

❖ 한글 처리

```
import matplotlib as mpl
import matplotlib.pyplot as plt

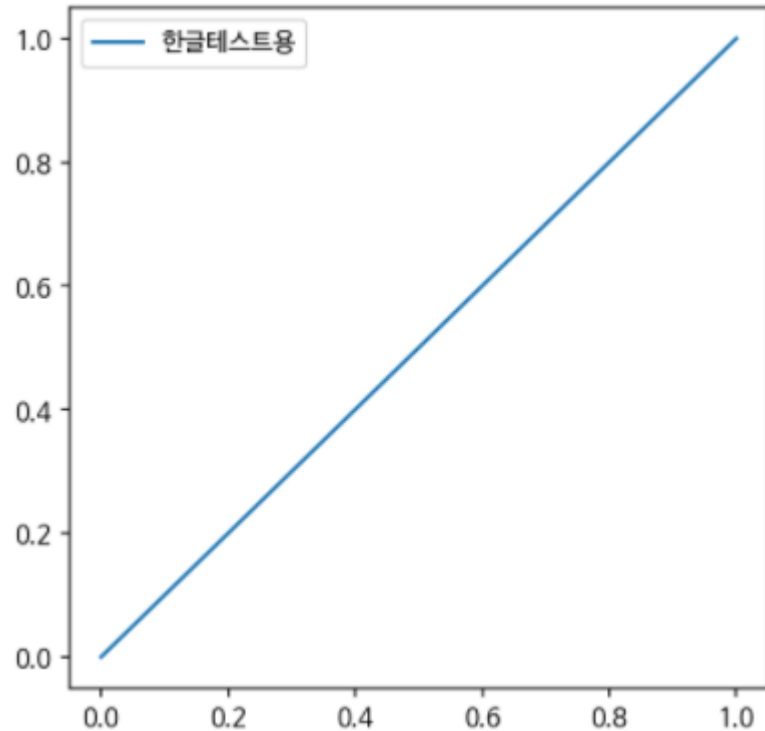
%config InlineBackend.figure_format = 'retina'

!apt -qq -y install fonts-nanum

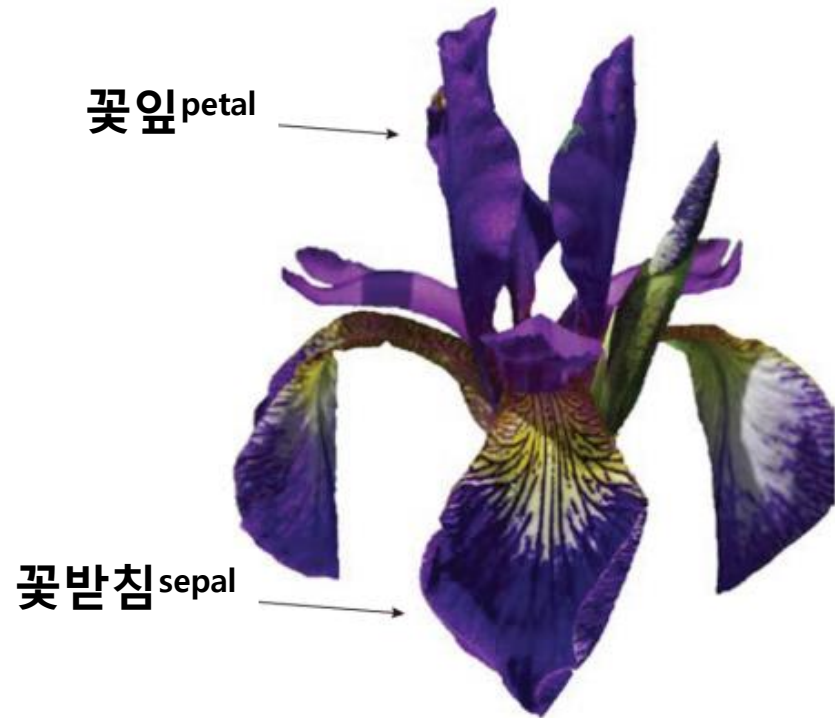
import matplotlib.font_manager as fm
fontpath = '/usr/share/fonts/truetype/nanum/NanumBarunGothic.ttf'
font = fm.FontProperties(fname=fontpath, size=9)
plt.rc('font', family='NanumBarunGothic')
mpl.font_manager._rebuild()
```

❖ 런타임 다시 시작

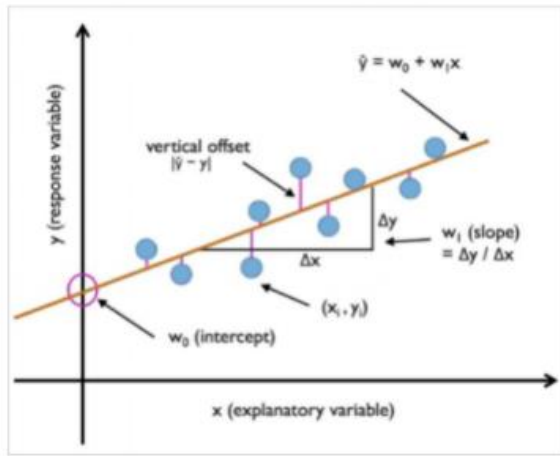
```
plt.figure(figsize=(5,5))  
plt.plot([0,1], [0,1], label='한글테스트용')  
plt.legend()  
plt.show()
```



- ❖ 붓꽃의 꽃잎_{petal}과 꽃받침_{sepal}의 폭과 길이를 센티미터 단위로 측정해 놓은 데이터를 통해, 새로 채집한 붓꽃의 품종을 예측하고자 한다.
- ❖ '붓꽃(Iris)'은 프랑스의 국화



- $Y = aX + b$
- (Y : 종속변수, X : 독립변수, a : 기울기=회귀계수, b : 절편)



선형 회귀 그래프

선형 회귀

•
•
•

라이브러리(library)들을 импорт(import)

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

1. 라이브러리 불러오기

```
[66] from sklearn import datasets
      from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LogisticRegression
      from sklearn.preprocessing import StandardScaler
      import pandas as pd
      import seaborn as sns
      import matplotlib.pyplot as plt
      import numpy as np
```

붓꽃 데이터를 가져오기

```
data = datasets.load_iris()

#데이터셋
input_data = data['data'] # 꽃의 특징 (input data)
target_data = data['target'] #꽃 종류를 수치로 나타낸 것 (0 ~ 2) (target data)

flowers = data['target_names'] # 꽃 종류를 이름으로 나타낸 것
feature_names = data['feature_names'] # 꽃 특징들의 명칭

#sepal : 꽃받침
#petal : 꽃잎
print('꽃을 결정짓는 특징 : {}'.format(feature_names))
print('꽃 종류 : {}'.format(flowers))
```

