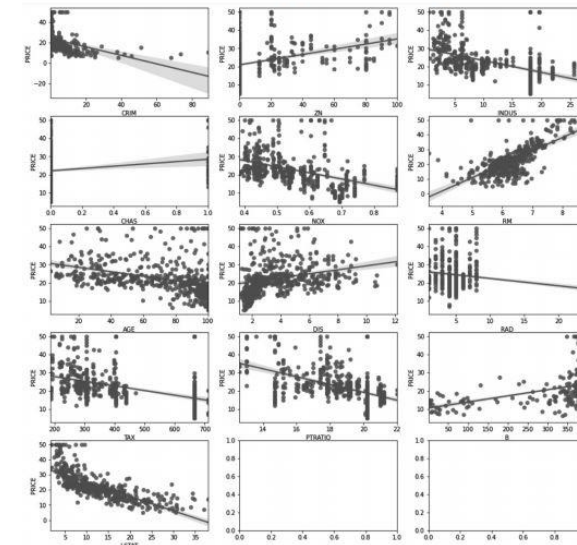


머신러닝 실습 — 수치 예측 (보스톤 주택가격)

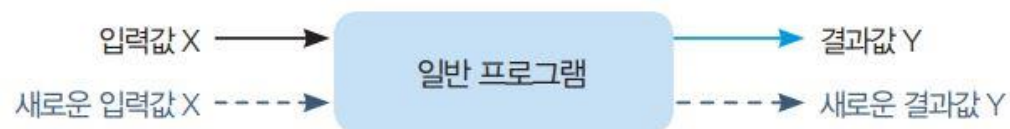


환경에 따른 주택 가격 예측하기	
목표	보스턴 주택 가격 데이터에 머신러닝 기반의 회귀 분석을 수행하여 주택 가격에 영향을 미치는 환경 변수를 확인하고, 그에 따른 주택 가격을 예측한다.
핵심 개념	머신러닝, 머신러닝 프로세스, 지도 학습, 사이킷런, 사이킷런의 내장 데이터셋, 분석 평가 지표
데이터 수집	보스턴 주택 가격 데이터: 사이킷런 내장 데이터셋
데이터 준비 및 탐색	1. 사이킷런 데이터셋 확인: <code>boston.DESCR</code> 2. 사이킷런 데이터셋에 지정된 X 피처와 타깃 피처 결합
분석 모델 구축	사이킷런의 선형 회귀 모델 구축
결과 시각화	

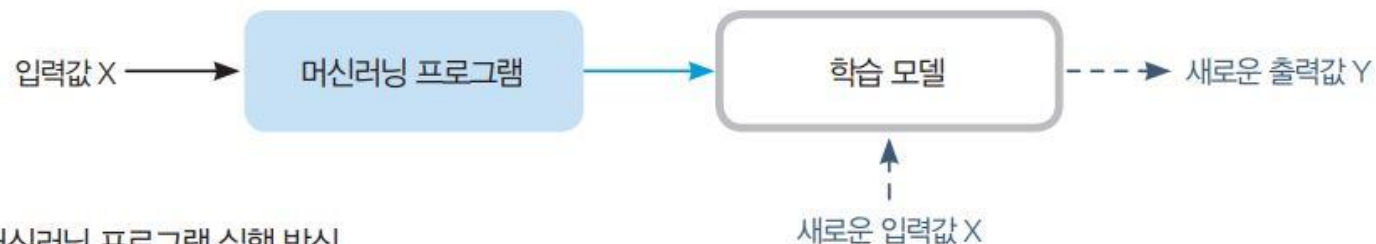
데이터가 주택 가격에 미치는 영향을 산점도와 선형 회귀 그래프로 시각화



보스턴 주택 가격 데이터에 머신러닝 기반의 회귀 분석을 수행
주택 가격에 영향을 미치는 변수를 확인하고 그 값에 따른 주택 가격을 예측



(a) 일반 프로그램 실행 방식



(b) 머신러닝 프로그램 실행 방식

그림 10-1 일반 프로그램과 머신러닝 프로그램 실행 방식 비교

- 핵심 개념 이해

- 머신러닝 프로세스

- 데이터 수집 → 데이터 전처리 및 훈련/테스트 데이터 분할 → 모델 구축 및 학습 → 모델 평가 → 예측

- 지도 학습

- 학습을 하기 위한 훈련 데이터에 입력과 출력을 같이 제공
 - 문제(입력)에 대한 답(출력, 결과값)을 아는 상태에서 학습하는 방식
 - 입력: 예측 변수, 속성, 특징
 - 출력: 반응 변수, 목표 변수, 클래스, 레이블

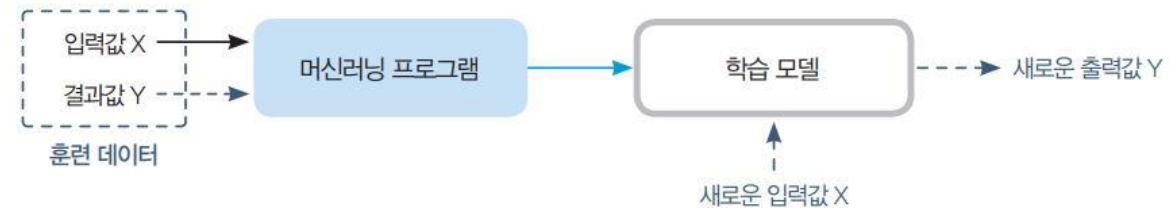


그림 10-2 머신러닝의 지도 학습 방식

- 사이킷런

- 파이썬으로 머신러닝을 수행하기 위한 쉽고 효율적인 개발 라이브러리를 제공
 - 보스턴 주택 가격 데이터, 붓꽃 데이터 등과 같은 머신러닝 분석용 데이터셋 을 제공
 - 전체 n개의 컬럼 중 앞에서 (n-1)개의 컬럼은 독립 변수 X를 의미
 - 마지막 컬럼 은 종속 변수 Y이며, 데이터셋 객체의 target 배열로 관리

