Python을 이용한 데이터 분석

- Seaborn패키지의 mpg.csv 파일에 대한 기술통계분석
- 자동차 연비mpg에 영향을 미치는 요인에 대한 분석함

이　　름 : 김대현
학　　번 : 202000449
이　　름 : 김대현
학　　과 : 컴퓨터공학부

HANKUK UNIVERSITY OF FOREIGN STUDIES

탐색적 데이터 분석 (explorative data analysis)

- Univariate data analysis (단일 변수에 대한 분석)

각 변수 들을 이해하기 위한 기술 통계량 및 그래프 작성

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

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

Out[2]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86.0	2790	15.6	82	usa	ford mustang gl
394	44.0	4	97.0	52.0	2130	24.6	82	europe	vw pickup
395	32.0	4	135.0	84.0	2295	11.6	82	usa	dodge rampage
396	28.0	4	120.0	79.0	2625	18.6	82	usa	ford ranger
397	31.0	4	119.0	82.0	2720	19.4	82	usa	chevy s-10

398 rows × 9 columns

- seaborn 패키지에서 mpg(자동차 연비).csv 파일을 추출한다

In [3]:
```
mpg.info()
```

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

mpg:연비 displacement:배기량 horsepower:마력 acceleration:가속력

```
In [5]: mpg.describe() #기술통계량
```

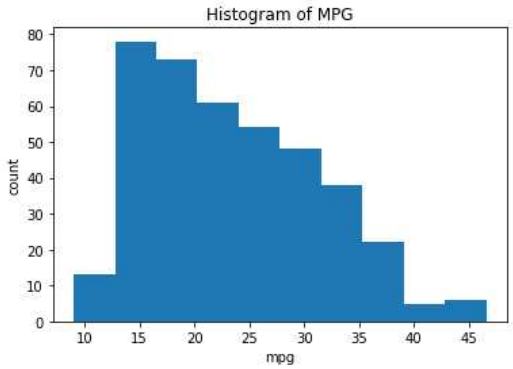
Out [5]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
count	398.000000	398.000000	398.000000	392.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	15.568090	76.010050
std	7.815984	1.701004	104.269838	38.491160	846.841774	2.757689	3.697627
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000	73.000000
50%	23.000000	4.000000	148.500000	93.500000	2803.500000	15.500000	76.000000
75%	29.000000	8.000000	262.000000	126.000000	3608.000000	17.175000	79.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000

- mpg의 기술 통계량 추출 (평균, 표준편차, 최소, 최대값)

```
In [6]: plt.hist(mpg['mpg'], bins=10)
        plt.title("Histogram of MPG")
        plt.xlabel("mpg")
        plt.ylabel("count")
```

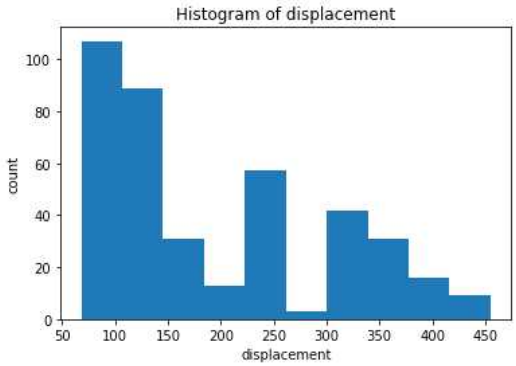
Out [6]: Text(0, 0.5, 'count')



- 차별 mpg(연비) 수치

```
In [7]: plt.hist(mpg['displacement'], bins=10)
        plt.title("Histogram of displacement")
        plt.xlabel("displacement")
        plt.ylabel("count")
```

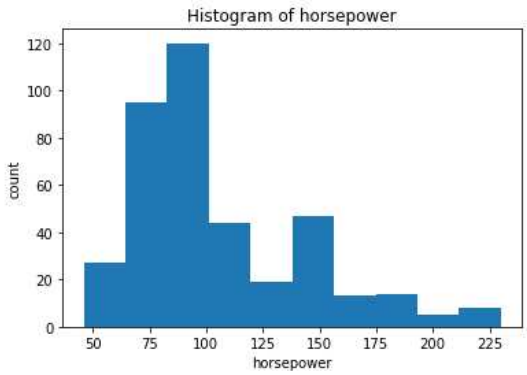
Out [7]: Text(0, 0.5, 'count')



