 제공하는 open dataset은 DESCR이라는 데이터에 대한 정보를 같이 제공합니다. 출력된 결과물을 확인하시면 각 컬럼들에 대한 설명과 길이가 적힌 것을

확인할 수 있습니다. 그림 14는 boston dataset의 정보 확인에 대한 내용입니다.

```
1 print(boston_house_prices.DESCR)
```

```
Boston House Prices dataset
===========================

Notes
------
Data Set Characteristics:

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive

    :Median Value (attribute 14) is usually the target

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's
```

그림 14 boston dataset 정보 확인

1.6.3 Boston dataset을 데이터프레임으로 정제

다음은 보스턴 데이터 셋을 데이터프레임으로 정제하도록 하겠습니다. 첫 번째 줄은 boston_house_price 변수에 전체 데이터 중 data에 해당하는 값만 DataFrame형으로 변경 후 data_frame에 저장합니다. 두 번째 줄은 tailer.tail함수를 사용해 전체 데이터 중 마지막 5개 데이터만 출력합니다. 5개가 출력되는 이유는 뭘까요?

이유는 괄호 안에 숫자를 입력하지 않을 시 기본 값으로 5개가 지정되기 때문입니다. 다음은 data_frame안에 저장된 데이터프레임에 컬럼 명을 교체하도록 하겠습니다. 기존 boston_frame 컬럼 이름을 확인하면 숫자로 구성된 것을 확인할 수 있었습니다. 그림 15는 boston dataset을 데이터프레임 형식으로 정제한 내용입니다.

```
data_frame = pd.DataFrame(boston_house_prices.data)
data_frame.tail()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

그림 15 boston dataset을 데이터프레임 형식으로 정제

이제 이 숫자로 구성된 컬럼 명을 보스턴 데이터에 원래 컬럼 명인 feature_names로 교체하도록 하겠습니다. 간단히 데이터프레임이 저장된 변수 뒤에 columns를 입력해 주고 교체를 원하는 feature_names를 저장해주면 됩니다. 결과를 보면 컬럼 명이 바뀐 것을 확인할 수 있습니다.

다음은 예측하고자 하는 대상인 종속변수 y값을 데이터프레임에 추가하도록 하겠습니다. 첫 번째 줄은 data_frame에 'Price'라는 컬럼을 만들고 boston_house_prices에 저장된 데이터 중 "target"데이터 즉 종속변수를 데이터프레임에 저장합니다. 결과물을 확인하면 Price라는 컬럼이 추가된 것을 확인할 수 있습니다. 그림 16은 데이터프레임으로 정제한 후 컬럼명 변경에 대한 내용입니다.

```
data_frame['Price'] = boston_house_prices.target
data_frame.tail()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88	11.9

그림 16 데이터프레임 형식에서 컬럼명 변경

1.6.4 산점도로 표현

다음은 산점도록 데이터를 나타내도록 하겠습니다. Plot함수를 통해 x축엔 "RM" 독립변수 y축엔 "Price" 종속변수를 그렸습니다. 이때 새로운 파라미터 xlim, ylim이 생겨난 것을 확인할 수 있습니다. xlim, ylim파라미터는 x축과 y축에 범위를 사용자 임의로 설정을 해줄 때 사용하는 파라미터입니다. X축

을 보면 입력된 4에서 8까지 숫자로 되어있는 것을 확인할 수 있습니다. 그림 17은 산점도 표현에 대한 내용입니다.

```
1 data_frame.plot(kind="scatter", x="RM", y="Price", figsize=(6,6),
2                 color="black", xlim = (4,8), ylim = (10,45))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xd07c048>
```

