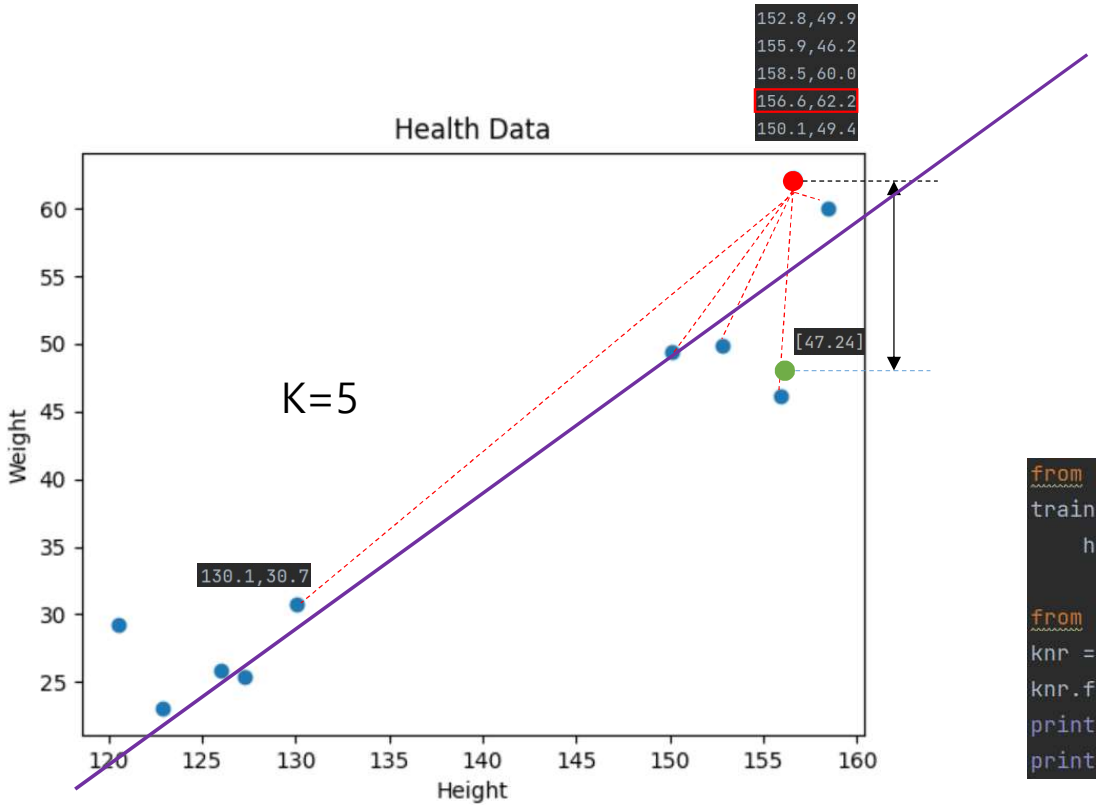
선형회귀

이웃회귀



	H
0	130.1
1	120.5
2	127.3
3	122.9
4	126.0
5	152.8
6	155.9
7	158.5
8	156.6
9	150.1

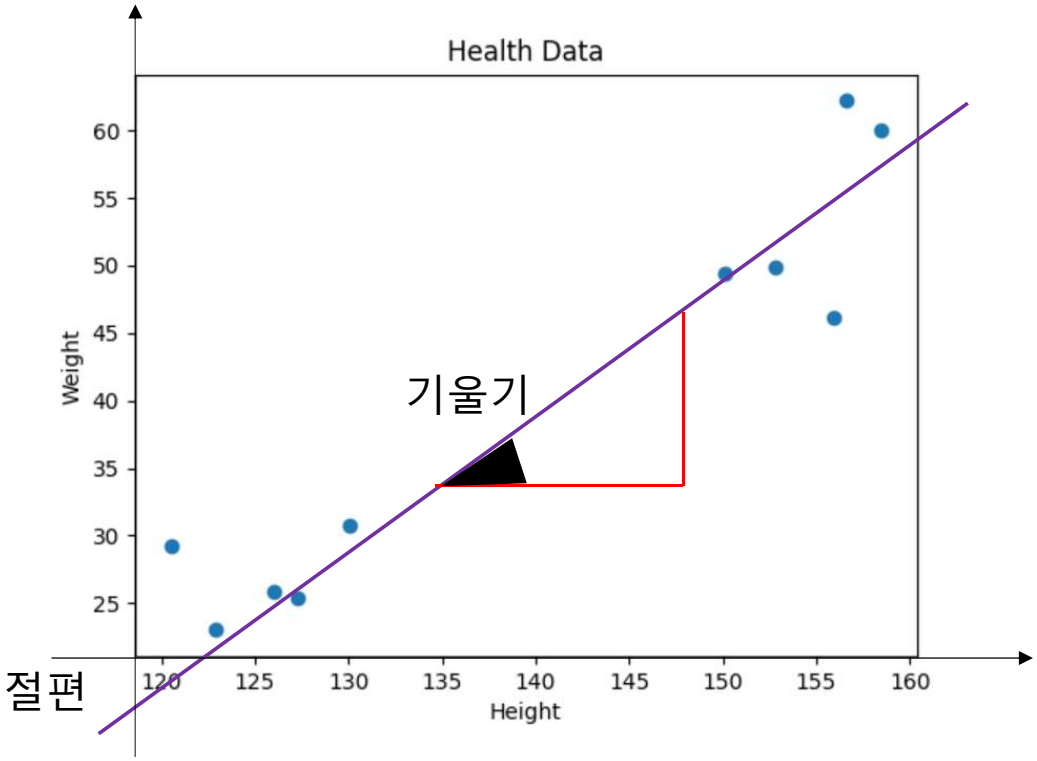
	W
0	30.7
1	29.2
2	25.4
3	23.0
4	25.8
5	49.9
6	46.2
7	60.0
8	62.2
9	49.4

```
from sklearn.model_selection import train_test_split
train_data, test_data, train_target, test_target = train_test_split(
    h, w, test_size = 0.2, random_state = 42)

from sklearn.neighbors import KNeighborsRegressor
knr = KNeighborsRegressor()
knr.fit(train_data, train_target)