2.dataset loading

```
[67] data = datasets.load_iris()

#데이터셋
input_data = data['data'] # 꽃의 특징 (input data)
target_data = data['target'] #꽃 종류를 수치로 나타낸 것 (0 ~ 2) (target data)

flowers = data['target_names'] # 꽃 종류를 이름으로 나타낸 것
feature_names = data['feature_names'] # 꽃 특징들의 명칭

#sepal : 꽃받침
#petal : 꽃잎
print('꽃을 결정짓는 특징 : {}'.format(feature_names))
print('꽃 종류 : {}'.format(flowers))

꽃을 결정짓는 특징 : ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
꽃 종류 : ['setosa' 'versicolor' 'virginica']
```

입력 데이터(input data)와 특징(feature)들을 보기

```
iris_df = pd.DataFrame(input_data, columns=feature_names)
iris_df['species'] = target_data
```

```
#맨 위에 있는 데이터 10개 출력
print(iris_df.head(10))
#데이터의 정보 출력
print(iris_df.describe())
```

```
iris_df = pd.DataFrame(input_data, columns=feature_names)
iris_df['species'] = target_data
```

```
#맨 위에 있는 데이터 10개 출력
print(iris_df.head(10))
#데이터의 정보 출력
print(iris_df.describe())
```

	sepal length (cm)	sepal width (cm)	...	petal width (cm)	species
0	5.1	3.5	...	0.2	0
1	4.9	3.0	...	0.2	0
2	4.7	3.2	...	0.2	0
3	4.6	3.1	...	0.2	0
4	5.0	3.6	...	0.2	0
5	5.4	3.9	...	0.4	0
6	4.6	3.4	...	0.3	0
7	5.0	3.4	...	0.2	0
8	4.4	2.9	...	0.2	0
9	4.9	3.1	...	0.1	0

```
[10 rows x 5 columns]
```

	sepal length (cm)	sepal width (cm)	...	petal width (cm)	species
count	150.000000	150.000000	...	150.000000	150.000000
mean	5.843333	3.057333	...	1.199333	1.000000

#4가지 변수(특징)의 관계를 'seaborn' 라이브러리에서 제공하는 pairplot() 메소드로 표현한 그래프 16가지

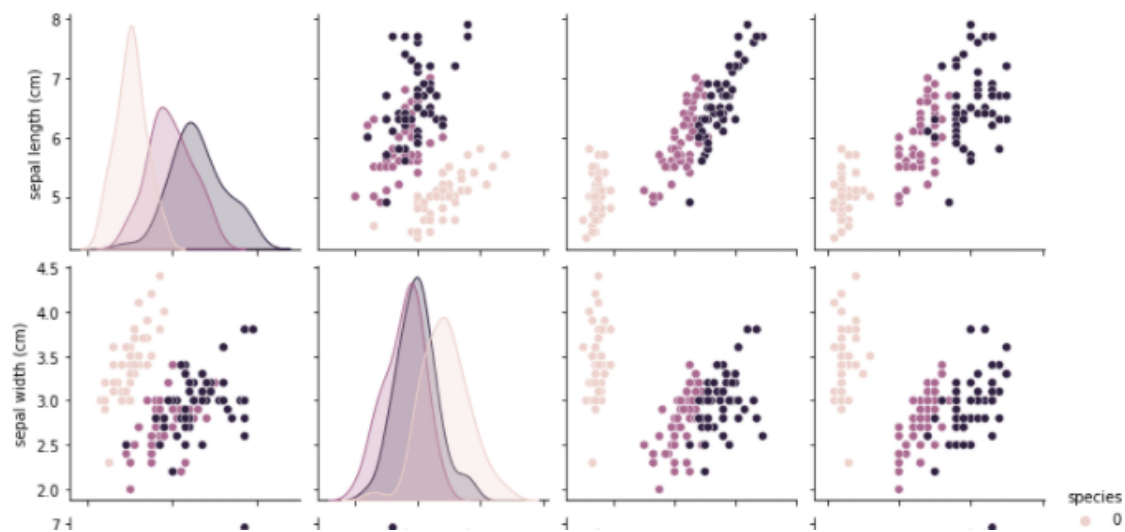
```
sns.pairplot(iris_df, hue='species', vars=feature_names)
```

```
plt.show()
```

#데이터 개괄적 특징 파악 : 4가지 변수(특징)의 관계를 'seaborn' 라이브러리에서 제공하는 pairplot() 메소드로 표현한 그래프 16가지

```
sns.pairplot(iris_df, hue='species', vars=feature_names)
```

```
plt.show()
```



변수(X, y) 선택 : 등간/비율 척도

```
#훈련 데이터와 테스트 데이터 분리
train_input, test_input, train_target, test_target = train_test_split(
    input_data, target_data, random_state=42)
```

```
#표준점수로 데이터 스케일링
scaler = StandardScaler()
train_scaled = scaler.fit_transform(train_input)
test_scaled = scaler.transform(test_input)
```

```
#훈련 데이터와 테스트 데이터 분리
train_input, test_input, train_target, test_target = train_test_split(
    input_data, target_data, random_state=42)

#표준점수로 데이터 스케일링
scaler = StandardScaler()
train_scaled = scaler.fit_transform(train_input)
test_scaled = scaler.transform(test_input)
```

회귀모델 생성

```
lr = LogisticRegression(max_iter=1000)
```

```
#로지스틱 회귀 학습  
lr.fit(train_scaled, train_target)
```

```
#테스트 데이터 예측  
pred = lr.predict(test_scaled[:5])  
print(pred)
```

```
lr = LogisticRegression(max_iter=1000)
```

```
#로지스틱 회귀 학습  
lr.fit(train_scaled, train_target)
```

```
#테스트 데이터 예측  
pred = lr.predict(test_scaled[:5])  
print(pred)
```

```
[1 0 2 1 1]
```

회귀모델 생성 원리 설명

각 특징들의 가중치(weight)와 절편(bias)을 확인

```
#로지스틱 회귀 모델의 가중치와 절편  
#다중 분류 가중치와 절편을 출력하면, 각 클래스마다의 가중치 절편을 출력한다.  
print(lr.coef_, lr.intercept_)
```

```
#로지스틱 회귀 모델의 가중치와 절편  
#다중 분류 가중치와 절편을 출력하면, 각 클래스마다의 가중치 절편을 출력한다.  
print(lr.coef_, lr.intercept_)
```

```
[[-0.97511573  1.08893052 -1.78416098 -1.65224049]  
 [ 0.5072161  -0.30353329 -0.3290721  -0.69052199]  
 [ 0.46789963 -0.78539723  2.11323308  2.34276248]] [-0.39150253  1.92427457 -1.53277204]
```

- 첫번째 배열이 setosa에 대한 가중치와 절편이므로 해당 값들로 식을 구성해보면,
- $\text{setosa}(z1) = (-0.96 * \text{sepal_length}) + (1.09 * \text{sepal_width}) + (-1.78 * \text{petal_length}) + (-1.66 * \text{petal_width}) - 0.39$