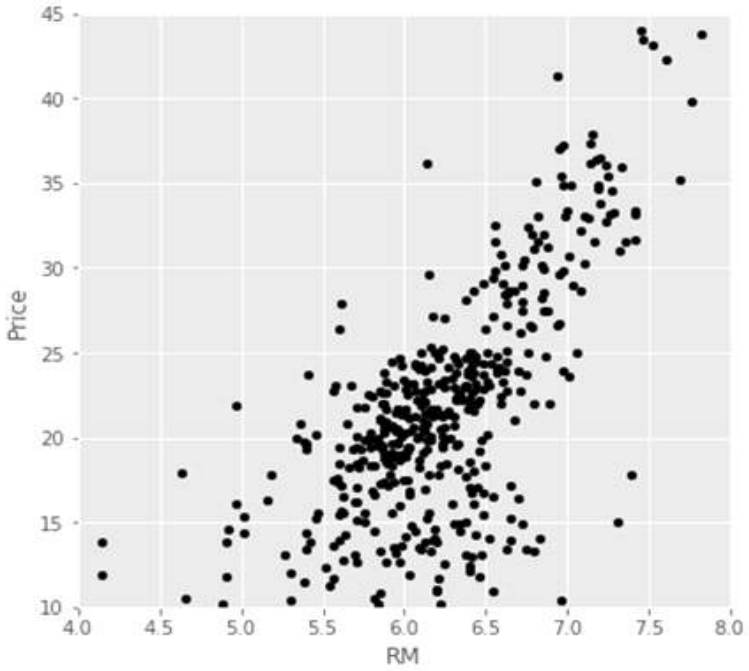


그림 17 산점도 표현

1.6.5 데이터 학습

다음은 선형회귀모델을 만들어 데이터를 학습시켜보도록 하겠습니다. 첫 번째 줄은 linear_model. LinearRegression 함수를 통해 선형회귀모델을 만들어 linear_regression 변수 안에 저장합니다. 두 번째 줄은 linear_regression.fit 함수를 이용해 모델을 학습하게 하는 함수입니다. 앞에서 언급 했듯이 X값은 꼭 2차원 형태로 입력해야 하고 y값은 기존형태로 입력하면 됩니다. X값엔 data 변수 안에 있는 “RM”데이터 y값엔 “Price”데이터를 입력했습니다.

세 번째 줄은 linear_regression.predict함수를 통해 학습된 선형회귀모델에 “RM”값을 입력 값으로 해서 y값을 예측합니다. 예측된 y값은 prediction 변수에 저장됩니다. 네 번째 줄은 linear_regression.intercept_를 통해 선형회귀모델의 a계수를 출력합니다. 다섯 번째 줄은 linear_regression.coef_를 통해 선형회귀모델의 b계수를 출력합니다. 그림 18은 데이터 학습에 대한 내용입니다.

```
1 linear_regression = linear_model.LinearRegression()
2 linear_regression.fit(X = pd.DataFrame(data_frame["RM"]), y = data_frame["Price"])
3 prediction = linear_regression.predict(X = pd.DataFrame(data_frame["RM"]))
4 print('a value = ', linear_regression.intercept_)
5 print('b balue =', linear_regression.coef_)
```

```
a value =  -34.6706207764
b balue = [ 9.10210898]
```

그림 18 데이터 학습

1.6.6 적합도 검증

다음은 잔차를 구하도록 하겠습니다. 첫 번째 줄은 잔차를 구하는 공식인 실제 값 “Price”에서 prediction에 저장된 예측 값 y를 빼주어 residuals 변수에 저장합니다. 두 번째 줄은 decribe함수를 통해 다양한 요약 통계를 생성합니다. 그림 19는 적합도 검증 중 잔차에 대한 내용입니다.

```
1 residuals = data_frame["Price"] - prediction
2 residuals.describe()
```

```
count    5.060000e+02
mean     1.899227e-15
std      6.609606e+00
min     -2.334590e+01
25%     -2.547477e+00
50%      8.976267e-02
75%      2.985532e+00
max      3.943314e+01
Name: Price, dtype: float64
```

그림 19 적합도 검증 - 잔차

다음은 적합도 검증 방법 중 결정계수를 구하도록 하겠습니다. 첫 번째 줄에서 residuals에 저장된 잔차 값을 제곱한 값을 numpy.sum함수를 이용해 더해준 후 SSE 변수에 저장합니다. 두 번째 줄에서 “Price”에 저장된 실제 y값에서 numpy.mean 함수를 이용해 “Price”를 평균한 값을 빼고 제곱한 값을 numpy.sum함수를 이용해 더해준 후 SST변수에 저장합니다.