print("Train-Eval:", knr.score(train_data, train_target))
print("Test-Eval :", knr.score(test_data, test_target))
```

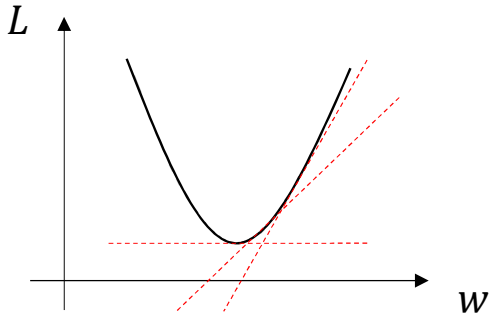
선형(Linear)회귀

- 가장 직관적인고 간단한 직선으로 독립변수와 종속변수의 관계를 모델링하는 기법
- 주어진 데이터로부터 독립변수와 종속변수의 관계를 가장 잘 나타내는 직선 방정식을 찾는 것



$$y' = h(x) = wx + b$$

$$L(y, h(x)) = \frac{1}{m}\sum(h(x) - y)^2$$



sklearn.linear_model.LinearRegression

class sklearn.linear_model.LinearRegression(*, *fit_intercept=True*, *normalize='deprecated'*, *copy_X=True*, *n_jobs=None*, *positive=False*) [source]

Ordinary least squares Linear Regression.