- 차별 displacement(배기량) 수치

```
In [8]: plt.hist(mpg['horsepower'], bins=10)
        plt.title("Histogram of horsepower")
        plt.xlabel("horsepower")
        plt.ylabel("count")
```

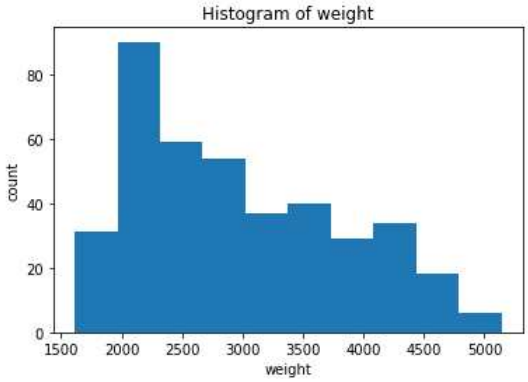
Out [8]: Text(0, 0.5, 'count')



- 차별 horsepower(마력) 수치

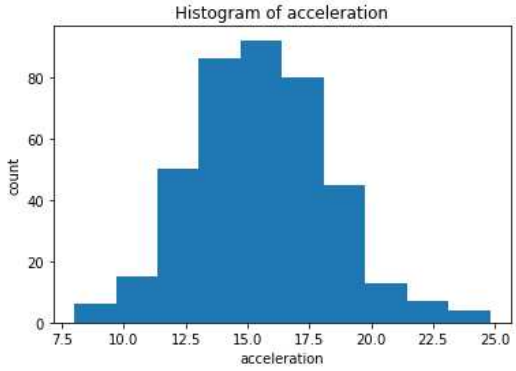
```
In [9]: plt.hist(mpg['weight'], bins=10)
        plt.title("Histogram of weight")
        plt.xlabel("weight")
        plt.ylabel("count")
```

Out [9]: Text(0, 0.5, 'count')



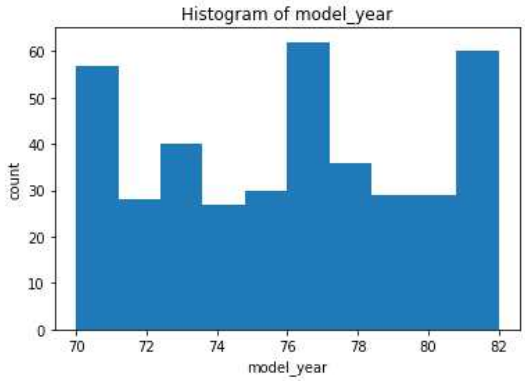
- 차별 weight(무게) 수치

```
In [10]: plt.hist(mpg['acceleration'], bins=10)
         plt.title("Histogram of acceleration")
         plt.xlabel("acceleration")
         plt.ylabel("count")
Out[10]: Text(0, 0.5, 'count')
```



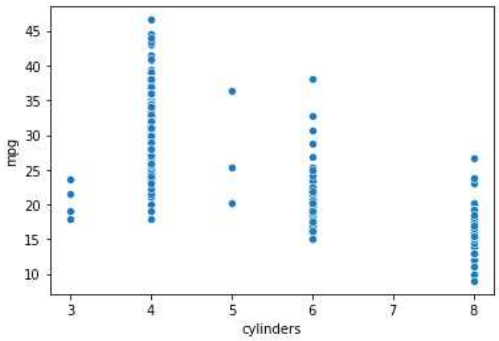
- 차별 acceleration(가속력) 수치

```
In [11]: plt.hist(mpg['model_year'], bins=10)
         plt.title("Histogram of model_year")
         plt.xlabel("model_year")
         plt.ylabel("count")
Out[11]: Text(0, 0.5, 'count')
```