세 번째 줄에서 결정계수 값을 계산합니다. 결정계수를 구하는 식과 같이 SSE에서 SST를 나누고 1에서 빼준 값을 R_squared 변수에 저장합니다. 네 번째 줄에서 R_squared 변수에 저장된 결정계수 값을 출력합니다.
출력 된 결정계수 48.35%로 결과를 통해 x값이 y값에 영향을 주는 것을 확인했습니다. 이것은 낮은 수치인 것 같지만 실제로 13개의 독립변수 중 1개인 것을 감안한다면 매우 높은 수치인 것을 확인할 수 있습니다. 그림 20은 적합도 검증 중 결정계수에 대한 내용입니다.

```
SSE = (residuals**2).sum()
SST = ((data_frame["Price"]-data_frame["Price"].mean())**2).sum()
R_squared = 1 - (SSE/SST)
print('R_squared = ', R_squared)
```

```
R_squared =  0.48352545599133423
```

그림 20 적합도 검증 - 결정계수

1.6.7 예측하여 플롯으로 표현

다음은 예측한 값을 이용해 산점도에 선형회귀선을 그리도록 하겠습니다. 첫 번째 줄에선 plot함수를 통해 산점도를 그렸습니다. 다섯 번째 줄은 산점도 위에 선형회귀선을 그립니다. 출력된 결과를 보면 구하고자 했던 선형회귀선이 그려진 것을 확인할 수 있습니다. 그림 21은 선형회귀선 및 산점도 표현에 대한 내용입니다.

```
data_frame.plot(kind="scatter",x="RM",y="Price",figsize=(6,6),
                color="black", xlim = (4,8), ylim = (10,45))

# Plot regression line
plt.plot(data_frame["RM"],prediction,color="blue")
```

```
[<matplotlib.lines.Line2D at 0xcf76c18>]
```

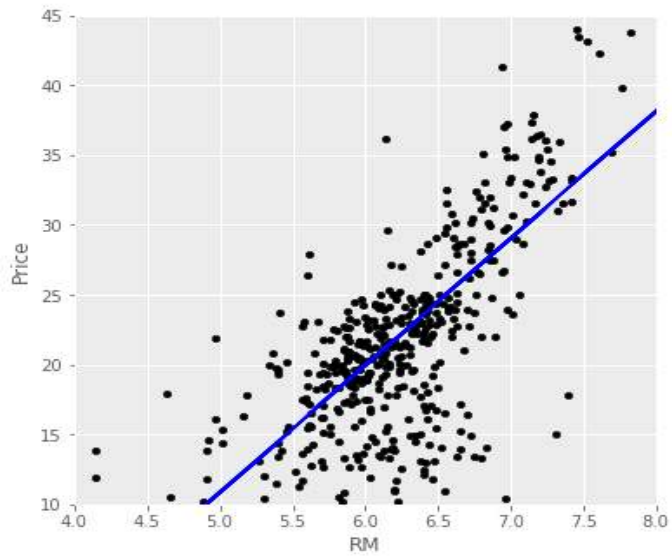


그림 21 선형회귀선 및 산점도 표현

1.6.8 성능 평가

다음은 생성된 회귀분석모델에 대한 성능평가를 진행하도록 하겠습니다. 첫 번째 줄은 score 함수로 예측한 결과 값과 정확한 결과 값을 비교해서 성

능을 평가합니다. 파라미터를 살펴보면 독립변수 “RM”를 2차원 DataFrame 형태로 교체 후 X값으로 지정해주고 종속변수 Price”값은 y값으로 지정해서 학습한 모델을 통해 성능을 평가합니다.

두 번째 줄은 sklearn 패키지에서 제공하는 mean_squared_error모듈을 이용해 평균제곱오차 값을 구합니다. 해당 모듈에 파라미터 값은 학습한 모델을 통해 나온 예측 값 prediction 변수와 실제 값이 저장된 “Price”값을 입력합니다. 세 번째 줄은 RMSE 값을 구합니다. 구하는 방법은 세 번째 줄에서 구한 평균제곱오차 값에 루트를 씌워주었습니다. 루트는 **0.5 수식으로 구하였습니다. 이제 결과를 확인해 보도록 하겠습니다. 그림 22는 성능평가에 대한 내용입니다.

```
1 print('score = ', linear_regression.score(X = pd.DataFrame(data_frame["RM"]), y = data_frame["Price"]))
2 print('Mean_Squared_Error = ', mean_squared_error(prediction, data_frame["Price"]))
3 print('RMSE = ', mean_squared_error(prediction, data_frame["Price"])**0.5)
```

```
score =  0.483525455991
Mean_Squared_Error =  43.6005517712
RMSE =  6.60307138922
```

그림 22 성능평가

보시면 아시겠지만 앞의 실습 1에서 확인했던 수치보다 낮은 것을 확인할 수 있습니다. 또한 RMSE 결과도 기존 실습 1 보다 오차가 높은 것을 확인했습니다. 지금까지 sklearn패키지를 사용해 단일선형회귀분석 하는 방법을 배웠습니다.