회귀모델 생성 원리 설명

- 경우의 수(클래스)가 3가지 이상이다.
- 이런 경우에는 시그모이드(Sigmoid) 함수가 아닌 소프트맥스(Softmax) 함수를 사용

$$e_sum = e^{z1} + e^{z2} + e^{z3}$$

- $z1$ 은 setosa, $z2$ 는 versicolor, $z3$ 은 virginica의 방정식이다.
- 각 클래스들의 z 값을 모두 구한 후, 자연상수 e 의 제곱으로 나타내어 모두 더한다. (= e_sum)
- 확률을 구하고 싶은 클래스의 e^z 을 전체 합으로 나누어주면,
- 해당 클래스의 확률이 되는 것이다. (= e^z / e_sum)
- setosa의 확률을 구하는 식

$$\text{setosa} \text{ 확률} = \frac{e^{z1}}{e_sum}$$

회귀모델 생성 원리 설명

```

setosa_z1 = (-0.96 * 5.1) + (1.09 * 3.5) + (-1.78 * 1.4) + (-1.66 * 0.2) - 0.39
versicolor_z2 = (0.51 * 5.1) + (-0.30 * 3.5) + (-0.32 * 1.4) + (-0.7 * 0.2) - 1.92
virginica_z3 = (0.47 * 5.1) + (-0.79 * 3.5) + (2.11 * 1.4) + (2.34 * 0.2) - 1.53
print(setosa_z1)
print(versicolor_z2)
print(virginica_z3)

```

```

setosa_rs=setosa_z1/(setosa_z1+versicolor_z2+virginica_z3)
versicolor_rs=versicolor_z2/(setosa_z1+versicolor_z2+virginica_z3)
virginica_rs=virginica_z3/(setosa_z1+versicolor_z2+virginica_z3)
print(setosa_rs)
print(versicolor_rs)
print(virginica_rs)

```

```
iris_df.head(3)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0

테스트



```

setosa_z1 = (-0.96 * 5.1) + (1.09 * 3.5) + (-1.78 * 1.4) + (-1.66 * 0.2) - 0.39
versicolor_z2 = (0.51 * 5.1) + (-0.30 * 3.5) + (-0.32 * 1.4) + (-0.7 * 0.2) - 1.92
virginica_z3 = (0.47 * 5.1) + (-0.79 * 3.5) + (2.11 * 1.4) + (2.34 * 0.2) - 1.53
print(setosa_z1)
print(versicolor_z2)
print(virginica_z3)

```

```

-4.294999999999999
-0.957
1.5239999999999994

```

```

setosa_rs=setosa_z1/(setosa_z1+versicolor_z2+virginica_z3)
versicolor_rs=versicolor_z2/(setosa_z1+versicolor_z2+virginica_z3)
virginica_rs=virginica_z3/(setosa_z1+versicolor_z2+virginica_z3)
print(setosa_rs)
print(versicolor_rs)
print(virginica_rs)

```

```

1.1520922746781115
0.25670600858369097
-0.40879828326180245

```

setosa가 확률 값이 가장 큼. 따라서 잘 분류되고 있는 것을 알 수 있음

결정 함수(decision_function)

- decision_function()에 테스트 데이터 5개를 넣고 소수점 2자리까지 출력

#결정 함수(decision_function)로 z1 ~ z3의 값을 구한다.

```
decision = lr.decision_function(test_scaled[:5])
```

```
print(np.round(decision, decimals=2))
```

```
#결정 함수(decision_function)로 z1 ~ z3의 값을 구한다.
```

```
decision = lr.decision_function(test_scaled[:5])
```

```
print(np.round(decision, decimals=2))
```

```
[[-2.21  2.1   0.1 ]  
 [ 5.87  2.56 -8.43]  
 [-9.33  1.8   7.53]  
 [-2.29  1.73  0.56]  
 [-3.59  2.33  1.26]]
```

- z값으로 직접 확률을 구할 필요가 없다. `decision_function()`을 통해 구한 값을 `scipy`에서 제공하는 `softmax` 함수에 전달해주면 각 클래스에 대한 확률을 구해주기 때문이다.

#소프트맥스 함수를 사용한 각 클래스들의 확률
from scipy.special import softmax

```
proba = softmax(decision, axis=1)
print(np.round(proba, decimals=3))
```

#소프트맥스 함수를 사용한 각 클래스들의 확률

```
from scipy.special import softmax
```

```
proba = softmax(decision, axis=1)
print(np.round(proba, decimals=3))
```

```
[[0.012 0.87  0.118]
 [0.965 0.035 0.   ]
 [0.    0.003 0.997]
 [0.013 0.752 0.234]
 [0.002 0.745 0.253]]
```

```
lr = LogisticRegression(max_iter=1000)
```

#로지스틱 회귀 학습

```
lr.fit(train_scaled, train_target)
```

#테스트 데이터 예측

```
pred = lr.predict(test_scaled[:5])
```

```
print(pred)
```

```
[1 0 2 1 1]
```

비교해 보자!!!

- 모델이 예측한 결과값 [1 0 2 1 1] 과 비교해보면,
- 첫번째는 데이터는 87%의 확률로 1(versicolor)이라고 예상했고,
- 두번째 데이터는 97%확률로 0(setosa)이라고 예상했다.

지도학습 회귀문제의 예- 주택가격 예측

- ❖ 다음은 회귀분석의 한 예로 scikit-learn 패키지에서 제공하는 주택가격을 예측하는 문제를 보였다. 이 문제는 범죄율, 공기 오염도 등의 주거 환경 정보 등을 사용하여 70년대 미국 보스턴시의 주택가격을 예측하는 문제이다.

- 데이터셋 요약

```
import pandas as pd
import seaborn as sns
from sklearn.datasets import load_boston
```

```
boston=load_boston()
boston.data.shape
```

```
Warning: warn(msg, category=
(506, 13)
```

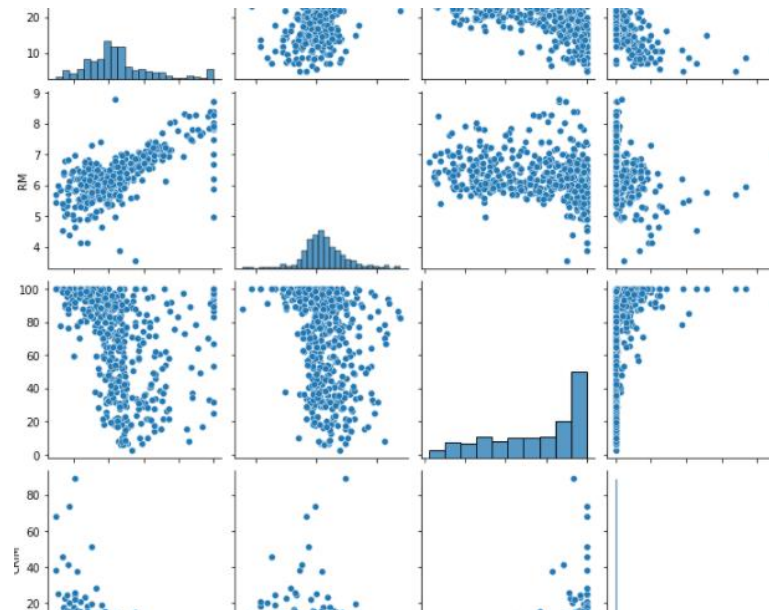
Samples total	506
Dimensionality	13
Features	real, positive
Targets	real 5. - 50.