COLAB에서 '10장_주택가격분석'으로 노트북 페이지를 추가하고 입력

In [1]:	<pre>!pip install sklearn</pre>
In [2]:	<pre>import numpy as np import pandas as pd from sklearn.datasets import load_boston boston = load_boston()</pre>

선형회귀 실습

- 데이터 수집, 준비 및 탐색
 - 데이터가 이미 정리된 상태이므로 데이터셋 구성을 확인

In [3]:	print(boston.DESCR)														
In [4]:	boston_df = pd.DataFrame(boston.data, columns = boston.feature_names) boston_df.head()														
Out[4]:		CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	
	0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	
	1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	
	2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	
	3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	
	4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	
In [5]:	boston_df['PRICE'] = boston.target boston_df.head()														
Out[5]:		CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
	0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
	1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
	2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
	3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
	4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

In [3]: 데이터셋에 대한 설명 `boston.DESCR` 을 확인

In [4]: 데이터셋 객체의 `data` 배열 `boston.data`, 즉 독립 변수 X가 되는 피쳐들을 `DataFrame` 자료형으로 변환하여 `boston_df`를 생성
boston_df의 데이터 5개를 확인 `boston_df.head()`

In [5]: 데이터셋 객체의 `target` 배열 `boston.target`, 즉 종속 변수인 주택 가격('PRICE') 컬럼을 `boston_df`에 추가
boston_df의 데이터 5개를 확인 `boston_df.head()`

- 데이터 수집, 준비 및 탐색
 - 2. 데이터가 이미 정리된 상태이므로 데이터셋 구성을 확인

In [6]:	print('보스턴 주택 가격 데이터셋 크기: ', boston_df.shape)
Out[6]:	보스턴 주택 가격 데이터셋 크기: (506, 14)
In [7]:	boston_df.info()
Out[7]:	<class 'pandas.core.frame.DataFrame'> RangeIndex: 506 entries, 0 to 505 Data columns (total 14 columns): CRIM 506 non-null float64 ZN 506 non-null float64 INDUS 506 non-null float64 CHAS 506 non-null float64 NOX 506 non-null float64 RM 506 non-null float64 AGE 506 non-null float64 DIS 506 non-null float64 RAD 506 non-null float64 TAX 506 non-null float64 PTRATIO 506 non-null float64 B 506 non-null float64 LSTAT 506 non-null float64 PRICE 506 non-null float64 dtypes: float64(14) memory usage: 55.4KB

In [6]: 데이터셋의 형태 `boston_df.shape`, 즉 행의 개수(데이터 개수)와 열의 개수(변수 개수)를 확인
행의 개수가 506이므로 데이터가 506개 있으며, 열의 개수가 14이므로 변수가 14개 있음
변수 중에서 13개는 독립 변수 X가 되고, 마지막 변수 'PRICE'는 종속 변수 Y가 됨

In [7]: `boston_df`에 대한 정보를 확인 `boston_df.info()`

• 14개의 독립 변수(피처)의 의미

- CRIM: 지역별 범죄 발생률
- ZN: 25,000평방피트를 초과하는 거주 지역 비율
- INDUS: 비상업 지역의 넓이 비율
- CHAS: 찰스강의 더미변수(1은 강에 인접, 0은 강에 인접 아님)
- NOX: 일산화질소 농도
- RM: 거주할 수 있는 방 개수
- AGE: 1940년 이전에 건축된 주택 비율
- DIS: 5개 주요 고용센터까지 가중 거리
- RAD: 고속도로 접근 용이도
- TAX: 10,000달러당 재산세 비율
- PTRATIO: 지역의 교사와 학생 수 비율
- B: 지역의 흑인 거주 비율
- LSTAT: 하위 계층의 비율
- PRICE(MEDV): 본인 소유 주택 가격의 중앙값

선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화
 1. 선형 회귀를 이용해 분석 모델 구축하기
 1. 사이킷런의 선형 분석 모델 패키지 `sklearn.linear_model` 에서 선형 회귀 `LinearRegression` 를 이용하여 분석 모델을 구축

In [8]:	<pre>from sklearn.linear_model import LinearRegression from sklearn.model_selection import train_test_split from sklearn.metrics import mean_squared_error, r2_score</pre>
In [9]:	<pre>#X, Y 분할하기 Y = boston_df['PRICE'] X = boston_df.drop(['PRICE'], axis = 1, inplace = False)</pre>
In [10]:	<pre>#훈련용 데이터와 평가용 데이터 분할하기 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.3, random_state = 156)</pre>

In [8]: 사이킷런을 사용하여 머신러닝 회귀 분석을 하기 위한 `LinearRegression`과 데이터셋 분리 작업을 위한 `train_test_split`, 성능 측정을 위한 평가 지표인 `mean_squared_error`, `r2_score`를 임포트

In [9]: PRICE 피처를 회귀식의 종속 변수 Y로 설정하고 PRICE를 제외 `drop()`한 나머지 피처를 독립 변수 X로 설정

In [10]: X와 Y 데이터 506개를 학습 데이터와 평가 데이터로 7:3 비율로 분할 `test_size=0.3`

선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀를 이용해 분석 모델 구축하기

1. 사이킷런의 선형 분석 모델 패키지 `sklearn.linear_model` 에서 선형 회귀 `LinearRegression` 를 이용하여 분석 모델을 구축

In [11]:	#선형 회귀 분석 : 모델 생성 <code>lr = LinearRegression()</code>
In [12]:	#선형 회귀 분석 : 모델 훈련 <code>lr.fit(X_train, Y_train)</code>
Out[12]:	<code>LinearRegression()</code>
In [13]:	#선형 회귀 분석 : 평가 데이터에 대한 예측 수행 -> 예측 결과 <code>Y_predict</code> 구하기 <code>Y_predict = lr.predict(X_test)</code>

In [11]: 선형 회귀 분석 모델 객체 `lr`을 생성

In [12]: 학습 데이터 `X_train` 와 `Y_train` 를 가지고 학습을 수행 `fit()`.

