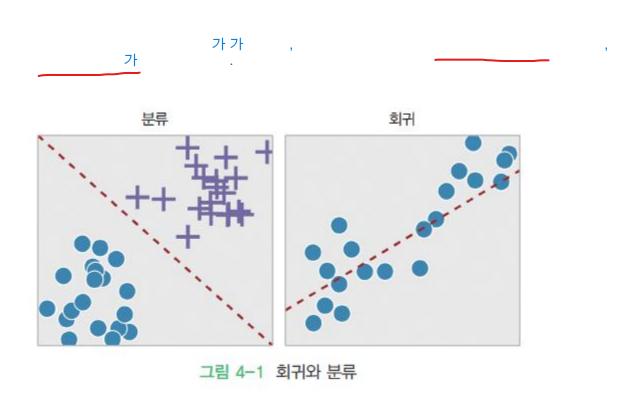
선형 회귀



- 회귀의 개념을 이해한다.
- 경사 하강법을 이해한다.
- 과잉 적합과 과소 적합을 이해한다.
- 파이썬과 sklearn을 이용하여 회귀를 구현해본다.

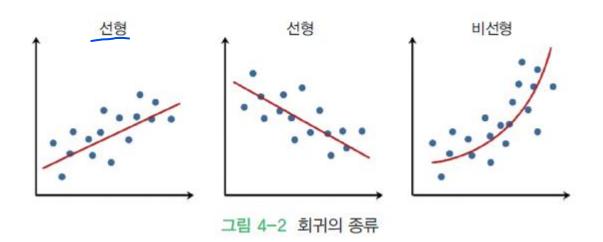


#### 회귀(regression)와 분류(classification)



# 선형 회귀

- 회귀란 일반적으로 데이터들을 2차원 공간에 찍은 후에 이들 데이터들을 가장 잘 설명하는 <mark>직선</mark>이나 곡선을 <u>찾는 문제라</u>고 할 수 있다.
- y = f(x)에서 출력 y가 실수이고 입력 x도 실수일 때 함수 f(x)를 예측하는 것이 회귀이다.



# 선형 회귀의 예

- 부모의 키와 자녀의 키의 관계 조사
- 면적에 따른 주택의 가격
- 연령에 따른 실업율 예측
- 공부 시간과 학점 과의 관계
- CPU 속도와 프로그램 실행 시간 예측

## 선형 회귀 소개

- <mark>직선</mark>의 방정식: f(x) = mx+b
- 선형 회귀는 입력 데이터를 가장 잘 설명하는 기울기와 절편값을 찾는 문 제이다
- 선형 회귀의 기본식: f(x) = Wx+b
  - 기울기 W->가중치
  - 절편 b 〉바이어스

 $\sqrt{\chi}$ 

# 선형 회귀 예제



그림 4-3 선형 회귀의 예제

## 선형 회귀의 종류

단순 선형 회귀: 단순 선형 회귀는 독립 변수(x)가 하나인 선형 회귀이다.

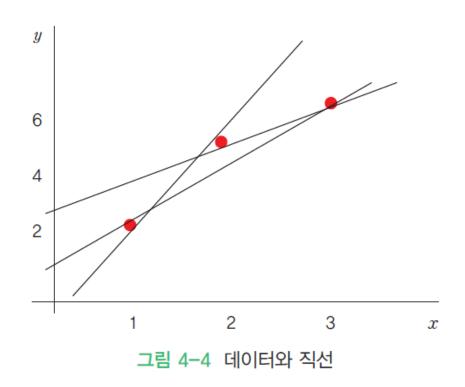
$$f(x) = wx + b \qquad \text{input}, \qquad 7$$

• 다중 선형 회귀: 독립 변수가 여러 개인 경우

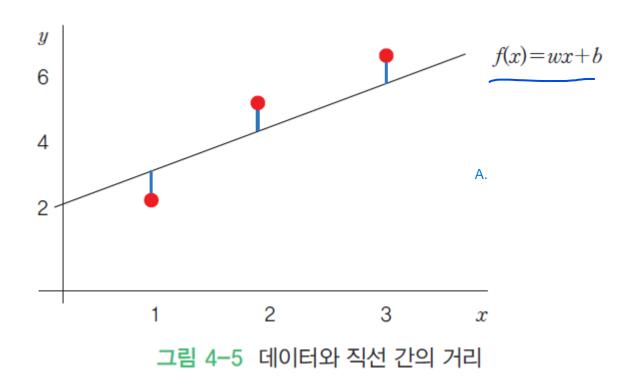
$$f(x,y,z)=w_0+w_1x+w_2y+w_3z$$
 매출  $=w_0+w_1\times$ 인터넷 광고 $+w_2\times$ 신문광고 $+w_3\times TV$ 광고