- 차별 model_year(출시 연도?) 수치

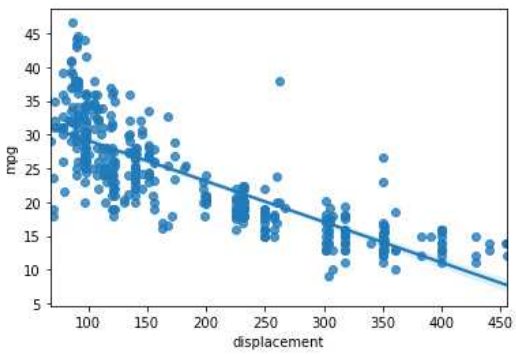
```
In [12]: sns.scatterplot(x='cylinders', y='mpg', data=mpg)
Out[12]: <AxesSubplot:xlabel='cylinders', ylabel='mpg'>
```



- 차별 cylinders(기통) 수치

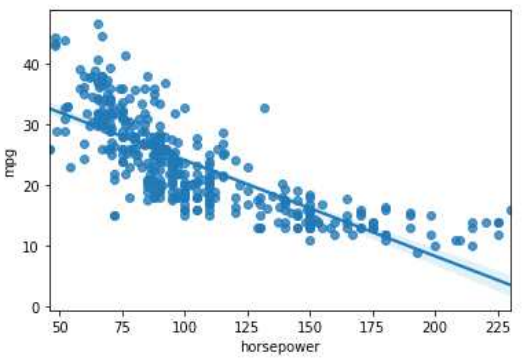
- Bivariate data analysis (1개 독립변수와 종속변수 mpg 관계)
- 단일 독립변수가 mpg에 영향을 미치는 기술통계량 및 그래프 작성, 핵심 요인 도출

```
In [15]: sns.regplot(x='displacement', y='mpg', data=mpg)
Out[15]: <AxesSubplot:xlabel='displacement', ylabel='mpg'>
```



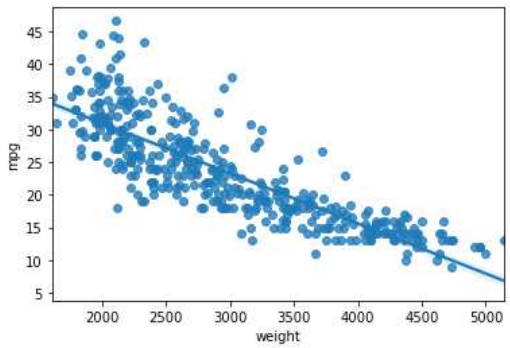
- mpg & displacement (배기량)
- 배기량이 낮을수록 연비가 높다.

```
In [16]: sns.regplot(x='horsepower', y='mpg', data=mpg)
Out[16]: <AxesSubplot:xlabel='horsepower', ylabel='mpg'>
```



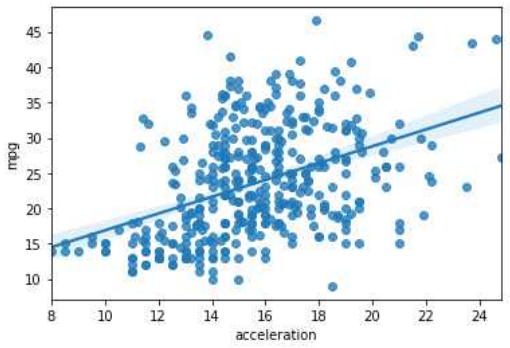
- mpg & horsepower (마력)
- 마력이 낮을수록 연비가 높다.

```
In [15]: sns.regplot(x='weight', y='mpg', data=mpg)
Out[15]: <AxesSubplot:xlabel='weight', ylabel='mpg'>
```



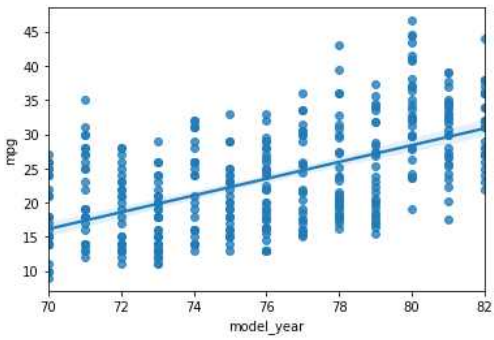
- mpg & weight (무게)
- 무게가 낮을수록 연비가 높다.

```
In [16]: sns.regplot(x='acceleration', y='mpg', data=mpg)
Out[16]: <AxesSubplot:xlabel='acceleration', ylabel='mpg'>
```