In [13]: 평가 데이터 `X_test` 를 가지고 예측을 수행하여 `predict()` 예측값 `Y_predict` 를 구함

선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화
 1. 선형 회귀를 이용해 분석 모델 구축하기
 2. 선형 회귀 분석 모델을 평가 지표를 통해 평가하고 회귀 계수를 확인하여 피처의 영향을 분석

In [14]:	<pre>mse = mean_squared_error(Y_test, Y_predict) rmse = np.sqrt(mse) print('MSE : {0:.3f}, RMSE : {1:.3f}'.format(mse, rmse)) print('R^2(Variance score) : {0:.3f}'.format(r2_score(Y_test, Y_predict)))</pre>
Out[14]:	<pre>MSE : 17.297, RMSE : 4.159 R^2(Variance score) : 0.757</pre>
In [15]:	<pre>print('Y 절편 값: ', lr.intercept_) print('회귀 계수 값: ', np.round(lr.coef_, 1))</pre>
Out[15]:	<pre>Y 절편 값: 40.995595172164336 회귀 계수 값: [-0.1 0.1 0. 3. -19.8 3.4 0. -1.7 0.4 -0. -0.9 0. -0.6]</pre>

In [14]: 회귀 분석은 지도 학습이므로 평가 데이터 X에 대한 결과값 Y_{test} 를 이미 알고 있는 상태에서 평가 데이터 Y_{test} 와

In [13]에서 구한 예측 결과 $Y_{predict}$ 의 오차를 계산하여 모델을 평가. 평가 지표 MSE를 구하고 `mean_squared_error()`
구한 값의 제곱근을 계산하여 `np.sqrt(mse)` 평가 지표 RMSE를 구함 그리고 평가 지표 R2 을 구함 `r2_score()`

In [15]: 선형 회귀의 Y절편 `lr.intercept_`과 각 피처의 회귀 계수 `lr.coef_`를 확인

선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화
 - 선형 회귀를 이용해 분석 모델 구축하기
 - 선형 회귀 분석 모델을 평가 지표를 통해 평가하고 회귀 계수를 확인하여 피처의 영향을 분석

In [16]:	coef = pd.Series(data = np.round(lr.coef_, 2), index = X.columns) coef.sort_values(ascending = False)	
Out[16]:	RM 3.35 CHAS 3.05 RAD 0.36 ZN 0.07 INDUS 0.03 B 0.01 AGE 0.01 TAX -0.01 CRIM -0.11 LSTAT -0.57 PTRATIO -0.92 DIS -1.74 NOX -19.80 dtype: float64	

In [16]: 회귀 모델에서 구한 회귀 계수 값 `lr.coef_` 과 피처 이름 `X.columns` 을 묶어서 Series 자료 형으로 만들고, 회귀 계수 값을 기준으로 내림차순으로 정렬하여 `ascending=False` 확인 `sort_values()`

회귀 모델 결과를 토대로 보스톤 주택 가격에 대한 회귀식

$$Y_{\text{PRICE}} = -0.11X_{\text{CRIM}} + 0.07X_{\text{ZN}} + 0.03X_{\text{INDUS}} + 3.05X_{\text{CHAS}} - 19.80X_{\text{NOX}} + 3.35X_{\text{RM}} + 0.01X_{\text{AGE}} - 1.74X_{\text{DIS}} + 0.36X_{\text{RAD}} - 0.01X_{\text{TAX}} - 0.92X_{\text{PTRATIO}} + 0.01X_{\text{B}} - 0.57X_{\text{LSTAT}} + 41.00$$

선형회귀 실습

- 회귀 분석 결과를 산점도 + 선형 회귀 그래프로 시각화하기
- 2. 선형 회귀를 이용해 분석 모델 구축하기

In [17]:	<pre>import matplotlib.pyplot as plt import seaborn as sns</pre>
In [18]:	<pre>fig, axs = plt.subplots(figsize = (16, 16), ncols = 3, nrows = 5) x_features = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT'] for i, feature in enumerate(x_features): row = int(i/3) col = i%3 sns.regplot(x = feature, y = 'PRICE', data = boston_df, ax = axs[row][col])</pre>