2. 다중선형회귀분석

2.1 다중선형회귀분석이란?

2.1.1 다중선형회귀분석 소개

- 두 개 이상의 독립변수들과 하나의 종속변수의 관계를 분석하는 방법
- 단순회귀를 확장한 것

2.1.2 다중선형회귀분석 가정

- 오차항 ϵ_i는 n개의 독립변수 x_1,x_2,x_3,...,x_n의 각각의 독립적
- 독립변수와 관려된 측정오차는 존재하지 않음
- 오차항의 기대 값은 0이며 일정한 분산을 갖는 정규분포를 이룸
- 어떤 두 오차도 서로 상관이 없다. 즉, 그들의 공분산은 0
- 독립변수들은 서로 선형함수로 완전히 관련되어 있지 않음

다중선형회귀분석은 두 개 이상의 독립변수들과 하나의 종속변수의 관계를 분석하는 방법입니다. 여기서 단일선형회귀분석과 다른 점을 인지하셨나요? 네. 바로 두 개 이상의 독립변수입니다. 이전 강의에서 단일선형회귀분석은 하나의 독립변수와 하나의 종속변수의 관계를 분석하였습니다.

다중선형회귀분석은 단일선형회귀분석을 확장한 것이라고 생각하시면 쉽습니다. 단일선형회귀분석에서와 같이 다중선형회귀분석에서도 가정을 하고 회귀분석을 진행합니다. 첫 번째로 오차항은 각각의 독립변수에 독립적이다라는 가정을 합니다. 즉 독립변수와 관련된 측정오차는 존재하지 않는다는 얘기입니다. 두 번째 가정은 오차항의 기대 값은 0이라는 것입니다. 또한 일정한 분산을 갖는 정규분포를 이룬다는 가정을 합니다.

세 번째 가정은 어떤 두 오차도 서로 상관이 없다는 가정입니다. 즉, 각각의 오차에 대한 공분산 값은 0이라는 가정입니다. 마지막으로 각각의 독립변수들은 서로 선형함수로 완전히 관련되어 있지 않다는 가정을 합니다. 물론 이것보다 더욱 많은 가정이 존재하지만 대표적인 가정에 대하여 살펴봤습니다.

2.1.3 다중선형회귀모델 식

$$y_i = \alpha + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_{k-1} x_{k-1i} + \beta_k x_{ki} + \varepsilon_i$$

- 위 식은 x라는 독립변수들이 y라는 종속변수에 주는 영향력을 식으로 나타낸 것
- k : 1~k까지에 상수 값(여러 개 독립변수 중 n번째 독립변수를 뜻함)
- y_i : i번째 관측치에 대한 종속변수의 값
- x_i : 이미알려진 독립변수의 i번째 값
- α : X값이 변해도 Y의 변동에는 영향을 주지 않는 회귀 계수
- β : X의 영향력을 크기와 부호로 나타내 주는 회귀 계수, 독립변수 X의 기울기
- ϵ_i : i번째 관측치에 대한 오차항

자 이제 다중선형회귀모델 식에 대해서 알아보도록 하겠습니다. 식을 확인하시면 단일선형회귀모델 식은 간단했는데 복잡해지셨지요? 하지만 간단하게 생각하시면 됩니다. 독립변수 x값이 늘어남에 따라 각 독립변수에 베타값도 해당 개수만큼 늘어났다고 보시면 간단합니다. 물론 알파 값과 오차항 값은 하나입니다. 자 이제부턴 수식을 통해 다중선형회귀분석식의 회귀 계수를 추정하는 방법을 알아보도록 하겠습니다.