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> # y = 1 * x_0 + 2 * x_1 + 3
>>> y = np.dot(X, np.array([1, 2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
1.0
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
3.0...
>>> reg.predict(np.array([[3, 5]]))
array([16.])
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

coef_ : *array of shape (n_features,) or (n_targets, n_features)*

Estimated coefficients for the linear regression problem.

intercept_ : *float or array of shape (n_targets,)*

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression

선형회귀, 이웃회귀 정확도

```python
import pandas as pd
data3 = pd.read_csv("data/health.csv")
h = data3[["H"]]
w = data3[["W"]]

from sklearn.model_selection import train_test_split
train_data, test_data, train_target, test_target = train_test_split(
    h, w, test_size = 0.2, random_state = 42)

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(train_data, train_target)
print("Train-Eval:", lr.score(train_data, train_target))
print("Test-Eval :", lr.score(test_data, test_target))
```

```
Train-Eval: 0.9470530489315999
Test-Eval : 0.7265384110806861
```

선형회귀

```python
import pandas as pd
data3 = pd.read_csv("data/health.csv")
h = data3[["H"]]
w = data3[["W"]]
```

```python
from sklearn.model_selection import train_test_split
train_data, test_data, train_target, test_target = train_test_split(
    h, w, test_size = 0.2, random_state = 42)

from sklearn.neighbors import KNeighborsRegressor
knr = KNeighborsRegressor()
knr.fit(train_data, train_target)
print("Train-Eval:", knr.score(train_data, train_target))
print("Test-Eval :", knr.score(test_data, test_target))
```

```
Train-Eval: 0.7913472691260918
Test-Eval : 0.5839169880624423
```

이웃회귀

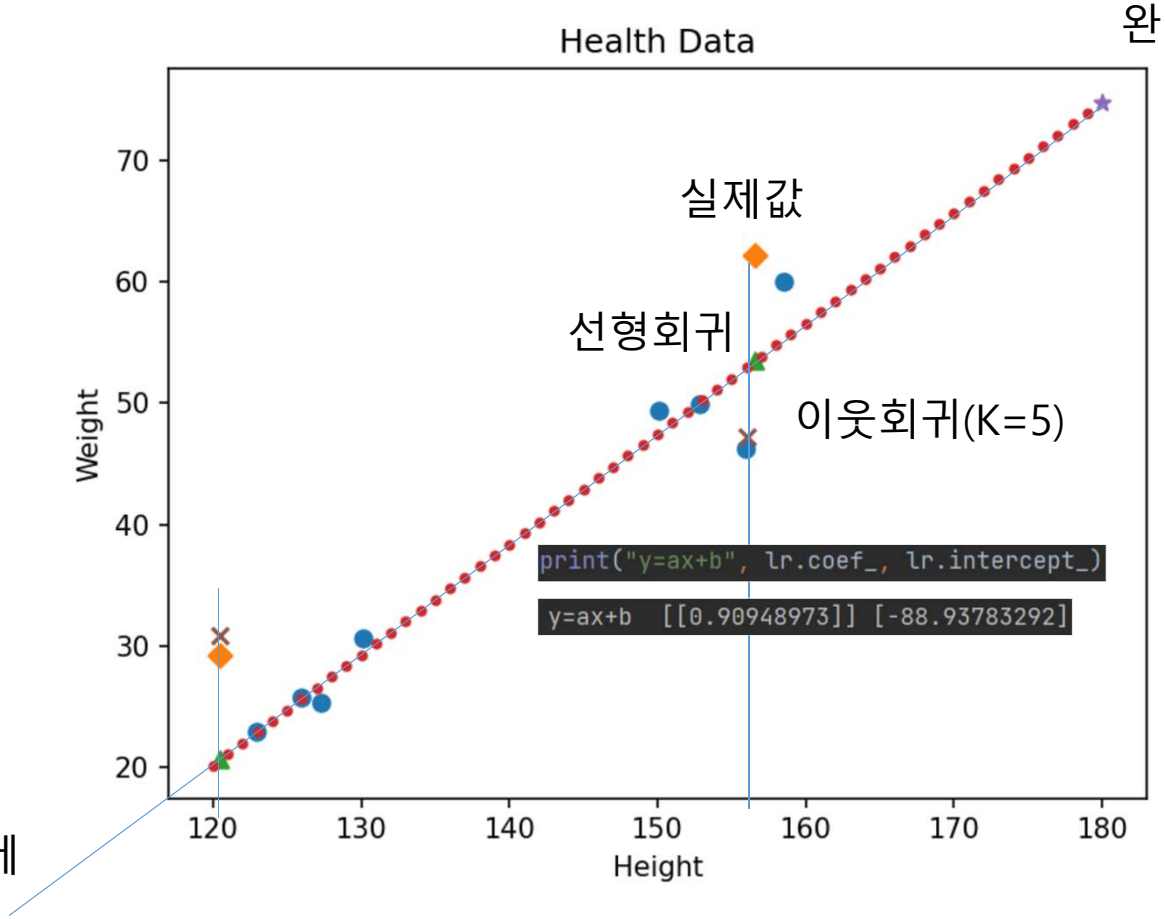
선형회귀, 이웃회귀 예측

```
test_pred = lr.predict(test_data)
```