In [17]: 시각화에 필요한 모듈을 임포트

In [18]: 독립 변수인 13개 피처와 종속 변수인 주택 가격, PRICE와의 회귀 관계를 보여주는 13개 그래프를 subplots()를

사용하여 5행 3열 구조로 모아서 나타냄

aborn의 regplot()은 산점도 그래프와 선형 회귀 그래프를 함께 그려줌

선형회귀 실습

- 회귀 분석 결과를 산점도 + 선형 회귀 그래프로 시각화하기
- 2. 선형 회귀를 이용해 분석 모델 구축하기

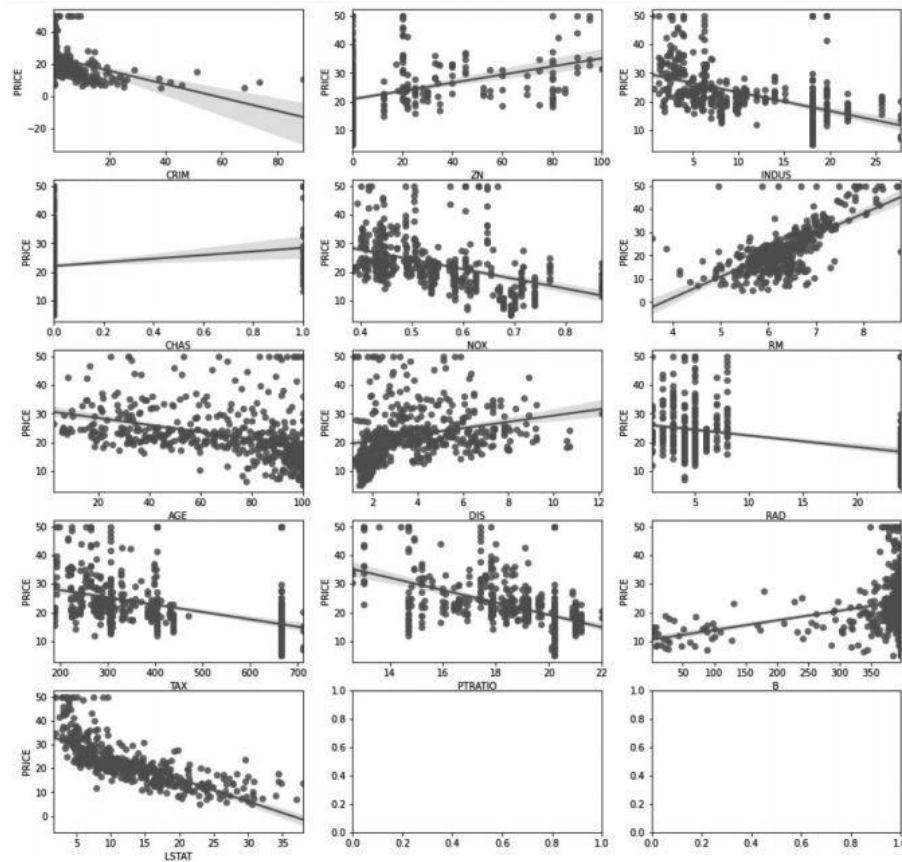


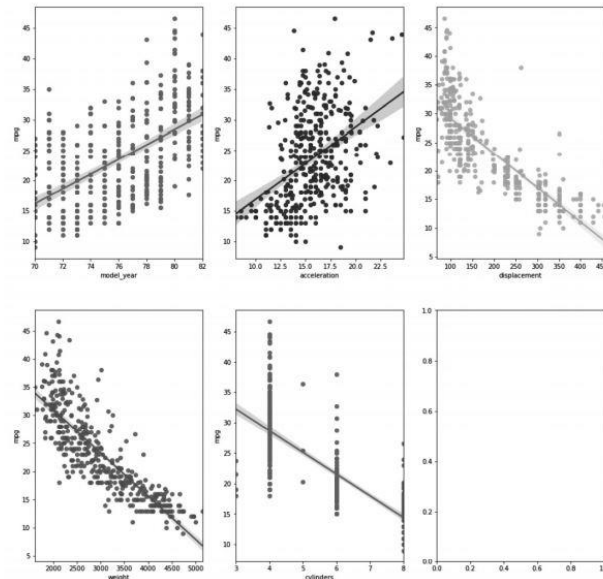
그림 10-3 13개 피처와 주택 가격의 회귀 관계를 나타낸 산점도/선형 회귀 그래프

머신러닝 실습 — 수치 예측 (자동차 연비 예측)



항목에 따른 자동차 연비 예측하기	
목표	자동차 연비 데이터에 머신러닝 기반의 회귀 분석을 수행하여 연비에 영향을 미치는 항목을 확인하고 그에 따른 자동차 연비를 예측한다.
핵심 개념	머신러닝, 머신러닝 프로세스, 지도 학습, 사이킷런, 사이킷런의 내장 데이터셋, 분석 평가 지표
데이터 수집	자동차 연비 데이터: UCI Machine Learning Repository에서 다운로드
데이터 준비 및 탐색	1. 필요 없는 컬럼 제거 2. X 변수와 Y 변수 확인
분석 모델 구축	사이킷런의 선형 회귀 모델 구축
결과 시각화	

X 변수와 Y 변수에 대한 산점도 그래프와 선형 회귀 그래프



- 목표설정
 - 목표: 자동차 연비 데이터에 머신러닝 기반의 회귀 분석을 수행
연비에 영향을 미치는 항목을 확인하고, 그에 따른 자동차 연비를 예측
- 핵심 개념 이해
 - 1절의 프로젝트와 동일한 개념에 대한 이해가 필요

