Boston 집값 선형회귀 예제

라이브러리 및 패키지 Import

In [14]:

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

import matplotlib.pylab as plt
import matplotlib
%matplotlib inline
matplotlib.style.use('ggplot')

from sklearn.linear_model import LinearRegression
from sklearn.datasets import fetch_openml
```

데이터셋 불러오기

In [15]:

```
boston_dataset = fetch_openml(name='boston')
```

In [16]:

```
# 로드한 boston 전체 데이터에 key 값을 출력
print(boston_dataset.keys())
# boston 전체 데이터 중 data에 대한 전체 행, 열 길이를 출력
print(boston_dataset.data.shape)
# boston 데이터에 컬럼 이름을 출력
print(boston_dataset.feature_names)
```

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

In [17]:

```
print(boston_dataset.DESCR)
```

```
**Author**:
**Source**: Unknown - Date unknown
**Please cite**:

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.
Variables 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


Information about the dataset
CLASSTYPE: numeric
CLASSINDEX: last

Downloaded from openml.org.
```

데이터 전처리

In [18]:

```
# DataFrame 형태로 변경
data = pd.DataFrame(boston_dataset.data)
data.tail()
```

Out[18]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88

In [19]:

```
# 칼럼명 변경
data.columns = boston_dataset.feature_names
data.tail()
```

Out[19]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88

In [20]:

```
# 타겟 변수 지정
data['Price'] =  boston_dataset.target
data.tail()
```

Out[20]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

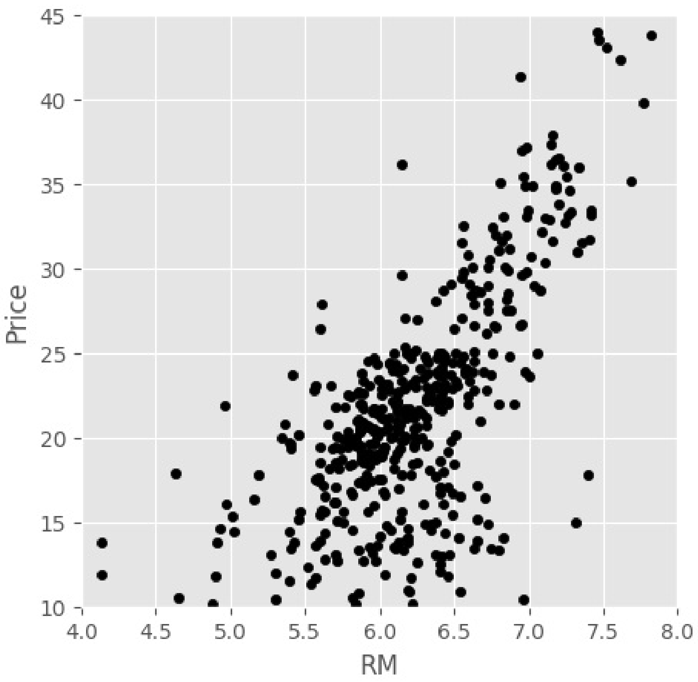
Scatter Plot

In [21]:

```
# 데이터 분포 확인
data.plot(kind='scatter', x ="RM", y="Price", figsize=(5, 5), color='black', xlim=(4,8), ylim=(10,45))
```

Out[21]:

```
<Axes: xlabel='RM', ylabel='Price'>
```



데이터 학습

In [22]:

```
linear_regression = LinearRegression()
linear_regression.fit(X=pd.DataFrame(data['RM']), y=data['Price'])
prediction = linear_regression.predict(X=pd.DataFrame(data['RM']))
print('Y = ax+b 일 때,')
print('a 값: ', linear_regression.coef_)
print('b 값: ',linear_regression.intercept_)
```

```
Y = ax+b 일 때,
a 값:  [9.10210898]
b 값:  -34.67062077643857
```

적합도 검정

In [23]:

```
residuals = data['Price'] - prediction
residuals.describe()
```

Out[23]:

```
count    5.060000e+02
mean     2.134437e-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
```

In [24]:

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

```
R_squared:  0.48352545599133423
```

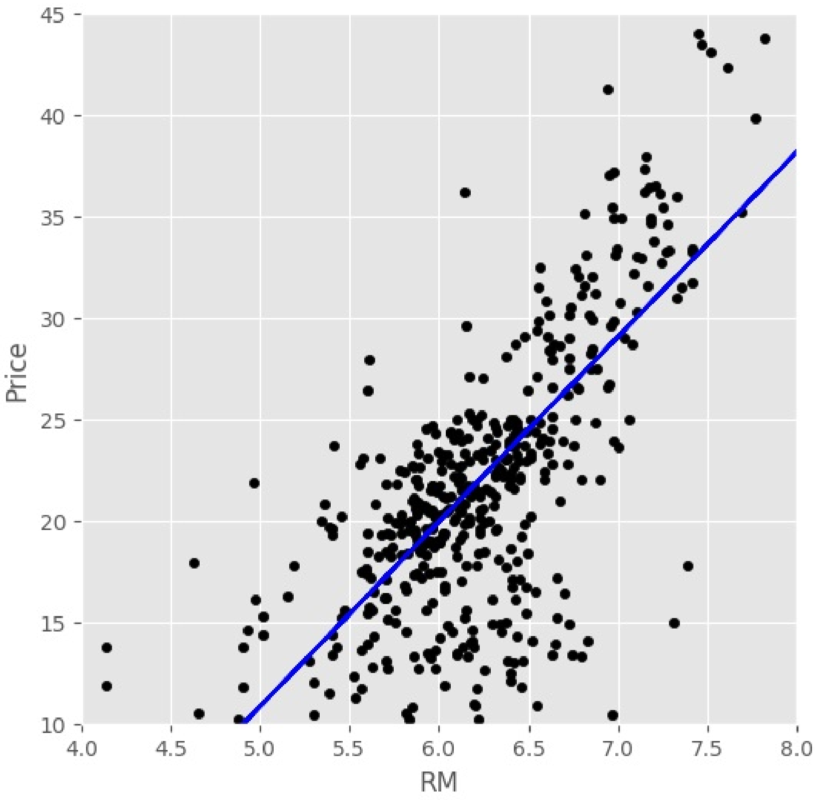
In [25]:

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

plt.plot(data['RM'], prediction, color='b')
```

Out[25]:

```
[<matplotlib.lines.Line2D at 0x23ea5017f80>]
```



성능 평가

In [26]:

```
from sklearn.metrics import mean_squared_error

print('score: ', linear_regression.score(X=pd.DataFrame(data['RM']), y= data['Price']))
print('Mean Squared Error: ', mean_squared_error(prediction, data['Price']))
print('RMSE: ', mean_squared_error(prediction, data['Price'])**.5)
```

```
score:  0.48352545599133423
Mean Squared Error:  43.60055177116956
RMSE:  6.603071389222561
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js