- mpg & acceleration (가속력)
- 가속력이 높을수록 연비가 높다고 그래프는 볼수도 있으나, 연비에 따라서 가속력이 각각 다르다고 볼수도 있다.

```
In [17]: sns.regplot(x='model_year', y='mpg', data=mpg)
Out[17]: <AxesSubplot:xlabel='model_year', ylabel='mpg'>
```

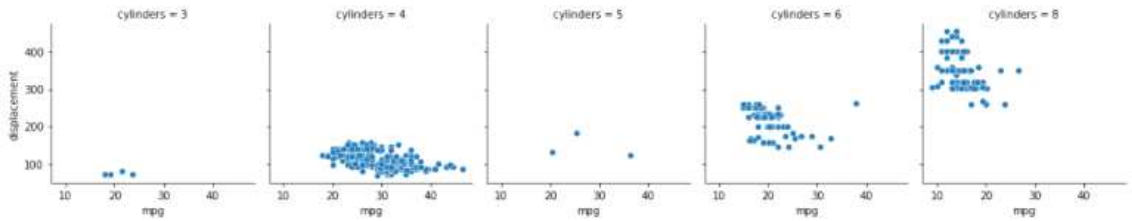


- mpg & model_year (연도), 차량의 연도가 낮을수록 연비가 낮다.

Multivariate data analysis (여러 개의 독립변수와 종속변수 mpg 관계)

- 여러 개의 변수가 mpg에 미치는 영향의 기술통계량 및 그래프 작성
- mpg에 영향을 미치는 변수에 대한 종합적 분석하여 최종 모형을 구축하기 위한 전략수립

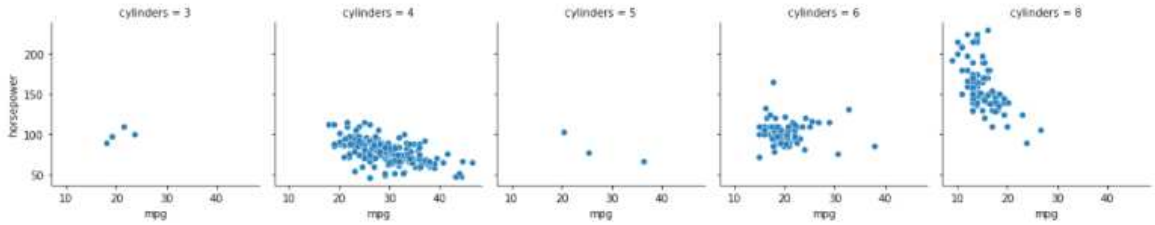
```
In [31]: facet = sns.FacetGrid(mpg, col='cylinders')
         facet.map(sns.scatterplot, 'mpg', 'displacement')
Out[31]: <seaborn.axisgrid.FacetGrid at 0x124c126e6a0>
```



- 그래프를 cylinder(기통)의 개수별로 차종을 분류, x축은 mpg연비 별로 분류할수 있게 고정값을 잡아 놓았다 (공통사항), y축은 배기량으로 잡아 놓았다. 그래프에선 차의 cylinder가 많을수록 배기량이 높다 & 배기량이 높을수록 연비값이 낮아진다 라고 표현.

```
In [24]: facet = sns.FacetGrid(mpg, col='cylinders')
         facet.map(sns.scatterplot, 'mpg', 'horsepower')
```