```
df = pd.DataFrame(boston.data, columns=boston.feature_names)
df["주택가격"] = boston.target
g = sns.pairplot(df[["주택가격", "RM", "AGE", "CRIM"]])
g.fig.suptitle("보스턴 주택가격 데이터 일부 (RM: 방 개수, AGE: 노후화, CRIM: 범죄율)", y=1.02)
plt.show()
```


지도학습 회귀문제의 예- 주택가격 예측

```
import pandas as pd
import seaborn as sns
from sklearn.datasets import load_boston

boston = load_boston()
df = pd.DataFrame(boston.data, columns=boston.feature_names)
df["주택가격"] = boston.target
g = sns.pairplot(df[["주택가격", "RM", "AGE", "CRIM"]])
g.fig.suptitle("보스턴 주택가격 데이터 일부 (RM: 방 개수, AGE: 노후화, CRIM: 범죄율)", y=1.02)
plt.show()
```



지도학습 회귀문제의 예- 주택가격 예측

- ❖ 이 문제를 회귀분석 방법으로 풀면 다음 결과 그래프와 같다. 결과 그래프에서 하나의 점은 하나의 데이터를 뜻한다. 점의 가로축 값은 실제 가격을 나타내고 세로축 값은 회귀분석 결과이다. 만약 회귀분석 방법으로 가격을 정확하게 예측했다면 결과는 기울기가 1인 직선과 같은 형태가 되어야 하지만 실제로는 타원 모양이 되는 경우가 많다.

```
from sklearn.linear_model import LinearRegression

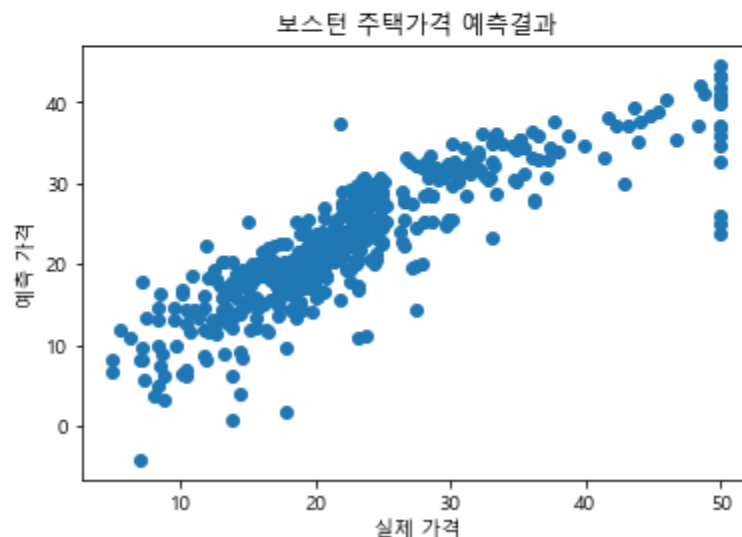
model = LinearRegression().fit(boston.data, boston.target)
predicted = model.predict(boston.data)
plt.scatter(boston.target, predicted)
plt.xlabel("실제 가격")
plt.ylabel("예측 가격")
plt.title("보스턴 주택가격 예측결과")
plt.show()
```

지도학습 회귀문제의 예- 주택가격 예측



```
In [73]: from sklearn.linear_model import LinearRegression

model = LinearRegression().fit(boston.data, boston.target)
predicted = model.predict(boston.data)
plt.scatter(boston.target, predicted)
plt.xlabel("실제 가격")
plt.ylabel("예측 가격")
plt.title("보스턴 주택가격 예측결과")
plt.show()
```



fieldtrip.



회귀분석

❖ 목표:

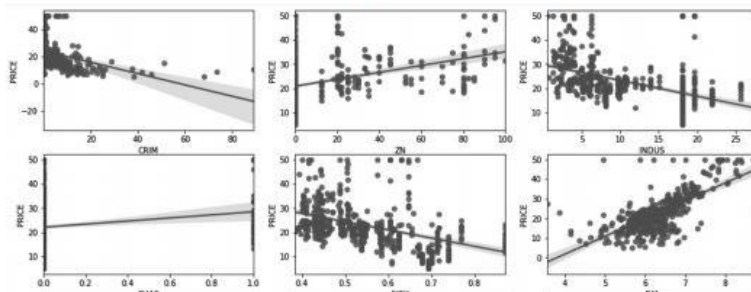
- 보스턴 주택 가격 데이터에 머신러닝 기반의 회귀 분석을 수행. 주택 가격에 영향을 미치는 변수를 확인하고 그 값에 따른 주택 가격을 예측

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



1. 주피터 노트북에서 '10장_주택가격분석'으로 노트북 페이지를 추가하고 입력

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

In [2]: 사이킷런에서 제공하는 데이터셋 `sklearn.datasets` 중에서 보스턴 주택 가격 데이터셋을 사용하기 위해 `load_boston`을 임포트하고, 데이터셋을 로드하여 `load_boston()` 객체 `boston`를 생성

2. 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 `X`가 되는 피쳐들을 `DataFrame` 자료형으로 변환하여 `boston_df`를 생성

`boston_df`의 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_df`의 데이터 5개를 확인 `boston_df.head()`

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
```

• 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 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

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.info()`

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`

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]:	LinearRegression()
:	
In [13]:	#선형 회귀 분석 : 평가 데이터에 대한 예측 수행 -> 예측 결과 Y_predict 구하기 <code>Y_predict = lr.predict(X_test)</code>

In [11]: 선형 회귀 분석 모델 객체 lr을 생성

In [12]: 학습 데이터 XX_train와 YY_train를 가지고 학습을 수행fit().

In [13]: 평가 데이터 XX_test를 가지고 예측을 수행하여predict() 예측값YY_predict를 구함

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 YY_test를 이미 알고 있는 상태에서 평가 데이터 YY_test와 In [13]에서 구한 예측 결과Y_predict의 오차를 계산하여 모델을 평가. 평가 지표 MSE를 구하고 mean_squared_error() 구한 값의 제곱근을 계산하여np.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