# 선형 회귀의 원리

X	У
1	2
2	5
3	6



# 직선과 데이터의 거리



가

가

가



직선과 데이터 사이의 간격을 제곱하여 합한 값을 손실 함수(loss function) 또는 비용 함수(cost function)라고 한다.

$$Loss = \frac{1}{3}((f(x_1) - y_1)^2 + (f(x_2) - y_2)^2 + (f(x_3) - y_3)^2)$$

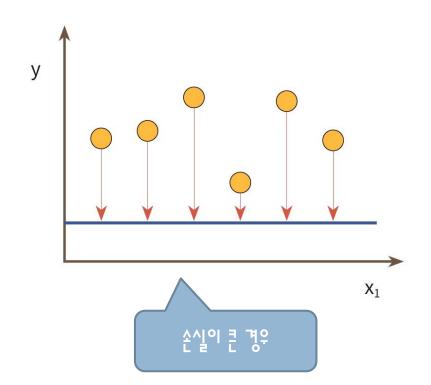


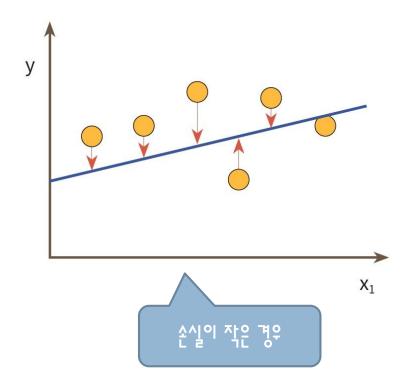
$$Loss\left( {\left. {W,b} \right) = \frac{1}{n}\sum\limits_{i = 1}^n {(f({x_i}) - {y_i})^2 } = \frac{1}{n}\sum\limits_{i = 1}^n {(({\left. {W{x_i} + b} \right) - {y_i})^2 }$$



argmin Loss(W, b)W, b

# 손실 함수





#### 선형 회귀에서 손실 함수 최소화 방법

분석적인 방법: 독립 변수와 종속 변수가 각각 하나인 선형 회귀

$$w = \frac{\sum_{i=1}^{n} (x_i - \overline{x}) (y_i - \overline{y})}{\sum_{i=1}^{n} (x_i - \overline{x})^2}, \quad b = \overline{y} - w\overline{x}$$

- 경사 하강법(gradient descent method):
  - 경사 하강법은 손실 함수가 어떤 형태이라도, 또 매개 변수가 아무리 많아도 적용할 수 있는 일반적인 방법이다.
  - 점진적인 학습이 가능하다



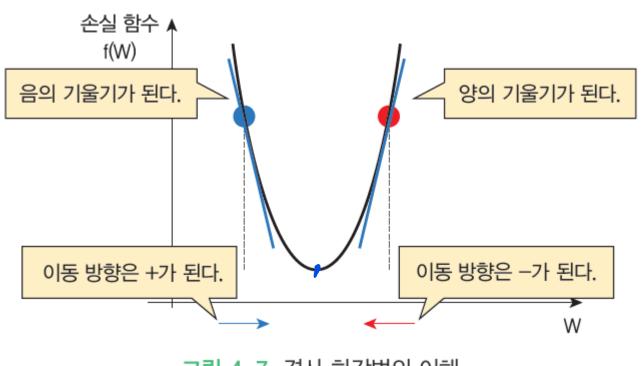


그림 4-7 경사 하강법의 이해

w 가 w .

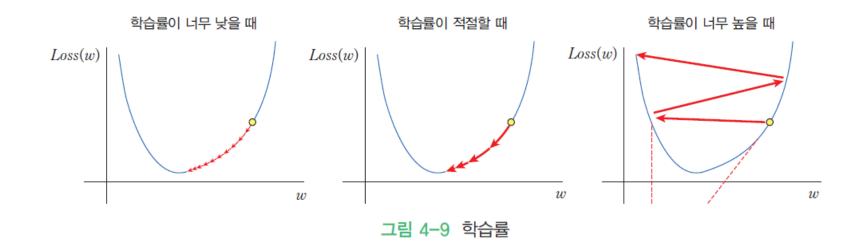
# 경사하강법



이것은 마치 산에서 내려오는 것과 유사합니다. 현재 위치에서 산의 기울기를 계산하여서 기울기의 반대 방향으로 이동하면산에서 내려오게 됩니다.