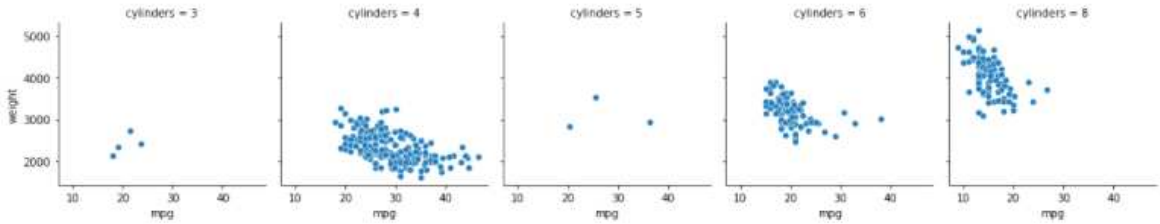
```
Out[24]: <seaborn.axisgrid.FacetGrid at 0x124bc4bc400>
```



- **이 그래프에선 y축은 마력으로 잡아 놓았다. 그래프에선 차의 cylinder가 많을수록 & 차의 마력이 높을수록 연비값이 낮아진다.**

```
In [25]: facet = sns.FacetGrid(mpg, col='cylinders')
         facet.map(sns.scatterplot, 'mpg', 'weight')
```

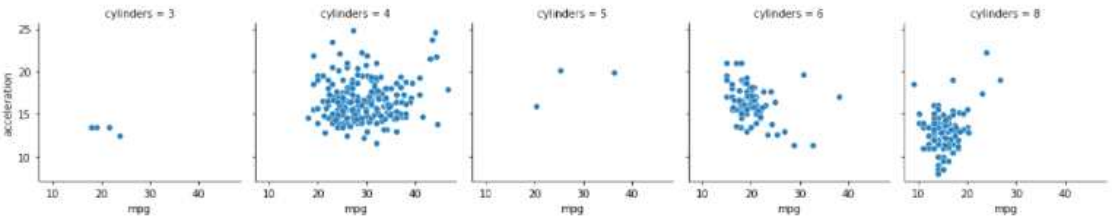
```
Out[25]: <seaborn.axisgrid.FacetGrid at 0x124bd5e4b20>
```



- **이 그래프에선 y축은 차의 무게 으로 잡아 놓았다. 그래프에선 차의 cylinder가 많을수록 & 차의 무게가 무거울수록 연비값이 낮아진다.**

```
In [26]: facet = sns.FacetGrid(mpg, col='cylinders')
         facet.map(sns.scatterplot, 'mpg', 'acceleration')
```

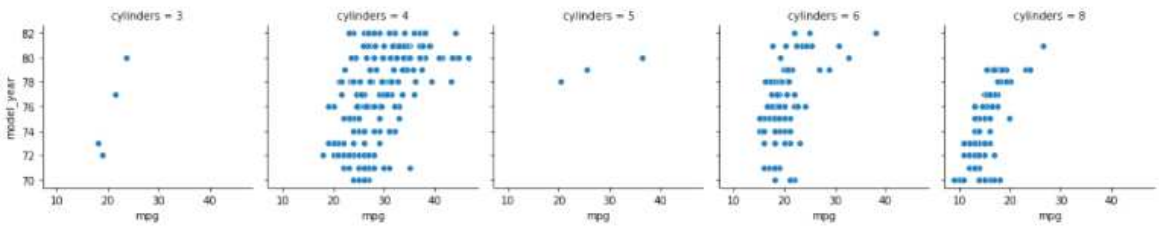
```
Out[26]: <seaborn.axisgrid.FacetGrid at 0x124bd74cf70>
```



- **이 그래프에선 y축은 차의 가속력으로 잡아 놓았다. 그래프에선 차의 cylinder가 많을수록 & 차의 가속력이 느릴수록 무거울수록 연비값이 낮아진다.**

```
In [28]: facet = sns.FacetGrid(mpg, col='cylinders')
         facet.map(sns.scatterplot, 'mpg', 'model_year')
```

```
Out[28]: <seaborn.axisgrid.FacetGrid at 0x124bded2e50>
```



- **이 그래프에선 y축은 차의 연도로 잡아 놓았다. 그래프에선 차의 cylinder가 많을수록 & 차의 연도가 적을수록 연비값이 낮아진다.**
-