선형회귀 실습

- 데이터 수집

1. 자동차 연비 데이터 다운로드하기

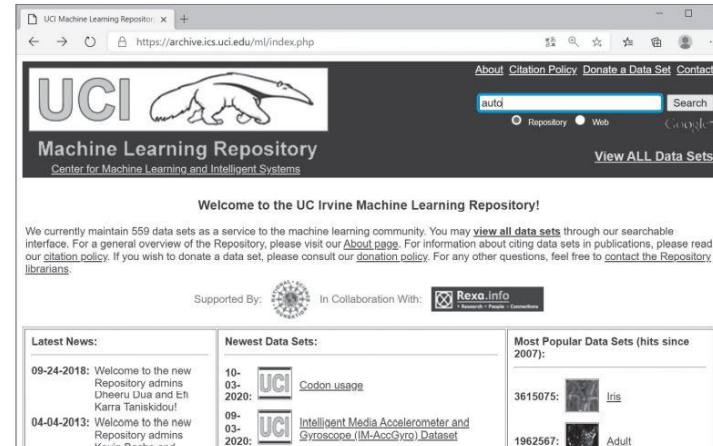


그림 10-4 UCI Machine Learning Repository 사이트에서 'auto' 검색

2. 검색 결과 목록에서 'Auto MPG Data Set - UCI Machine Learning Repository' 클릭

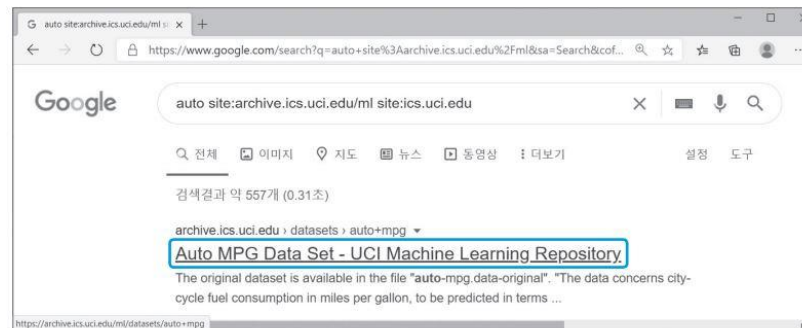


그림 10-5 검색 목록에서 다운로드할 데이터셋 선택

선형회귀 실습

• 데이터 수집

3. Data Folder를 클릭하여 'auto-mpg.data'를 다운로드

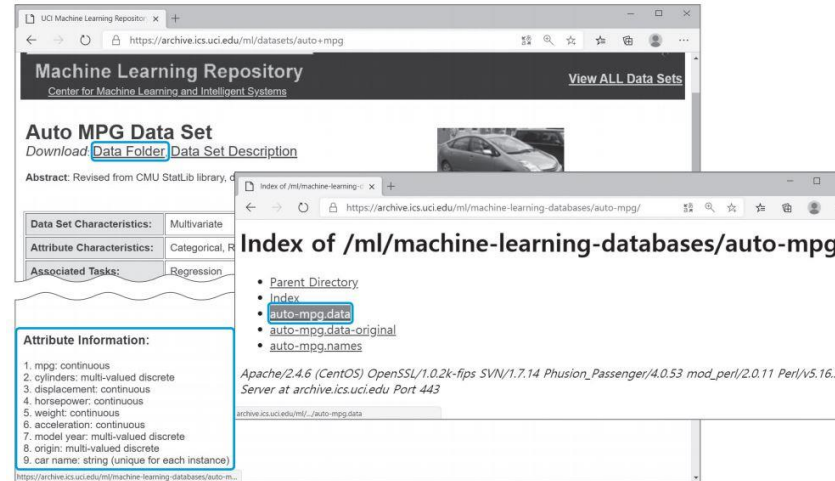


그림 10-6 데이터셋 다운로드

4. CSV 파일로 변경하기

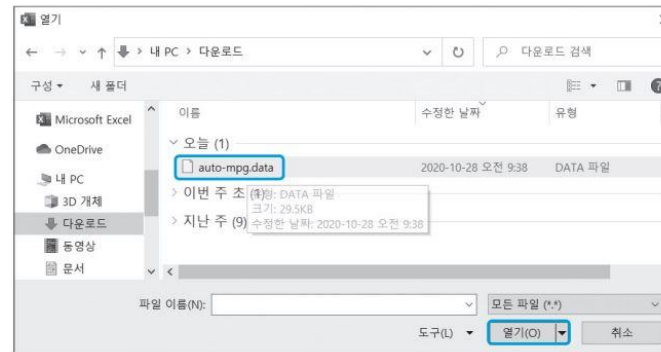
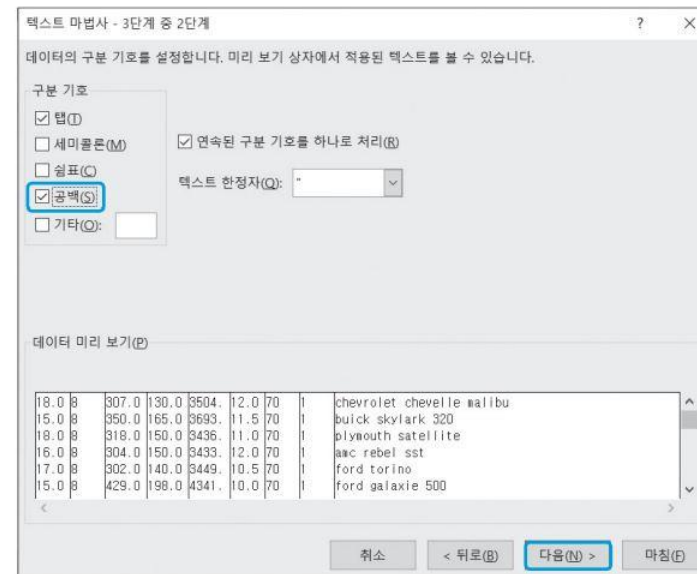


그림 10-7 CSV 파일로 변경하기 1 - 엑셀 프로그램에서 파일 열기

선형회귀 실습

- 데이터 수집

5. 1단계는 버튼을 클릭, 2단계에서는 [구분 기호]로 '공백'을 선택하고 버튼을 클릭



6. 텍스트 마법사 3단계에서 데이터 미리 보기를 확인하고 버튼을 클릭

그림 10-8 CSV 파일로 변경하기 2 - 텍스트 마법사로 파일 정리

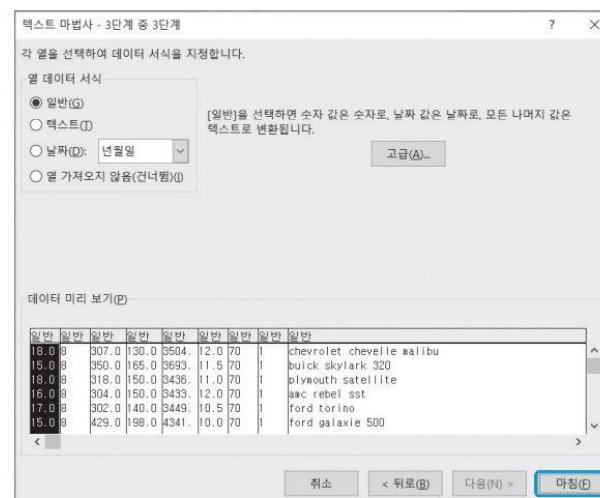


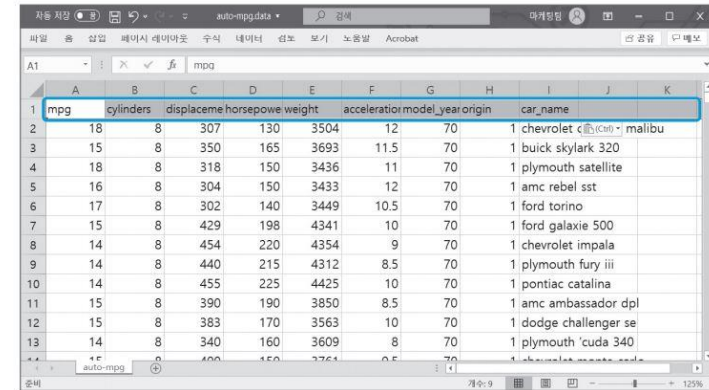
그림 10-9 CSV 파일로 변경하기 3 - 미리보기로 확인 후 텍스트 마법사 종료