1. 선형 회귀를 이용해 분석 모델 구축하기

2) 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 보스턴 주택 가격에 대한 회귀식

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

회귀 분석 결과를 산점도 + 선형 회귀 그래프로 시각화하기

1. 선형 회귀를 이용해 분석 모델 구축하기

2) 선형 회귀 분석 모델을 평가 지표를 통해 평가하고 회귀 계수를 확인하여 피처의 영향을 분석

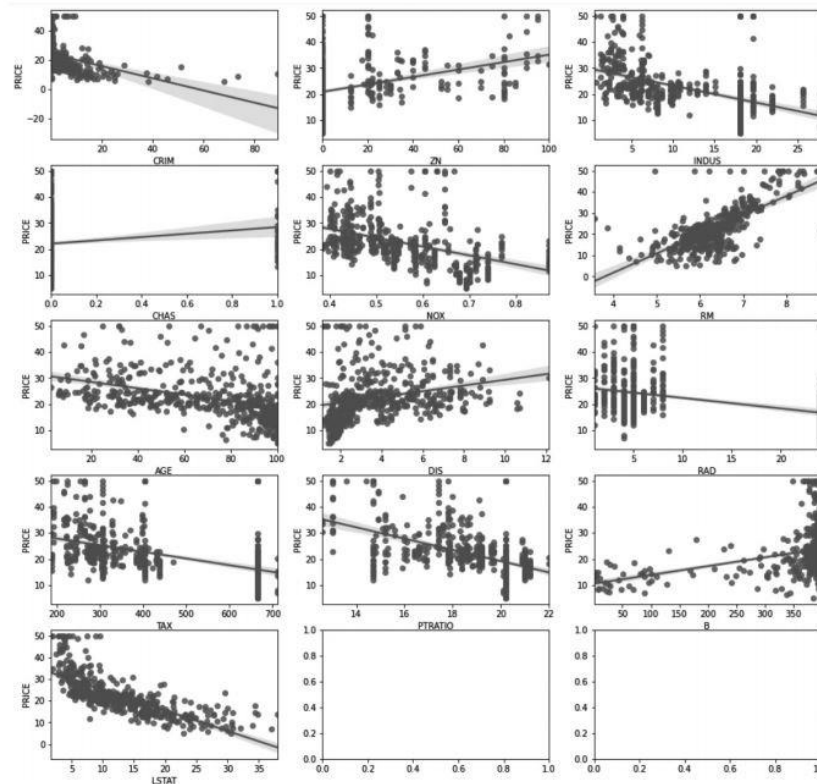
In [17]:	import matplotlib.pyplot as plt import seaborn as sns
In [18]:	fig, axs = plt. subplots (figsize = (16, 16), ncols = 3, nrows = 5) x_features = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT'] for i, feature in enumerate(x_features): row = int(i/3)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

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. **aborn**의 **regplot()**은 산점도 그래프와 선형 회귀 그래프를 함께 그려

줌

1. 선형 회귀를 이용해 분석 모델 구축하기
 - 2) 선형 회귀 분석 모델을 평가 지표를 통해 평가하고 회귀 계수를 확인하여 피처의 영향을 분석
- 13개의 피처와 주택 가격의 회귀 관계를 나타낸 산점도/선형 회귀 그래프

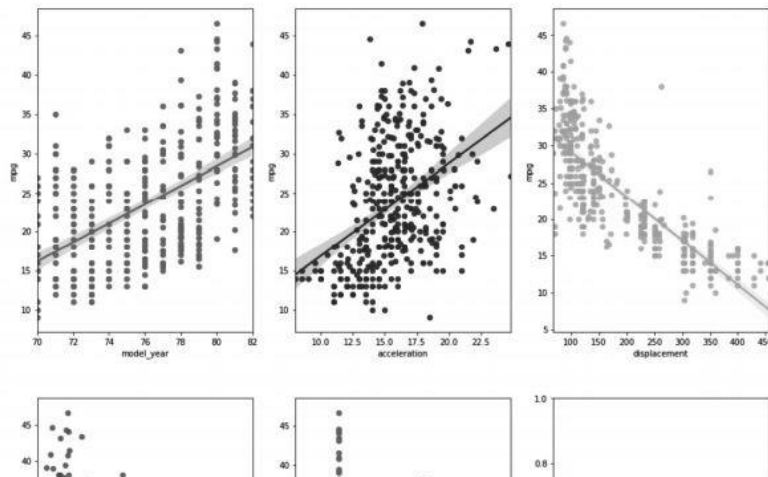


항목에 따른 자동차 연비 예측하기

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



❖ 목표: 자동차 연비 데이터에 머신러닝 기반의 회귀 분석을 수행

- 연비에 영향을 미치는 항목을 확인하고, 그에 따른 자동차 연비를 예측
- <https://archive.ics.uci.edu/ml/index.php> > auto 검색

UCI Machine Learning Repository: x +

← → ↻ archive.ics.uci.edu/ml/index.php

업 컨테이너수업 파이썬 생환 도서 음악 장고 교육 클라우드 맵 AIDeep 경진대회 팔송부르기 컨테이너

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

About Citation Policy Donate a Data Set Contact

auto Search



Repository Web

View ALL Data Sets

Check out the [beta version](#) of the new UCI Machine Learning Repository we are currently testing! [Contact us](#) if you have any issues, questions, or concerns. [Click here to try out the new site.](#)

Welcome to the UC Irvine Machine Learning Repository!


We currently maintain 588 data sets as a service to the machine learning community. You may [view all data sets](#) through our searchable interface. For a general overview of the Repository, please visit our [About page](#). For information about citing data sets in publications, please read our [citation policy](#). If you wish to donate a data set, please consult our [donation policy](#). For any other questions, feel free to [contact the Repository librarians](#).

Supported By:  In Collaboration With: 

Latest News:

09-24-2018: Welcome to the new Repository admins Dheeru Dua and Efi Karra Taniskidou!
04-04-2013: Welcome to the new Repository admins Kevin Bache and Moshe Lichman!
03-01-2010: [Note](#) from donor regarding Netflix data
10-16-2009: Two new data sets have been added.
09-14-2009: Several data sets have been added.
03-24-2008: New data sets have been added!
06-25-2007: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope


Featured Data Set: Flags





Task: Classification
Data Type: Multivariate
Attributes: 30
Instances: 194


From Collins Gem Guide to Flags, 1986


Newest Data Sets:


04-21-2021:  [Synchronous Machine Data Set](#)


04-20-2021:  [Wikipedia Math Essentials](#)


04-20-2021:  [Wikipedia Math Essentials](#)


02-17-2021:  [Hungarian Chickenpox Cases](#)

12-09-2020:  [Myocardial Infarction complications](#)


10-14-2020:  [Gait Classification](#)


10-03-2020:  [Codon usage](#)


09-15-2020:  [In-vehicle coupon recommendation](#)


09-14-2020:  [Dry Bean Dataset](#)


Most Popular Data Sets (hits since 2007):


4375297:  [Iris](#)


2332660:  [Adult](#)


1804856:  [Wine](#)


1723994:  [Wine Quality](#)

1708498:  [Heart Disease](#)

1626171:  [Breast Cancer Wisconsin \(Diagnostic\)](#)

1624423:  [Bank Marketing](#)

1479260:  [Car Evaluation](#)

1187422:  [Abalone](#)

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

The screenshot shows a Google search result for the query 'auto site:archive.ics.uci.edu/ml site:ics.uci.edu'. The search bar at the top displays the query. Below the search bar, the results are listed. The first result, 'Auto MPG Data Set - UCI Machine Learning Repository', is highlighted with a red box. This result includes the URL 'https://archive.ics.uci.edu/datasets/auto+mpg', the title 'Auto MPG Data Set - UCI Machine Learning Repository', and a source description: 'Source: This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset was used in the 1983 American...'. Below this, other results are visible, including 'Automobile Data Set - UCI Machine Learning Repository' and 'Index of /ml/machine-learning-databases/auto-mpg'. At the bottom, there is a section for 'auto site:archive.ics.uci.edu/ml site:ics.uci.edu 관련 이미지' (related images) showing four small images of cars and a car diagram, with a '모두 보기' (view all) button below them.

auto site:archive.ics.uci.edu/ml site:ics.uci.edu

검색결과 약 805개 (0.42초)

<https://archive.ics.uci.edu/datasets/auto+mpg>

Auto MPG Data Set - UCI Machine Learning Repository

Source: This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset was used in the 1983 American...