2.1.4 회귀 계수 추정

- 수집된 데이터에 가장 적절한 회귀 직선을 구하는 것
- 최소자승법 사용

회귀 계수를 추정한다는 것은 독립변수 X와 종속변수 Y에 대해 가장 적절한 회귀 직선을 구하는 것입니다. 회귀직선을 구하는 이유는 임의의 독립변수 X값이 주어졌을 때 종속변수 Y값을 추정하기 위해서입니다. 이전 단일선형회귀분석에서 추정한 회귀 계수는 a 와 b 값이었습니다. 그렇지만 다중선형회귀분석에선 회귀계수 a와 독립변수개수 만큼의 회귀계수 b값을 추정합니다.

2.1.5 최소자승법

잔차를 자승한 값들의 합이 최소가 되도록 표본회귀식의 a와 b를 구하는 방법

$$\sum e_i^2 = \sum (Y_i - \hat{Y}_i)^2 = \sum (Y_i - a - b_1 X_{1i} - b_2 X_{2i})^2$$

- 잔차(Residual)

독립변수 X의 값이 주어질 때 표본회귀선의 예측 값 $\hat{Y}$와 실제 값 Y_i 사이에 표본오차 때문에 발생하는 차이

$$e_i = Y_i - \hat{Y}$$

최소자승법은 잔차를 자승한 값들의 합이 최소가 되도록 하는 표본회귀식의 a와 b를 구하는 방법입니다. 식에서 달라진 부분을 확인해보면 b와 x가 추가된 것을 알 수 있습니다. 식에선 두 가지 독립변수를 가정했기 때문에 b가 두 개이지만 독립변수가 n개만큼 추가되면 최소자승법의 b의 개수도 n개만큼 증가합니다.

2.1.6 표본회귀계수 구하는 예

자 그럼 이제 예시를 통해 표본회귀 계수를 구해보도록 하겠습니다. 먼저 예시에 다룰 데이터를 살펴보도록 하겠습니다. 데이터는 차량가격과 광고비, 판매액 총 세 개의 변수가 있습니다.

변수 3개 중 차량가격과 광고비는 판매액과의 관계를 알기 위해 사용되는 독립변수입니다. 표 7은 표본회귀계수 예시 데이터입니다. 표 8은 표본회귀계수를 구하는 예입니다.

차종	차량 가격(x_{1i})	광고비(x_{2i})	판매액(y)
A	13	9	20
B	18	7	22
C	17	17	30
D	20	11	27
E	22	8	35
F	21	10	32

표 7 표본회귀계수 예시 데이터

x_{1i}	x_{2i}	y_i	x_{1i}^*	x_{2i}^*	y_i^*	x_{1i}^{*2}	x_{2i}^{*2}	$x_{1i}{}^* x_{2i}{}^*$	$x_{1i}{}^* y_i{}^*$	$x_{2i}{}^* y_i{}^*$
13	9	20	-5.5	-1.33	-7.67	30.25	1.78	7.33	42.17	10.22
18	7	22	-0.5	-3.33	-5.67	0.25	11.11	1.67	2.83	18.89
17	17	30	-1.5	6.67	2.33	2.25	44.44	-1	-3.5	15.56
20	11	27	1.5	0.67	-0.67	2.25	0.44	1	-1	-0.44
22	8	35	3.5	-2.33	7.33	12.25	5.44	-8.17	25.67	-17.11
21	10	32	2.5	-0.33	4.33	6.25	0.11	-0.83	10.83	-1.44
111	62	166	0	0	0	53.5	63.33	-9	77	25.67
$\overline{x_1}=18.5,\quad \overline{x_2}=10.33,\quad \overline{y}=27.67 x_{1i}{}^* = x_{1i} - \overline{x_1},\quad x_{2i}{}^* = x_{2i} - \overline{x_2},\quad y_i{}^* = y_i - \overline{y}$										