선형회귀 실습

• 데이터 수집

7. 항목을 구분하기 위해 열 이름을 추가

- 행을 삽입하고 열 이름으로 mpg, cylinders, displacement, horsepower, weight, acceleration, model_year, origin, car_name을 각각 입력



	A	B	C	D	E	F	G	H	I	J	K
1	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name		
2	18	8	307	130	3504	12	70	1	chevrolet	malibu	
3	15	8	350	165	3693	11.5	70	1	buick skylark	320	
4	18	8	318	150	3436	11	70	1	plymouth satellite		
5	16	8	304	150	3433	12	70	1	amc rebel sst		
6	17	8	302	140	3449	10.5	70	1	ford torino		
7	15	8	429	198	4341	10	70	1	ford galaxie 500		
8	14	8	454	220	4354	9	70	1	chevrolet impala		
9	14	8	440	215	4312	8.5	70	1	plymouth fury iii		
10	14	8	455	225	4425	10	70	1	pontiac catalina		
11	15	8	390	190	3850	8.5	70	1	amc ambassador dpl		
12	15	8	383	170	3563	10	70	1	dodge challenger se		
13	14	8	340	160	3609	8	70	1	plymouth cuda 340		

그림 10-10 CSV 파일로 변경하기 4 - 항목 이름 추가

8. My_Python 폴더에 10장_data 폴더를 만들고 파일을 'auto-mpg.csv'로 저장

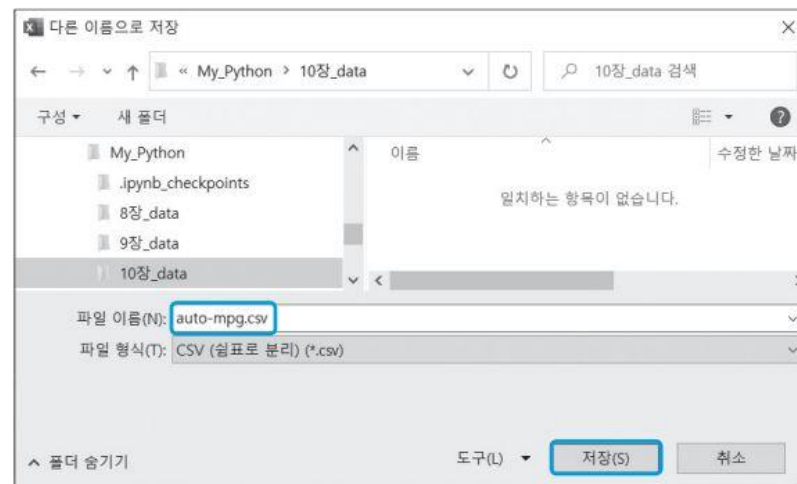


그림 10-11 CSV 파일로 변경하기 4 - CSV 파일 저장

선형회귀 실습

- 데이터 준비 및 탐색
 - 분석에 필요 없는 컬럼을 제거하고 데이터셋의 내용을 확인

In [1]:	<pre>import numpy as np import pandas as pd data_df = pd.read_csv('./10장_data/auto-mpg.csv', header = 0, engine = 'python')</pre>																																																												
In [2]:	<pre>print('데이터셋 크기: ', data_df.shape) data_df.head()</pre>																																																												
Out[2]:	<div>데이터셋 크기: (398, 9)</div> <table><tr><th></th><th>mpg</th><th>cylinders</th><th>displacement</th><th>horsepower</th><th>weight</th><th>acceleration</th><th>model_year</th><th>origin</th><th>car_name</th></tr><tr><td>0</td><td>18.0</td><td>8</td><td>307.0</td><td>130</td><td>3504</td><td>12.0</td><td>70</td><td>1</td><td>chevrolet chevelle malibu</td></tr><tr><td>1</td><td>15.0</td><td>8</td><td>350.0</td><td>165</td><td>3693</td><td>11.5</td><td>70</td><td>1</td><td>buick skylark 320</td></tr><tr><td>2</td><td>18.0</td><td>8</td><td>318.0</td><td>150</td><td>3436</td><td>11.0</td><td>70</td><td>1</td><td>plymouth satellite</td></tr><tr><td>3</td><td>16.0</td><td>8</td><td>304.0</td><td>150</td><td>3433</td><td>12.0</td><td>70</td><td>1</td><td>amc rebel sst</td></tr><tr><td>4</td><td>17.0</td><td>8</td><td>302.0</td><td>140</td><td>3449</td><td>10.5</td><td>70</td><td>1</td><td>ford torino</td></tr></table>		mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu	1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320	2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite	3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst	4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name																																																				
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu																																																				
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320																																																				
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite																																																				
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst																																																				
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino																																																				
In [3]:	<pre>data_df = data_df.drop(['car_name', 'origin', 'horsepower'], axis = 1, inplace = False) data_df.head()</pre>																																																												
Out[3]:	<table><tr><th></th><th>mpg</th><th>cylinders</th><th>displacement</th><th>weight</th><th>acceleration</th><th>model_year</th></tr><tr><td>0</td><td>18.0</td><td>8</td><td>307.0</td><td>3504</td><td>12.0</td><td>70</td></tr><tr><td>1</td><td>15.0</td><td>8</td><td>350.0</td><td>3693</td><td>11.5</td><td>70</td></tr><tr><td>2</td><td>18.0</td><td>8</td><td>318.0</td><td>3436</td><td>11.0</td><td>70</td></tr><tr><td>3</td><td>16.0</td><td>8</td><td>304.0</td><td>3433</td><td>12.0</td><td>70</td></tr><tr><td>4</td><td>17.0</td><td>8</td><td>302.0</td><td>3449</td><td>10.5</td><td>70</td></tr></table>		mpg	cylinders	displacement	weight	acceleration	model_year	0	18.0	8	307.0	3504	12.0	70	1	15.0	8	350.0	3693	11.5	70	2	18.0	8	318.0	3436	11.0	70	3	16.0	8	304.0	3433	12.0	70	4	17.0	8	302.0	3449	10.5	70																		
	mpg	cylinders	displacement	weight	acceleration	model_year																																																							
0	18.0	8	307.0	3504	12.0	70																																																							
1	15.0	8	350.0	3693	11.5	70																																																							
2	18.0	8	318.0	3436	11.0	70																																																							
3	16.0	8	304.0	3433	12.0	70																																																							
4	17.0	8	302.0	3449	10.5	70																																																							