<https://archive.ics.uci.edu/datasets/automobile>

Automobile Data Set - UCI Machine Learning Repository

Sources: 1) 1985 Model Import Car and Truck Specifications, 1985 Ward's Automotive Yearbook.
2) Personal Auto Manuals, Insurance Services Office, 160 Water ...

<https://archive.ics.uci.edu/machine-learning-databases>

Index of /ml/machine-learning-databases/auto-mpg

Index of /ml/machine-learning-databases/auto-mpg. Parent Directory · Index · auto-mpg.data · auto-mpg.data-original · auto-mpg.names.

auto site:archive.ics.uci.edu/ml site:ics.uci.edu 관련 이미지

피드백

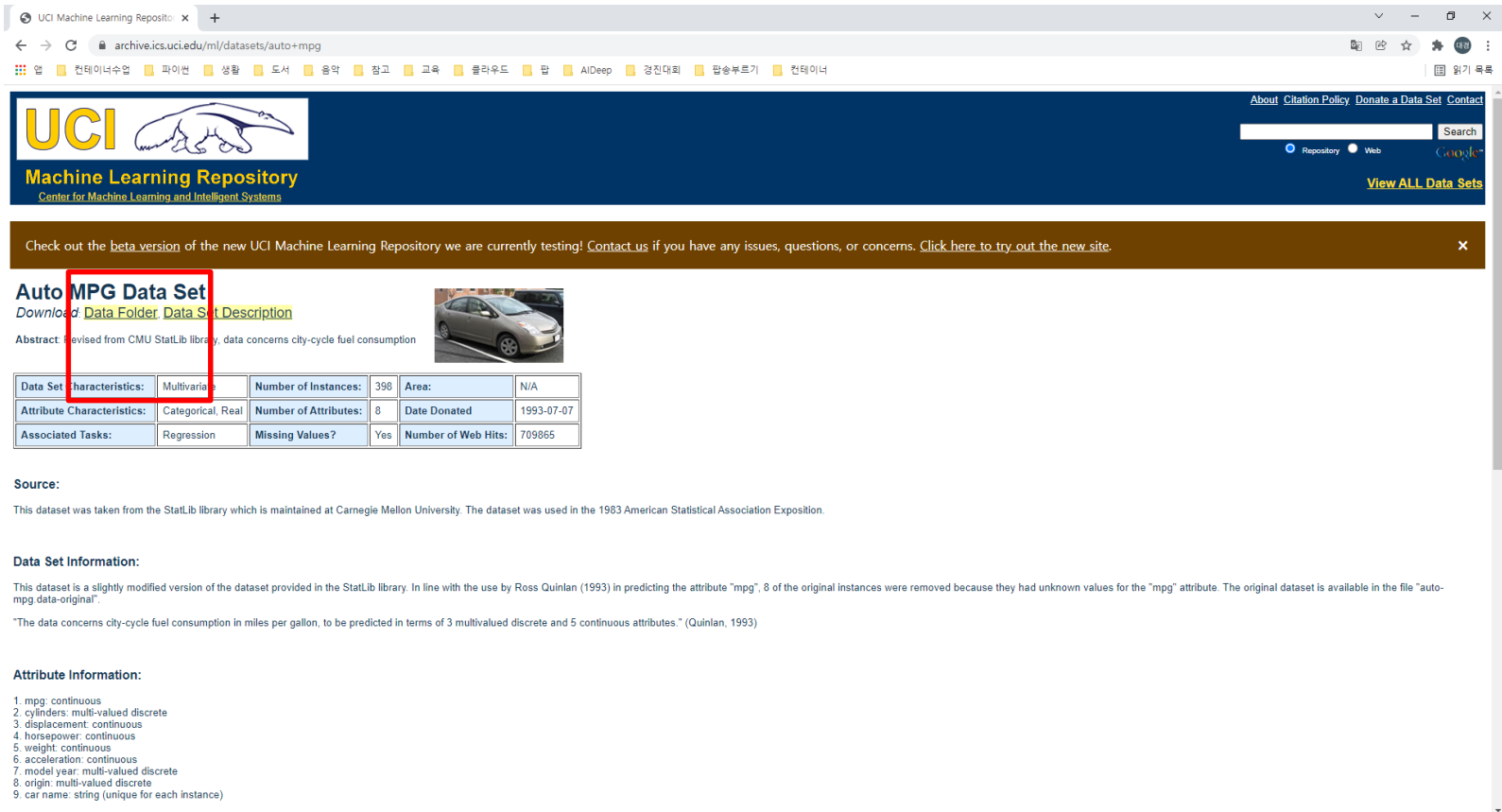
[모두 보기](#)

<https://archive.ics.uci.edu/support/auto+mpg>

Auto MPG Data Set: Support - UCI Machine Learning Repository

Auto MPG Data Set. Below are papers that cite this data set, with context shown. Papers were

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




UCI Machine Learning Repository

Check out the [beta version](#) of the new UCI Machine Learning Repository we are currently testing! [Contact us](#) if you have any issues, questions, or concerns. [Click here to try out the new site.](#)

Auto MPG Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Revised from CMU StatLib library, data concerns city-cycle fuel consumption



Data Set Characteristics:	Multivariate	Number of Instances:	398	Area:	N/A
Attribute Characteristics:	Categorical, Real	Number of Attributes:	8	Date Donated	1993-07-07
Associated Tasks:	Regression	Missing Values?	Yes	Number of Web Hits:	709865

Source:

This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset was used in the 1983 American Statistical Association Exposition.

Data Set Information:

This dataset is a slightly modified version of the dataset provided in the StatLib library. In line with the use by Ross Quinlan (1993) in predicting the attribute "mpg", 8 of the original instances were removed because they had unknown values for the "mpg" attribute. The original dataset is available in the file "auto-mpg.data-original".