# 지역최소값 문제

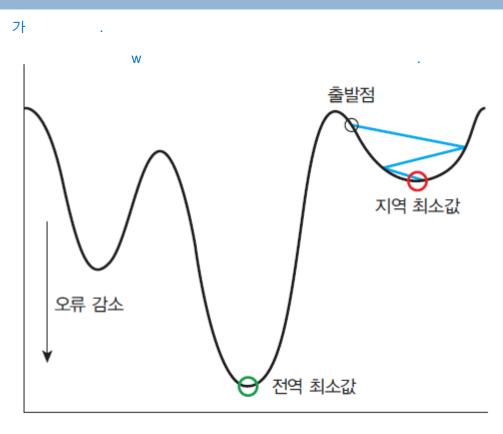


그림 4-10 지역 최소값과 전역 최소값

#### 선형 회귀에서 경사하강법

$$Loss(W, b) = \frac{1}{n} \sum_{i=1}^{n} ((Wx_i + b) - y_i)^2$$

$$\frac{\partial Loss\left(W,b\right)}{\partial W} = \frac{1}{n} \sum_{i=1}^{n} 2\left((Wx_{i} + b) - y_{i}\right)(x_{i}) = \frac{2}{n} \sum_{i=1}^{n} x_{i}((Wx_{i} + b) - y_{i})$$

$$\begin{split} \frac{\partial Loss\left(W,b\right)}{\partial b} &= \frac{2}{n} \sum_{i=1}^{n} ((Wx_i + b) - y_i) \\ W &= W - 0.01^* \frac{\partial Loss}{\partial W} \\ b &= b - 0.01^* \frac{\partial Loss}{\partial b} \end{split}$$

## 경사 하강법 구현

```
import numpy as np
import matplotlib.pyplot as plt
X = np.array([0.0, 1.0, 2.0])
y = np.array([3.0, 3.5, 5.5])
W = 0 # 기울기
b = 0 # 절편
lrate = 0.01 # 학습률
epochs = 1000 # 반복 횟수
n = float(len(X)) # 입력 데이터의 개수
# 경사 하강법
for i in range(epochs):
  y_pred = W*X + b # 예측값
  dW = (2/n) * sum(X * (y_pred-y))
  db = (2/n) * sum(y_pred-y)
  W = W - lrate * dW # 기울기 수정
  b = b - lrate * db # 절편 수정
```

```
# 기울기와 절편을 출력한다.
print (W, b)

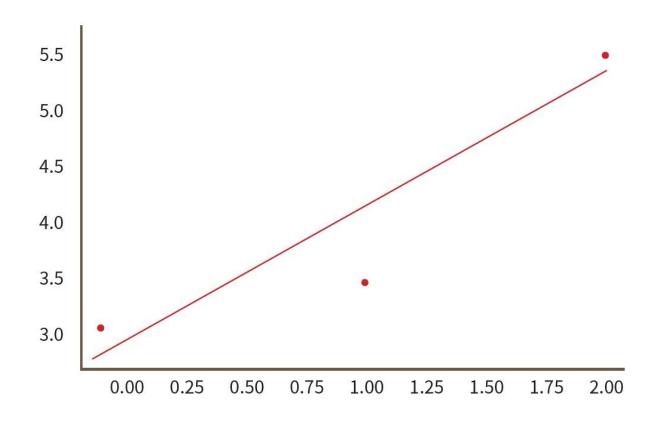
# 예측값을 만든다.
y_pred = W*X + b

# 입력 데이터를 그래프 상에 찍는다.
plt.scatter(X, y)

# 예측값은 선그래프로 그린다.
plt.plot([min(X), max(X)], [min(y_pred), max(y_pred)], color='red')
plt.show()
```