- 회귀분석(regression analysis)를 통한 예측모형구축

Scikit-learn을 이용하여 mpg를 예측하는 모형 구축,
모형의 타당성을 검토하기 위하여 train-test 데이터 셋을 구분하여 모형을 구축함

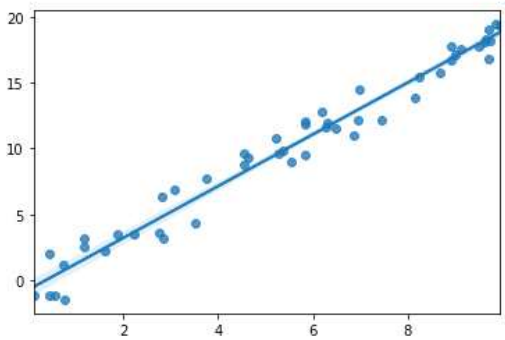
Scikit-Learn 분석 사례: supervised learning (회귀분석 모형 구축)

```
In [72]: mpg = np.random.RandomState(42)        # 랜덤데이터 생성객체 initiation
         x = 10 * rng.rand(50)
         y = 2 * x - 1 + rng.randn(50)

In [73]: x
Out[73]: array([0.56303276, 1.18817916, 1.17526247, 6.49210302, 7.46044879,
                5.83368765, 9.62172548, 3.7487058 , 2.85712086, 8.68599128,
                2.23595839, 9.63222539, 0.12154475, 9.69878827, 0.43159912,
                8.91143114, 5.27701109, 9.92964796, 0.73796565, 5.53854284,
                9.69302536, 5.23097844, 6.29398638, 6.95748689, 4.54541065,
                6.2755808 , 5.84314312, 9.0115801 , 0.4544638 , 2.8096319 ,
                9.50411484, 8.90263784, 4.55656753, 6.20132598, 2.77381183,
                1.8812116 , 4.63698405, 3.53352228, 5.83656112, 0.77734637,
                9.74394808, 9.86210744, 6.98161714, 5.36096366, 3.09527616,
                8.1379502 , 6.84731173, 1.62616939, 9.10927184, 8.22537243])

In [74]: y
Out[74]: array([-1.19416769,  3.20781709,  2.52996506, 11.51503039, 12.20776306,
                12.02124768, 18.12891112,  7.7352279 ,  3.11981407, 15.77260754,
                 3.47716047, 18.31143138, -1.20697598, 19.02042647, -1.20442219,
                16.68048279,  9.67431781, 19.37373476,  1.18754617,  8.9524436 ,
                16.85193654, 10.73963371, 11.92028677, 12.16648724,  9.64197327,
                11.66683624, 11.86558342, 17.09067869,  1.96967553,  6.37460463,
                17.75926553, 17.77684663,  8.75851101, 12.77128351,  3.5827002 ,
                 3.44847465,  9.33239259,  4.30830507,  9.48986372, -1.48453944,
                18.21848932, 19.44175715, 14.46559133,  9.79602211,  6.81916787,
                13.89579894, 10.99124101,  2.19679109, 17.60260914, 15.41805011])
```

```
In [75]: sns.regplot(x, y)
C:₩Users₩bigda₩anaconda3₩lib₩site-packages₩seaborn₩_d
bles as keyword args: x, y. From version 0.12, the on
other arguments without an explicit keyword will resu
  warnings.warn(
Out[75]: <AxesSubplot:>
```




```
In [76]: type(x)
Out[76]: numpy.ndarray

In [77]: x.shape
Out[77]: (50,)

In [78]: x.ndim
Out[78]: 1
```

- **Scikit-learn을 이용하여 mpg를 예측하는 모형 구축**

In [79]:
```
# 1. Choose a class of model
from sklearn.linear_model import LinearRegression
```

In [80]:
```
# 2. Choose model hyperparameters  (Instantiation)
model = LinearRegression(fit_intercept=True)
model
```

Out[80]:
```
LinearRegression()
```

In [82]:
```
#  3. Arrange data into a feature matrix and target vector
X = x[:, np.newaxis]
X.shape
```