표 8 표본회귀계수를 구하는 예

$$b_1 = \frac{\sum x_{2i}{}^{*2} \sum y_i{}^* x_{1i}{}^* - \sum x_{1i}{}^* x_{2i}{}^* \sum y_i{}^* x_{2i}{}^*}{\sum x_{1i}{}^{*2} \sum x_{2i}{}^{*2} - (\sum x_{1i}{}^* x_{2i}{}^*)^2} = \frac{63.33(77) - (-9)(25.67)}{53.5(63.33) - (-9)^2} = 1.5444$$

$$b_2 = \frac{\sum x_{1i}{}^{*2} \sum y_i{}^* x_{2i}{}^* - \sum x_{1i}{}^* x_{2i}{}^* \sum y_i{}^* x_{1i}{}^*}{\sum x_{1i}{}^{*2} \sum x_{2i}{}^{*2} - (\sum x_{1i}{}^* x_{2i}{}^*)^2} = \frac{53.5(25.67) - (-9)(77)}{53.5(63.33) - (-9)^2} = 0.6248$$

$$a = \bar{y} - b_1 \bar{x}_1 - b_2 \bar{x}_2 = 27.67 - 1.5444(18.5) - 0.6248(10.33) = -7.3537$$

$$\hat{y}_i = -7.3537 + 1.5444 x_{1i} + 0.6248 x_{2i}$$

수식을 확인하시면 별표 모양이 있는 것을 확인할 수 있습니다. 여기서 별표모양은 해당 변수의 값과 평균값을 빼서 구한 편차 값입니다. 식을 통해 나오는 값은 각 독립변수에 대한 b값과 a값이 나오는 것을 확인할 수 있습니다. 결과물로 나온 표본회귀계수 a 값-7.3537과 b1 값 1.5444 그리고 b2 값 0.6248을 표본회귀선 식에 대입하면 마지막 방정식과 같이 y= -7.3537 + 1.5444x1 + 0.6248x2이 구하는 표본회귀 방정식입니다.

2.2 적합도 검증

2.2.1 적합도 검증이란?

- 표본자료를 사용하여 구한 표본회귀식이 종속변수의 값을 어느 정도 정확하게 예측할 수 있는가의 정도를 검증
- 두 개 이상의 변수 값들이 표본회귀선 주위에 몰려 있으면 종속변수의 실제 값과 예측 갑 차이인 잔차가 줄어들어 예측의 정확성이 높아짐

2.2.2 적합도 검증방법

- 추정의 표준오차
- 결정계수

자 이제는 적합도 검증 부분은 앞의 단일선형회귀분석 부분과 유사하니 생략하도록 하겠습니다. 자 이제 학습한 내용과 관련된 코드를 살펴보겠습니

다. 코드의 내용은 대부분 앞의 단일선형회귀분석 코드와 유사하므로 중복된 부분에 대한 설명은 생략하겠습니다. 코드는 먼저 간단한 데이터를 생성해 다중선형회귀분석을 하고 다음으론 실제 데이터를 사용해 다중선형회귀분석을 해보는 순서로 진행됩니다.

2.3 다중선형회귀분석 실습 - Basic 1

2.3.1 Python package 로드 및 matplotlib 출력 옵션 설정

먼저, 첫 번째 줄부터 일곱 번째 줄까지는 사용할 패키지들을 불러와서 포함시키는 코드입니다. 그림 23은 python package들을 가져오는 것과 matplotlib 출력 옵션 설정에 대한 내용입니다.

```
1 from sklearn import linear_model
2 import numpy as np
3 import pandas as pd
4 import matplotlib
5 import matplotlib.pyplot as plt
6 %matplotlib inline
7 matplotlib.style.use('ggplot')
```

그림 23 python package 로드 및 matplotlib 출력 옵션 설정

2.3.2 데이터 생성

첫 번째 줄은 data 변수 안에 dictionary형태의 x1, x2, y key이름을 가진 데이터를 생성합니다. 네 번째 줄은 data변수 안에 저장된 데이터를 pandas.DataFrame 함수를 통해 2차원의 수정 가능한 테이블 형태의 구조로 변경 후 data 변수에 저장합니다. 다섯 번째 줄은 독립변수들을 따로 변수에 저장합니다. “data”데이터프레임 안에 독립변수 “x1”과 “x2”를 “X”라는 변수에 저장하는 겁니다.