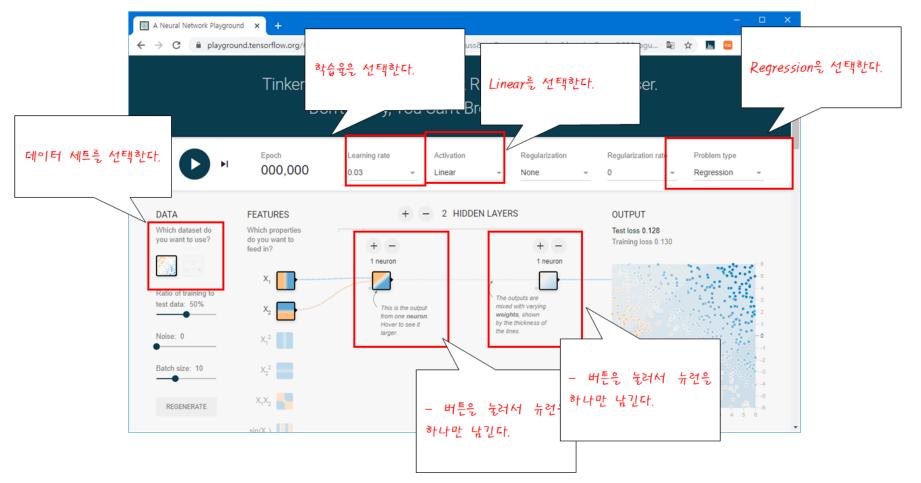
1.2532418085611319 2.745502230882486

#### 경사 하강법 구현

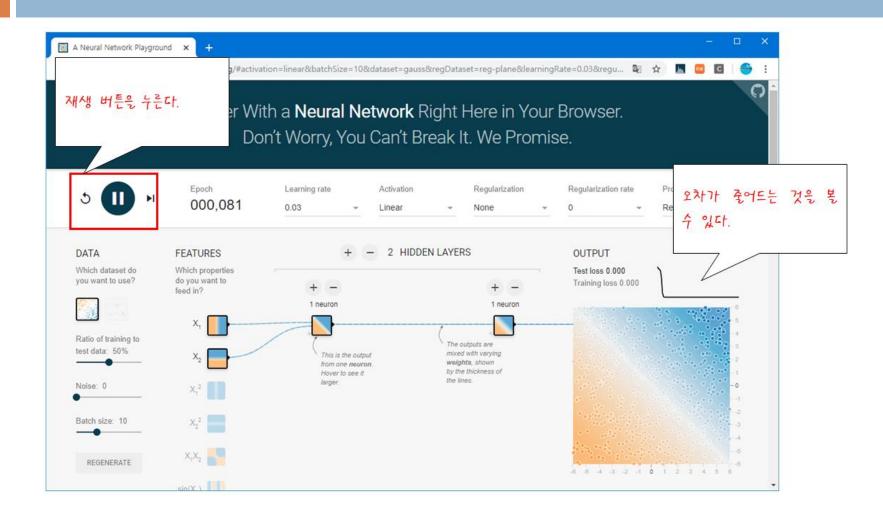


## Lab: 학습률 실습

 구글의 텐서 플로우 플레이그라운드는 이주 유용한 사이트 (https://playground.tensorflow.org)이다.



# Lab: 학습률 실습



# 선형 회귀 예제

이번 절에서는 아나콘다에 포함되어 있는 Scikit-Learn 라이브러리를 사용하여 회귀 함수를 구현하는 방법을 살펴본다.

```
import matplotlib.pylab as plt from sklearn import linear_model

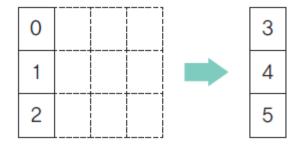
# 선형 회귀 모델을 생성한다.
reg = linear_model.LinearRegression()

# 데이터는 파이썬의 리스트로 만들어도 되고 아니면 넘파이의 배열로 만들어도 됨
X = [[0], [1], [2]] # 2차원으로 만들어야 함
y = [3, 3.5, 5.5] # y = x + 3

# 학습을 시킨다.
reg.fit(X, y)
```

#### 학습 데이터 만들기

학습 데이터는 반드시 2차원 배열이어야 한다(한 열만 있어도 반드시 2차원 배열 형태로 만들어야 한다). 따라서 리스트의 리스트를 만들어서 다음과 같은 2차원 배열을 생성한다.