"The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multivalued discrete and 5 continuous attributes." (Quinlan, 1993)

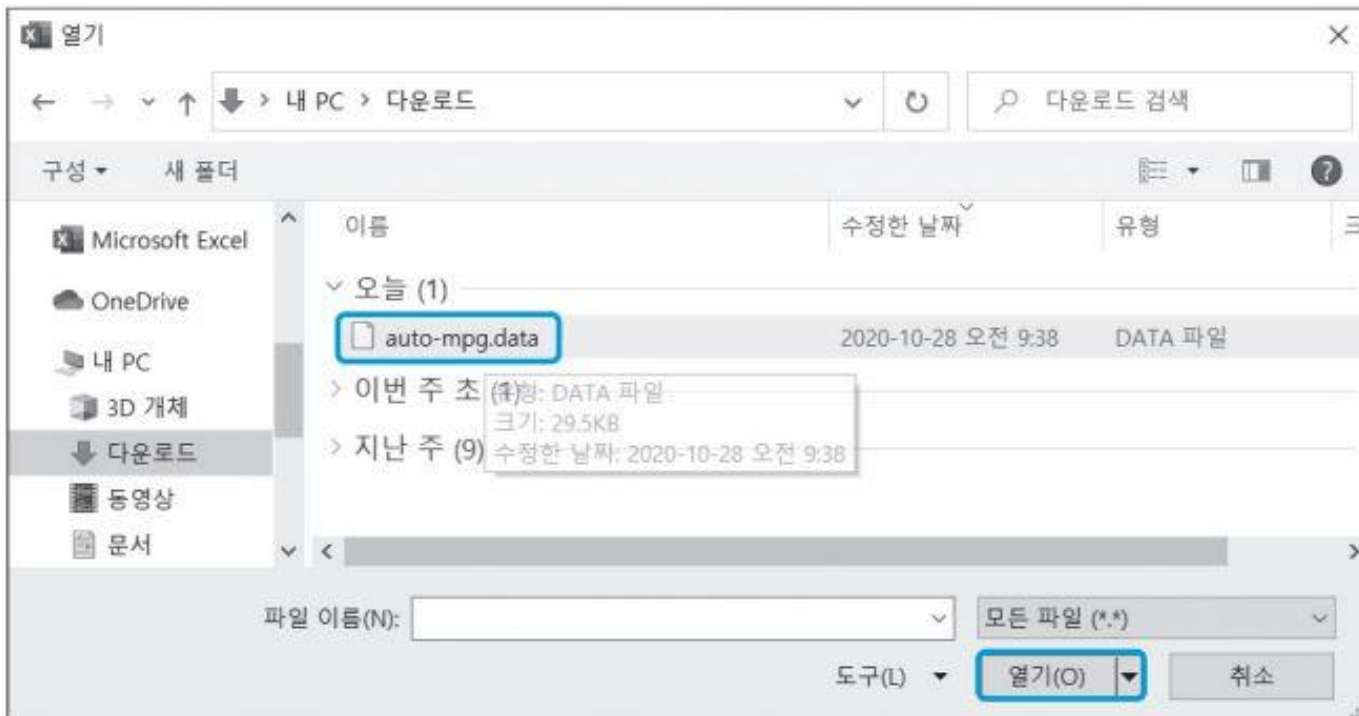
Attribute Information:

1. mpg: continuous
2. cylinders: multi-valued discrete
3. displacement: continuous
4. horsepower: continuous
5. weight: continuous
6. acceleration: continuous
7. model year: multi-valued discrete
8. origin: multi-valued discrete
9. car name: string (unique for each instance)

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



❖ CSV 파일로 변경하기



- ❖ 1단계는 버튼을 클릭, 2단계에서는 [구분 기호]로 '탭'과 '공백'을 선택하고 버튼을 클릭

텍스트 마법사 - 3단계 중 2단계

데이터의 구분 기호를 설정합니다. 미리 보기 상자에서 적용된 텍스트를 볼 수 있습니다.

구분 기호

☒ 탭(T) ☐ 세미콜론(M) ☐ 싹표(C) ☒ 공백(S) ☐ 기타(O):

☒ 연속된 구분 기호를 하나로 처리(R)

텍스트 한정자(Q): "

데이터 미리 보기(P)

18.0	8	307.0	130.0	3504.	12.0	70	1	chevrolet chevelle malibu
15.0	8	350.0	165.0	3693.	11.5	70	1	buick skylark 320
18.0	8	318.0	150.0	3436.	11.0	70	1	plymouth satellite
16.0	8	304.0	150.0	3433.	12.0	70	1	amc rebel sst
17.0	8	302.0	140.0	3449.	10.5	70	1	ford torino

취소 < 뒤로(B) 다음(N) > 마침(D)

- ❖ 텍스트 마법사 3단계에서 데이터 미리 보기를 확인하고 버튼을 클릭

텍스트 마법사 - 3단계 중 3단계

각 열을 선택하여 데이터 서식을 지정합니다.

열 데이터 서식

☒ 일반(O) ☐ 텍스트(T) ☐ 날짜(D): 년월일 ☐ 열 가져오지 않음(전나행)(U)

[일반]을 선택하면 숫자 같은 숫자로, 날짜 같은 날짜로, 모든 나머지 같은 텍스트로 변환됩니다.

고급(A)...

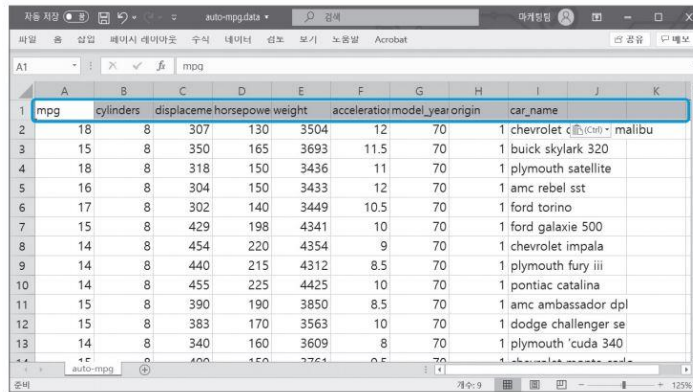
데이터 미리 보기(P)

일반	일반	일반	일반	일반	일반	일반	일반	일반
18.0	8	307.0	130.0	3504.	12.0	70	1	chevrolet chevelle malibu
15.0	8	350.0	165.0	3693.	11.5	70	1	buick skylark 320
18.0	8	318.0	150.0	3436.	11.0	70	1	plymouth satellite
16.0	8	304.0	150.0	3433.	12.0	70	1	amc rebel sst
17.0	8	302.0	140.0	3449.	10.5	70	1	ford torino
15.0	8	429.0	198.0	4341.	10.0	70	1	ford galaxie 500

취소 < 뒤로(B) 다음(N) > 마침(D)

❖ 항목을 구분하기 위해 열 이름을 추가

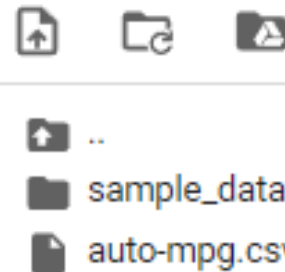
- 행을 삽입하고 열 이름으로 mpg, cylinders, displacement, horsepower, weight, acceleration, model_year, origin, car_name을 각각 입력



	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
1	18	8	307	130	3504	12	70	1	chevrolet chevelle malibu
2	15	8	350	165	3693	11.5	70	1	buick skylark 320
3	18	8	318	150	3436	11	70	1	plymouth satellite
4	16	8	304	150	3433	12	70	1	amc rebel sst
5	17	8	302	140	3449	10.5	70	1	ford torino
6	15	8	429	198	4341	10	70	1	ford galaxie 500
7	14	8	454	220	4354	9	70	1	chevrolet impala
8	14	8	440	215	4312	8.5	70	1	plymouth fury iii
9	14	8	455	225	4425	10	70	1	pontiac catalina
10	15	8	390	190	3850	8.5	70	1	amc ambassador dpl
11	15	8	383	170	3563	10	70	1	dodge challenger se
12	14	8	340	160	3609	8	70	1	plymouth cuda 340
13	14	8	400	150	3761	0.5	70	1	chevrolet monte carlo

❖ 파일을 'auto-mpg.csv'로 저장

파일



❖ 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]:	<p>데이터셋 크기: (398, 9)</p> <table><thead><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></thead><tbody><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></tbody></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><thead><tr><th></th><th>mpg</th><th>cylinders</th><th>displacement</th><th>weight</th><th>acceleration</th><th>model_year</th></tr></thead><tbody><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></tbody></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`, `horsepower`는 분석에 사용하지 않으므로 제거 `data_df.drop()` 후 확인 `data_df.head()`.