Out[82]:
```
(50, 1)
```

In [86]:
```
#  4. Fit the model to your data
model.fit(X,y)
```

Out[86]:
```
LinearRegression()
```

In [87]:
```
model.coef_
```

Out[87]:
```
array([1.97097559])
```

In [88]:
```
model.intercept_
```

Out[88]:
```
-0.750732282984286
```

In [89]:
```
# 5. 구간 내 데이터에 대한 예측
x_2 = np.arange(11)
X_2 = x_2[:, np.newaxis]
y_2 = model.predict(X_2)
```

In [90]:
```
x_2
```

Out[90]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [91]:
```
y_2
```

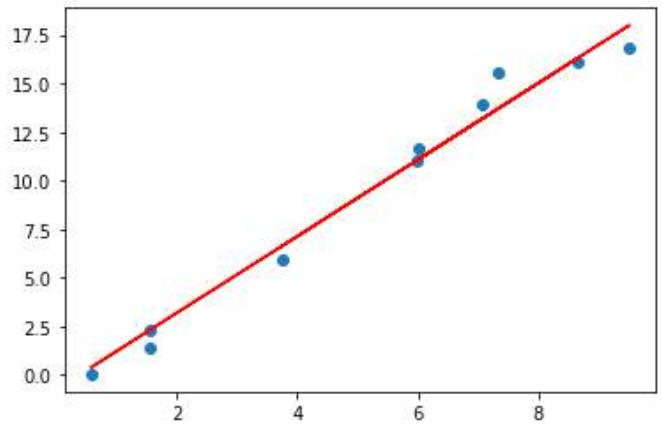
Out[91]:
```
array([-0.75073228,  1.22024331,  3.1912189 ,  5.16219449,  7.13317008,
        9.10414567, 11.07512126, 13.04609685, 15.01707245, 16.98804804,
       18.95902363])
```

모형의 타당성 (성능) 평가하기

In [93]:
```
# 동일한 시스템에서 새로운 데이터를 준비하고 예측에 대한 오차 산출하기
x_test = 10 * mpg.rand(10)
y_test = 2 * x_test - 1 + rng.randn(10)
X_test = x_test[:, np.newaxis]
```

In [94]:
```
# 새로운 데이터 셋에 대한 예측값에 대한 오차 그래프로 작성하기
y_pred = model.predict(X_test)
plt.scatter(x_test, y_test)
plt.plot(x_test, y_pred, color="red")
```

Out [94]:
```
[<matplotlib.lines.Line2D at 0x124c3a9f970>]
```

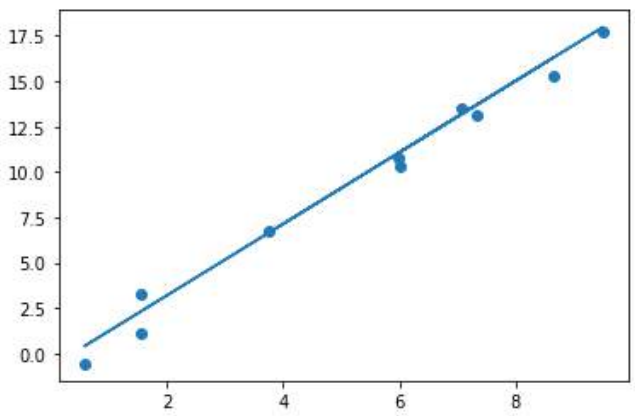


In [95]:
```
# 새로운 데이터에 대한 오차항의 제곱의 평균값(MSE) 산출하기
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)
```

Out [95]:
```
0.73560111676651
```

In [96]:
```
#  시스템에 변화가 있는 상황에서 새로운 데이터에 대한 예측 및 오차 산출하기
y_test = 2 * x_test - 0.02 * x_test**2 + mpg.randn(10)
ypred = model.predict(X_test)
plt.scatter(x_test, y_test)
plt.plot(x_test, y_pred)
```

Out [96]:
```
[<matplotlib.lines.Line2D at 0x124c3afb670>]
```



In [97]:
```
mean_squared_error(y_test, y_pred)
```

Out [97]:
```
0.5583327088174362
```