# 선형 회귀 예제

```
>>> reg.coef_ # 직선의 기울기
array([1.25])

>>> reg.intercept_ # 직선의 y-절편
2.750000000000004

>>> reg.score(X, y)
0.8928571428571429

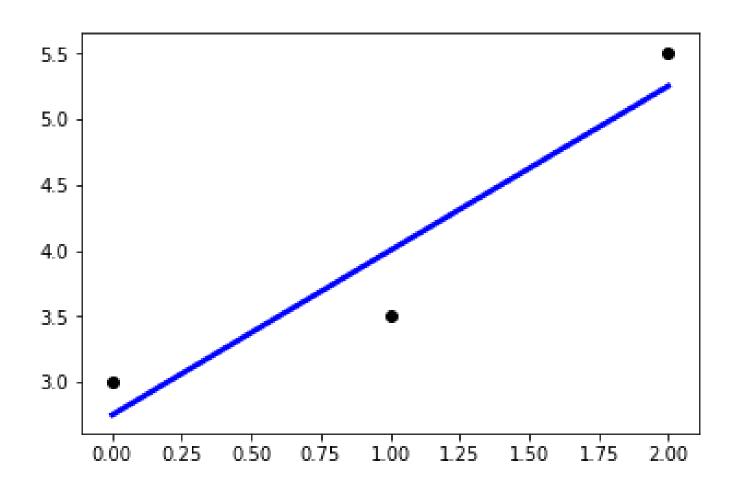
>>> reg.predict([[5]])
array([8.])
```

# **グ**그래프를 그려보자.

```
# 학습 데이터와 y 값을 산포도로 그린다.
plt.scatter(X, y, color='black')

# 학습 데이터를 입력으로 하여 예측값을 계산한다.
y_pred = reg.predict(X)

# 학습 데이터와 예측값으로 선그래프로 그린다.
# 계산된 기울기와 y 절편을 가지는 직선이 그려진다.
plt.plot(X, y_pred, color='blue', linewidth=3)
plt.show()
```



#### Lab: 선형 회귀 실습

인간의 키와 몸무게는 어느 정도 비례할 것으로 예상된다. 아래와 같은 데이터가 있을 때, 선형 회귀를 이용하여 학습시키고 키가 165cm일 때의예측값을 얻어보자. 파이썬으로 구현하여 본다.

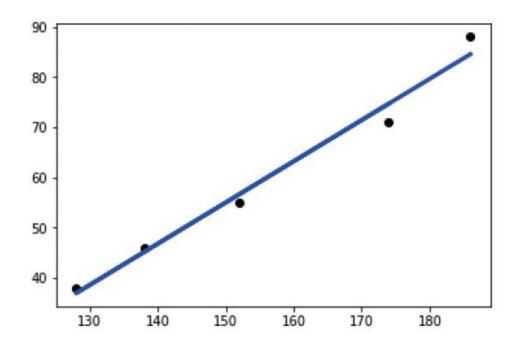




```
import matplotlib.pylab as plt
from sklearn import linear_model
reg = linear_model.LinearRegression()
X = [[174], [152], [138], [128], [186]]
y = [71, 55, 46, 38, 88]
reg.fit(X, y)
                                 # 학습
print(reg.predict([[165]]))
# 학습 데이터와 y 값을 산포도로 그린다.
plt.scatter(X, y, color='black')
# 학습 데이터를 입력으로 하여 예측값을 계산한다.
y_pred = reg.predict(X)
# 학습 데이터와 예측값으로 선그래프로 그린다.
#계산된 기울기와 y 절편을 가지는 직선이 그려진다.
plt.plot(X, y_pred, color='blue', linewidth=3)
plt.show()
```



#### [67.30998637]





- 과잉 적합(overfitting)이란 학습하는 데이터에서는 성능이 뛰어나지만 새로운 데이터(일반화)에 대해서는 성능이 잘 나오지 않는 모델을 생성하 는 것이다.
- 과소 적합(underfitting)이란 학습 데이터에서도 성능이 좋지 않은 경우이다. 이 경우에는 모델 자체가 적합지 않은 경우가 많다. 더 나은 모델을 찾아야 한다.

```
      가가
      .
      .
      .
      .
      .
      .
      .
      가가
      .
      가가
      .
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가
      가</td
```