In [2]: 데이터셋의 형태 `data_df.shape` 를 확인해보면, 398행과 9열로 구성되어 있음
398개 데이터에 9개 컬럼이 있으므로 파일 내용이 DataFrame으로 잘 저장되었다는 것을 알 수 있음
데이터 5개를 출력하여 내용을 확인 `data_df.head()` .

In [3] 피쳐 중에서 car_name, origin, horsepower는 분석에 사용하지 않으므로 제거 `data_df.drop()` 후 확인 `data_df.head()` .

선형회귀 실습

- 데이터 준비 및 탐색
 - 분석에 필요 없는 컬럼을 제거하고 데이터셋의 내용을 확인

In [4]:	print('데이터셋 크기: ', data_df.shape)
Out[4]:	데이터셋 크기: (398, 6)
In [5]:	data_df.info()
Out[5]:	<class 'pandas.core.frame.DataFrame'> RangeIndex: 398 entries, 0 to 397 Data columns (total 6 columns): mpg 398 non-null float64 cylinders 398 non-null int64 displacement 398 non-null float64 weight 398 non-null int64 acceleration 398 non-null float64 year 398 non-null int64 dtypes: float64(3), int64(3) memory usage: 18.7 KB

In [4]: 분석에 사용할 데이터셋의 형태 `data_df.shape` 를 확인

In [5]: 분석에 사용할 데이터셋의 정보 `data_df.info()` 를 확인

선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀 분석 모델 구축하기

1. 자동차 연비 예측을 위해 다음과 같이 선형 회귀 분석 모델을 구축

In [6]:	<pre>from sklearn.linear_model import LinearRegression from sklearn.model_selection import train_test_split from sklearn.metrics import mean_squared_error, r2_score</pre>
In [7]:	<pre><i>#X, Y 분할하기</i> Y = data_df['mpg'] X = data_df.drop(['mpg'], axis = 1, inplace = False)</pre>
In [8]:	<pre><i>#훈련용 데이터와 평가용 데이터 분할하기</i> X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0)</pre>

In [6]: 사이킷런을 사용하여 머신러닝 선형 회귀 분석을 하기 위한 LinearRegression과 데이터셋 분리 작업을 위한 train_test_split, 성능 측정을 위한 평가 지표인 mean_squared_error, r2_score를 импорт

In [7]: 자동차 연비를 예측하는 것이 프로젝트의 목표이므로, mpg 피처를 회귀식의 종속 변수 Y로 설정하고, mpg를 제외한 나머지 피처를 독립 변수 X로 설정

In [8]: 데이터를 7:3 비율 test_size=0.3로 분할하여 train_test_split() 학습 데이터와 평가 데이터로 설정

선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀 분석 모델 구축하기

1. 자동차 연비 예측을 위해 다음과 같이 선형 회귀 분석 모델을 구축

In [9]:	<i>#선형 회귀 분석 : 모델 생성</i> <code>lr = LinearRegression()</code>
In [10]:	<i>#선형 회귀 분석 : 모델 훈련</i> <code>lr.fit(X_train, Y_train)</code>
Out[10]:	LinearRegression()
In [11]:	<i>#선형 회귀 분석 : 평가 데이터에 대한 예측 수행 -> 예측 결과 Y_predict 구하기</i> <code>Y_predict = lr.predict(X_test)</code>

In [9]: 선형 회귀 분석 모델 객체인 lr을 생성

In [10]: 학습 데이터 X_{train} 와 Y_{train} 를 가지고 학습을 수행 `fit()`

In [11]: 평가 데이터 X_{test} 로 예측을 수행하여 `predict()` 예측값 $Y_{predict}$ 를 구함

선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀 분석 모델 구축하기

2. 평가 지표를 통해 선형 회귀 분석 모델을 평가하고 회귀 계수를 확인하여 자동차 연비에 끼치는 피처의 영향을 분석

In [12]:	<pre>mse = mean_squared_error(Y_test, Y_predict) rmse = np.sqrt(mse) print('MSE : {0:.3f}, RMSE : {1:.3f}'.format(mse, rmse)) print('R^2(Variance score) : {0:.3f}'.format(r2_score(Y_test, Y_predict)))</pre>
Out[12]:	<pre>MSE : 12.278, RMSE : 3.504 R^2(Variance score) : 0.808</pre>
In [13]	<pre>print('Y 절편 값: ', np.round(lr.intercept_, 2)) print('회귀 계수 값: ', np.round(lr.coef_, 2))</pre>
Out[13]:	<pre>Y 절편 값: -17.55 회귀 계수 값: [-0.14 0.01 -0.01 0.2 0.76]</pre>

In [12]: 회귀 분석은 지도 학습이므로 평가 데이터 X에 대한 Y_{test} 를 이미 알고 있음

평가 데이터의 결과값 Y_{test} 과 예측 결과값 $Y_{predict}$ 의 오차를 계산하여 모델을 평가하는데, `mean_squared_error()`를 이용하여 평가 지표 MSE를 구하고 구한 값의 제곱근을 계산하여 평가 지표 RMSE를 구한다. 그리고 `r2_score()`를 이용하여 평가 지표 R2를 구함