여섯 번째 줄은 종속변수를 따로 변수에 저장합니다. “data”데이터프레임 안에 종속변수 “y”를 변수 y에 저장하는 겁니다. 일곱 번째 줄은 데이터프레임을 출력합니다. x1, x2, y 컬럼 명을 가진 데이터프레임이 생성된 것을 확인할 수 있습니다. 그림 24는 데이터 생성에 대한 내용입니다.

```
data = {'x1' : [13, 18, 17, 20, 22, 21],
        'x2' : [9, 7, 17, 11, 8, 10],
        'y' : [20, 22, 30, 27, 35, 32]}
data = pd.DataFrame(data)
X = data[['x1', 'x2']]
y = data['y']
data
```

	x1	x2	y
0	13	9	20
1	18	7	22
2	17	17	30
3	20	11	27
4	22	8	35
5	21	10	32

그림 24 데이터 생성

2.3.3 데이터 학습

다음은 앞의 단일선형회귀모델 실습에서 보신 내용과 동일한 코드를 이용하여 학습을 진행합니다. 출력된 결과를 확인하면 하나의 a 값과 두 개의 b 값이 출력된 것을 확인할 수 있습니다. 앞에서 얘기했듯이 b값은 독립변수 개수만큼 생성되기 때문에 2개입니다. 그림 25는 데이터 학습에 대한 내용입니다.

```
linear_regression = linear_model.LinearRegression()
linear_regression.fit(X = pd.DataFrame(X), y = y)
prediction = linear_regression.predict(X = pd.DataFrame(X))
print('a value = ', linear_regression.intercept_)
print('b balue = ', linear_regression.coef_)
```

```
a value =  -7.35920177384
b balue = [ 1.5443459   0.62472284]
```

그림 25 데이터 학습

2.3.4 적합도 검증

다음은 결정계수를 구해봅니다. 앞의 단일선형회귀 코드와 동일한 코드를 이용합니다. 출력된 결정계수는 79.6%입니다. 즉, 독립변수들이 종속변수에 상당한 영향을 주는 것을 확인했습니다. 그림 26은 적합도 검증 중 잔차이고 그림 27은 적합도 검증 중 결정계수에 대한 내용입니다.

```
residuals = y-prediction
residuals.describe()
```

```
count    6.000000e+00
mean    -2.368476e-15
std      2.622371e+00
min     -3.399667e+00
25%     -1.987805e+00
50%      5.828714e-01
75%      1.415327e+00
max      3.385809e+00
Name: y, dtype: float64
```

그림 26 적합도 검증 - 잔차

```
SSE = (residuals**2).sum()
SST = ((y-y.mean())**2).sum()
R_squared = 1 - (SSE/SST)
print('R_squared = ', R_squared)
```

```
R_squared =  0.7969440176685 23
```

그림 27 적합도 검증 - 결정계수

2.3.5 성능 평가

다음은 앞의 단일선형회귀 코드와 유사한 방법으로, 생성된 회귀분석모델에 대한 결정계수, MSE, RMSE를 구해봅니다. RMSE가 2.39 정도로 오차도 작은 것을 확인했습니다. 그림 28은 성능평가에 대한 내용입니다.

```
from sklearn.metrics import mean_squared_error
print('score = ', linear_regression.score(X = pd.DataFrame(X), y=y))
print('Mean_Squared_Error = ', mean_squared_error(prediction, y))
print('RMSE = ', mean_squared_error(prediction, y)**0.5)
```

```
score =  0.796944017669
Mean_Squared_Error =  5.73069105691
RMSE =  2.39388618295
```

그림 28 성능 평가

지금까지 임의의 데이터를 사용하여 간단한 실습을 해보았습니다. 이번엔 앞의 단일 회귀분석에서 사용한 보스턴의 집값 관련한 데이터를 이용하여 실습을 해보겠습니다.

2.4 다중선형회귀분석 실습 - Basic 2

2.4.1 Boston dataset 로드

다음은 x=pd.DataFrame()이라는 함수를 사용하여 보스턴 데이터 셋을 데이터프레임으로 변환합니다. 첫 번째 줄은 boston_house_price 변수에 전체 데이터 중 독립변수에 해당하는 값만 DataFrame형으로 변경 후 변수 X에 저장합니다. 출력된 결과를 확인하면 하나의 a 값과 열세 개의 b값이 출력된 것을 확인할 수 있습니다. 앞에서 얘기했듯이 b값은 독립변수 개수만큼 생성되기 때문에 13개입니다. 그림 29는 boston dataset을 로드하고 그림 30은 dataset의 정보를 확인합니다. 그림 31은 boston dataset을 데이터프레임으로 정제하는 내용입니다. 그림 32는 데이터프레임의 컬럼명 변경에 대한 내용입니다.

```
1 from sklearn import datasets
2 boston_house_prices = datasets.load_boston()
3 print(boston_house_prices.keys())
4 print(boston_house_prices.data.shape)
5 print(boston_house_prices.feature_names)
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR'])
(506, 13)
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

그림 29 boston dataset 로드

2.4.2 Boston dataset 정보 확인