#### 과잉 적합 vs 과소 적합

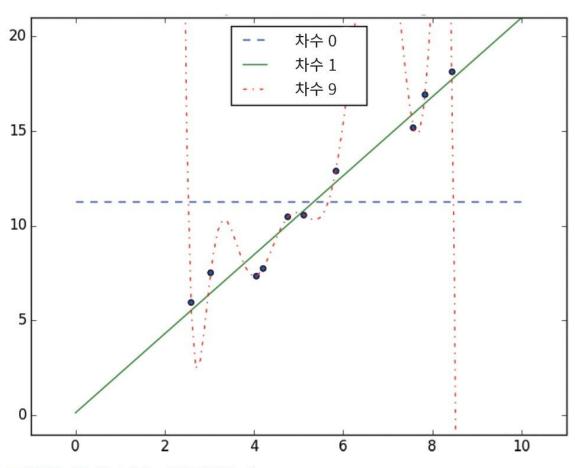


그림 10-5 회귀에서 과잉 적합의 예

## Lab: 당뇨병 예제

 sklearn 라이브러리에는 당뇨병 환자들의 데이터가 기본적으로 포함되어 있다.

특징(10개)											
	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	혈당
데이터 개수 (442)											
개수 ≺											
(442)											•••

# 선형 회귀 예제

```
import matplotlib.pylab as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn import datasets
# 당뇨병 데이터 세트를 적재한다.
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
# 하나의 특징(BMI)만 추려내서 2차원 배열로 만든다. BMI 특징의 인덱스가 2이다.
diabetes X new = diabetes X[:, np.newaxis, 2]
# 학습 데이터와 테스트 데이터를 분리한다.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(diabetes_X_new, diabetes_y,
test size=0.1, random state=0)
```

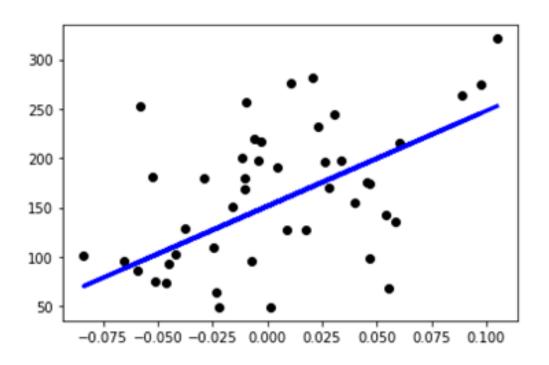
# 선형 회귀 예제

```
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

# 테스트 데이터로 예측해보자.
y_pred = model.predict(X_test)

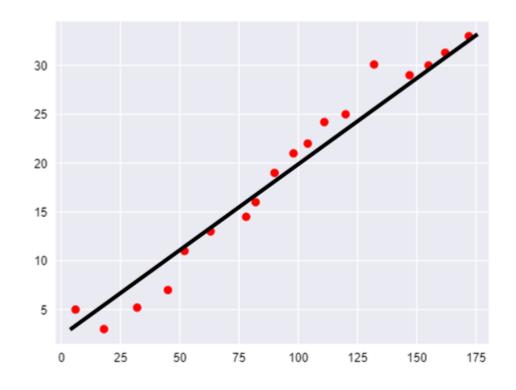
# 실제 데이터와 예측 데이터를 비교해보자.
plt.plot(y_test, y_pred, '.')

plt.scatter(X_test, y_test, color='black')
plt.plot(X_test, y_pred, color='blue', linewidth=3)
plt.show()
```



#### Mini Project: 면적에 따른 집값 예측

사용자가 아파트 면적을 입력하면 아파트의 가격이 출력되는 시스템을 만들어보자. 아파트 면적과 가격은 모두 실수이므로 기계 학습의 방법 중에서 선형 회귀를 사용하여야 한다





- 지도 학습에는 회귀(regression)와 분류(classification)가 있었다. 전자 는 연속적인 값을 예측하고 후자는 입력을 어떤 카테고리 중의 하나로 예 측한다.
- 선형 회귀는 입력 데이터를 가장 잘 설명하는 직선의 기울기와 절편값을 찾는 문제이다.
- 손실 함수(loss function)란 실제 데이터와 직선 간의 차이를 제곱한 값이다.
- 회귀란 손실 함수를 최소로 하는 직선의 기울기와 절편값을 계산하는 것이다.
- 손실 함수의 값이 작아지는 방향을 알려면 일반적으로 경사 하강법 (gradient descent method)과 같은 방법을 많이 사용한다.