❖ 분석에 필요 없는 컬럼을 제거하고 데이터셋의 내용을 확인

In [4]:	<code>print('데이터셋 크기: ', data_df.shape)</code>
Out[4]:	데이터셋 크기: (398, 6)
In [5]:	<code>data_df.info()</code>
Out[5]:	<pre> <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 </pre>

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

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

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 `LinearRegression`과 데이터셋 분리 작업을 위한 `train_test_split`, 성능 측정을 위한 평가 지표인 `mean_squared_error`, `r2_score`를 임포트

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=0.3`로 분할하여 `train_test_split()` 학습 데이터와 평가 데이터로 설정

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] :	<code>LinearRegression()</code>
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 XX_train와 YY_train를 가지고 학습을 수행fit()

In [11]: 평가 데이터 XX_test로 예측을 수행하여predict() 예측값 YY_predict를 구함

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 YY_test를 이미 알고 있음

평가 데이터의 결과값 Y_test과 예측 결과값 Y_predict의 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 계수lr.coef_를 확인

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

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

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

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 [15]:	import matplotlib.pyplot as plt import seaborn as sns
In [16]:	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])

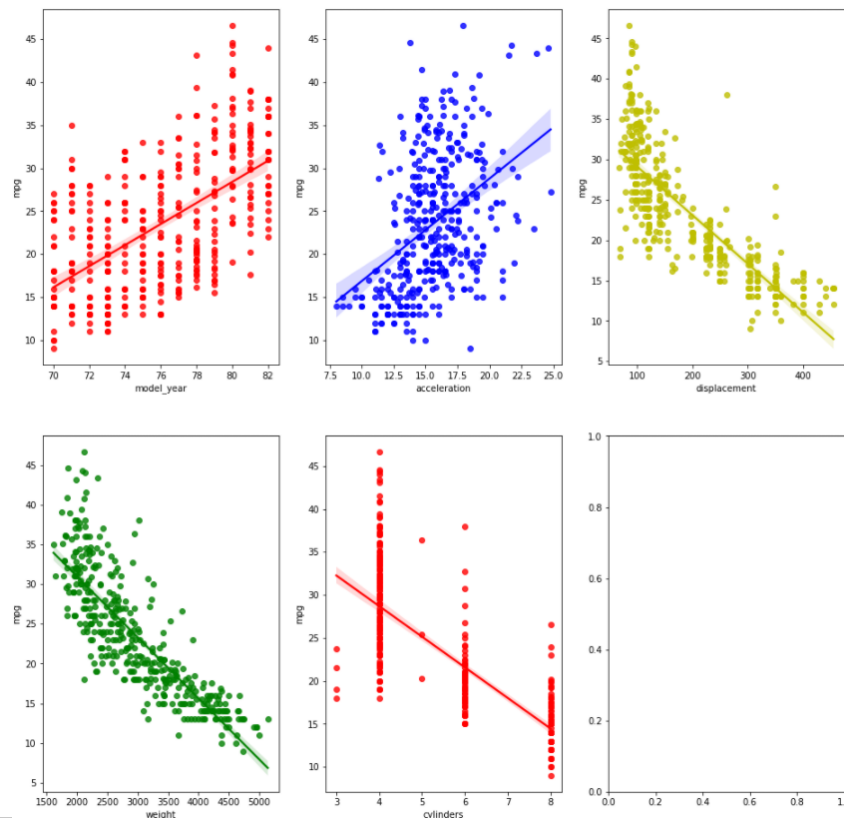
In [15]: 시각화에 필요한 모듈을 импорт

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

1. 선형 회귀 분석 모델 구축하기

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

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



1. 선형 회귀 분석 모델 구축하기

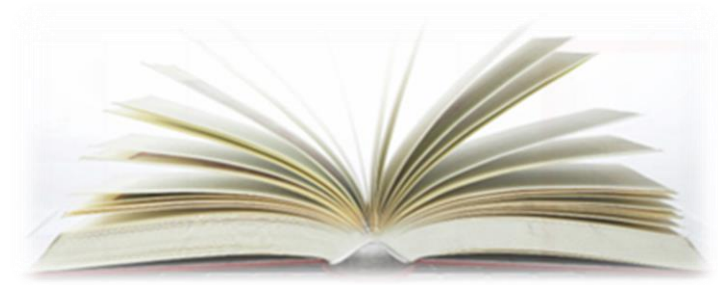
2) 완성된 자동차 연비 예측 모델을 사용하여 임의의 데이터를 입력하면 연비를 예측할 수 있음

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]:	<pre>연비를 예측하고 싶은 차의 정보를 입력해주세요. cylinders : 8 displacement : 350 weight : 3200 acceleration : 22 model_year : 99</pre> <div>키보드로 값을 입력한 후 [Enter] 누르기</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

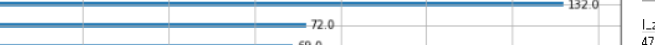
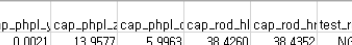
In [17]: 5개 항목(독립 변수)을 입력하면 변수에 저장

In [18]: 변수를 회귀 모델에 적용하여 예측 결과값을 구함

Unit #



Workbooks

A	B	C	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE					
모델	작업일시	작업일	볼트홀 위치 좌						좌우 볼트홀 대 권출 위치 우								진출 위치 좌							ROD 결합면		측정결과
			직각도	Z	Feature importance										Y	Z	경	좌	우							
MODEL_N\DT_MEASURE		dt_workday	cap_bhpr_z	cap_bh																						
303	2022-05-02 AM 8:44	1900-01-00	0.0225	110.0																						
303	2022-05-02 AM 8:55	1900-01-00	0.0400	110.0																						
303	2022-05-02 AM 9:06	1900-01-00	0.0224	110.0																						
303	2022-05-02 AM 9:14	1900-01-00	0.0172	109.5																						
303	2022-05-02 AM 11:16	1900-01-00	0.0140	109.5																						
303	2022-05-02 AM 11:21	1900-01-00	0.0291	109.5																						
303	2022-05-02 PM 1:52	1900-01-00	0.0115	110.0																						