```
1 print(boston_house_prices.DESCR)
```

```
Boston House Prices dataset
===========================

Notes
------
Data Set Characteristics:

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive

    :Median Value (attribute 14) is usually the target

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's
```

그림 30 boston dataset 정보 확인

2.4.3 Boston dataset을 데이터프레임으로 정제

```
X = pd.DataFrame(boston_house_prices.data)
X.tail()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

그림 31 boston dataset 데이터프레임으로 정제

```
X.columns = boston_house_prices.feature_names
X.tail()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

그림 32 데이터프레임의 컬럼명 변경

2.4.4 데이터 학습

다음은 잔차를 구하도록 하겠습니다. 첫 번째 줄은 잔차를 구하는 공식인 실제 값 “y”에서 prediction에 저장된 예측 값 y를 빼주어 residuals 변수에 저장합니다. 두 번째 줄은 decribe함수를 통해 다양한 요약 통계를 생성합니다. 잔차의 표준편차 std는 약 4정도 되는 것을 알 수 있습니다. 그림 33은 데이터를 학습시켜 회귀계수 a와 b를 출력한 내용입니다. 그림 34는 적합도 검증 중 잔차에 대한 내용입니다.

```
linear_regression = linear_model.LinearRegression()
linear_regression.fit(X = pd.DataFrame(X), y = y)
prediction = linear_regression.predict(X = pd.DataFrame(X))
print('a value = ', linear_regression.intercept_)
print('b balue =', linear_regression.coef_)
```

```
a value =  36.4911032804
b balue = [ -1.07170557e-01   4.63952195e-02   2.08602395e-02   2.68856140e+00
  -1.77957587e+01   3.80475246e+00   7.51061703e-04  -1.47575880e+00
   3.05655038e-01  -1.23293463e-02  -9.53463555e-01   9.39251272e-03
  -5.25466633e-01]
```

그림 33 회귀계수 출력

2.4.5 적합도 검증

```
residuals = y-prediction
residuals.describe()
```

```
count    5.060000e+02
mean     7.698716e-15
std      4.684137e+00
min     -1.557946e+01
25%     -2.725635e+00
50%     -5.164656e-01
75%      1.783116e+00
max      2.618865e+01
Name: Price, dtype: float64
```

그림 34 적합도 검증 - 잔차

다음은 적합도 검증 방법 중 결정계수를 구하도록 하겠습니다. 다음은 적합도 검증 방법 중 결정계수를 구하도록 하겠습니다. 단일 회귀 분석의 79% 보다는 조금 떨어지는 74% 정도 되는 것을 볼 수 있습니다. 그림 35는 적합도 검증 중 결정계수에 대한 내용입니다.

```
SSE = (residuals**2).sum()
SST = ((y-y.mean())**2).sum()
R_squared = 1 - (SSE/SST)
print('R_squared = ', R_squared)
```

```
R_squared =  0.7406077428649429
```

그림 35 적합도 검증 - 결정계수

2.4.6 성능 평가

다음은 생성된 회귀분석모델에 대한 성능평가를 진행합니다. 이제 결과를 확인해 보도록 하겠습니다. 이전 단일 회귀 분석에서의 RMSE는 6.6이었습니다. 방금 구한 다중회귀분석식의 RMSE는 4.68로 적합도가 많이 높아진 것을 알 수 있습니다. 그림 36은 성능평가에 대한 내용입니다.

```
print('score = ', linear_regression.score(X = pd.DataFrame(X), y = y))
print('Mean_Squared_Error = ', mean_squared_error(prediction, y))
print('RMSE = ', mean_squared_error(prediction, y)**0.5)
```

```
score =  0.740607742865
Mean_Squared_Error =  21.8977792177
RMSE =  4.67950630064
```

그림 36 성능 평가