In [13]: 선형 회귀의 Y절편 $lr.intercept_$ 과 각 피처의 회귀 계수 $lr.coef_$ 를 확인

선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀 분석 모델 구축하기

2. 평가 지표를 통해 선형 회귀 분석 모델을 평가하고 회귀 계수를 확인하여 자동차 연비에 끼치는 피처의 영향을 분석

In [14]:	<pre>coef = pd.Series(data = np.round(lr.coef_, 2), index = X.columns) coef.sort_values(ascending = False)</pre>
Out[14]:	<pre>model_year 0.76 acceleration 0.20 displacement 0.01 weight -0.01 cylinders -0.14 dtype: float64</pre>

In [14]: 회귀 모델에서 구한 회귀 계수 값 `lr.coef_` 과 피처 이름 `X.columns` 을 묶어서 Series 자료 형으로 만들고, 회귀 계수 값을 기준으로 내림차순 `ascending = False` 으로 정렬 `sort_values()` 하여 회귀 계수 값이 큰 항목을 확인

회귀 모델 결과로 자동차 연비를 예측하는 회귀식

$$Y_{\text{mpg}} = -0.14X_{\text{cylinders}} + 0.01X_{\text{displacement}} - 0.01X_{\text{weight}} + 0.20X_{\text{acceleration}} + 0.76X_{\text{model_year}} - 17.55$$

선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

- 2. 선형 회귀 분석 모델 구축하기

- 1. 평가 지표를 통해 선형 회귀 분석 모델을 평가하고 회귀 계수를 확인하여 자동차 연비에 끼치는 피처의 영향을 분석

In [15]:	<pre>import matplotlib.pyplot as plt import seaborn as sns</pre>
In [16]:	<pre>fig, axs = plt.subplots(figsize = (16, 16), ncols = 3, nrows = 2) x_features = ['model_year', 'acceleration', 'displacement', 'weight', 'cylinders'] plot_color = ['r', 'b', 'y', 'g', 'r'] for i, feature in enumerate(x_features): row = int(i/3) col = i%3 sns.regplot(x = feature, y = 'mpg', data = data_df, ax = axs[row][col], color = plot_color[i])</pre>

In [15]: 시각화에 필요한 모듈을 임포트

In [16]: subplots()를 사용하여 독립 변수인 5개 피처 `['model_year', 'acceleration', 'displacement', 'weight', 'cylinders']`와 종속 변수인 연비 mpg와의 회귀 관계를 보여주는 5개 그래프를 2행 3열 구조로 나타낸

선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

- 2. 선형 회귀 분석 모델 구축하기

- 1. 평가 지표를 통해 선형 회귀 분석 모델을 평가하고 회귀 계수를 확인하여 자동차 연비에 끼치는 피처의 영향을 분석

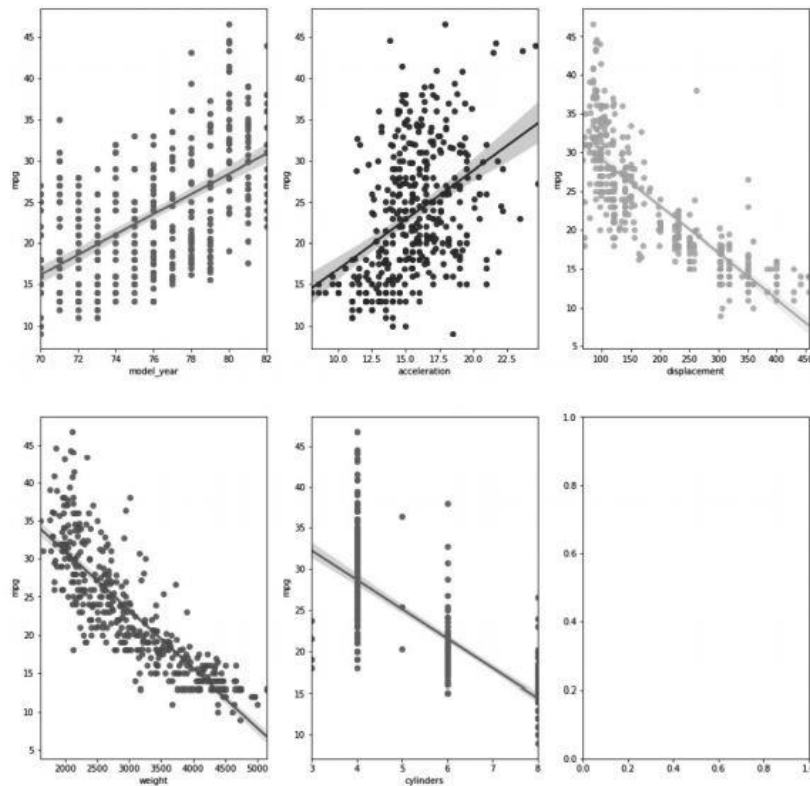


그림 10-12 5개 피처와 연비의 회귀 관계를 보여주는 산점도 + 선형 회귀 그래프

선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

- 2. 선형 회귀 분석 모델 구축하기

- 2. 완성된 자동차 연비 예측 모델을 사용하여 임의의 데이터를 입력하면 연비를 예측할 수 있음

In [17]:	<pre>print("연비를 예측하고 싶은 차의 정보를 입력해주세요.") cylinders_1 = int(input("cylinders : ")) displacement_1 = int(input("displacement : ")) weight_1 = int(input("weight : ")) acceleration_1 = int(input("acceleration : ")) model_year_1 = int(input("model_year : "))</pre>
Out[17]:	<div><div>연비를 예측하고 싶은 차의 정보를 입력해주세요. cylinders : 8 displacement : 350 weight : 3200 acceleration : 22 model_year : 99</div><div>키보드로 값을 입력한 후 [Enter] 누르기</div></div>
In [18]:	<pre>mpg_predict = lr.predict([[cylinders_1, displacement_1, weight_1, acceleration_1 , model_year_1]])</pre>
In [19]:	<pre>print("이 자동차의 예상 연비(MPG)는 %.2f입니다." %mpg_predict)</pre>
Out[19]:	이 자동차의 예상 연비(MPG)는 41.32입니다

In [17]: 5개 항목(독립 변수)을 입력하면 변수에 저장

In [18]: 변수를 회귀 모델에 적용하여 예측 결과값을 구함