완전 새로운
데이터

	H
8	156.6
1	120.5

테스트 데이터



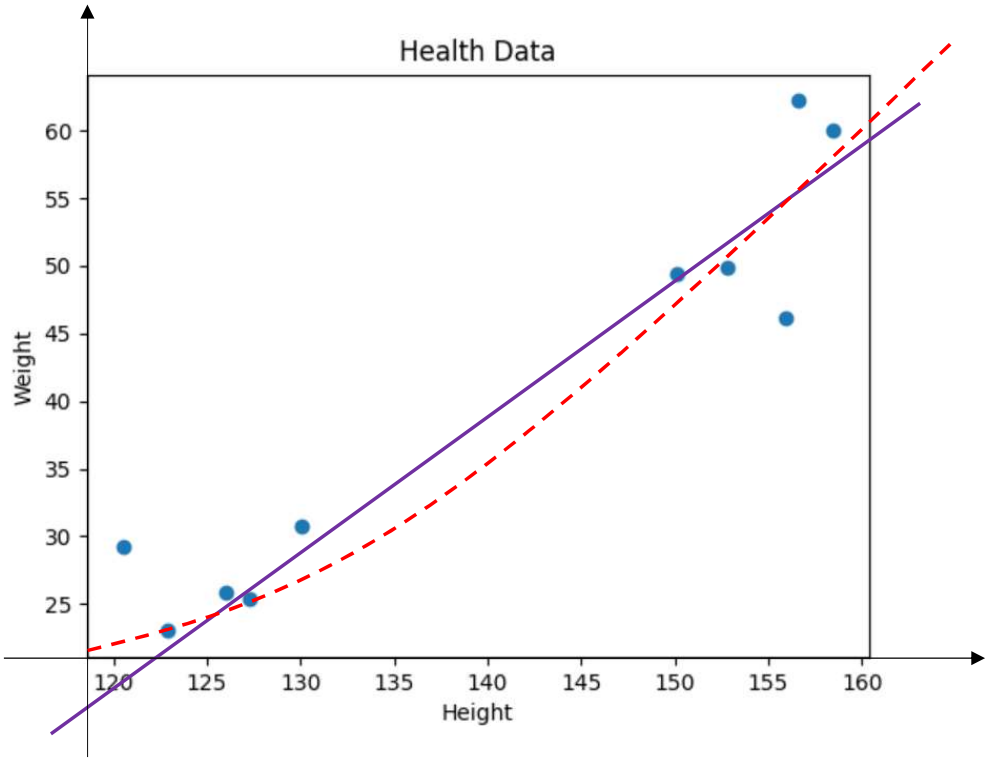

```
print("y=ax+b", lr.coef_, lr.intercept_)
```

```
y=ax+b  [[0.90948973]] [-88.93783292]
```

음수
몸무게

다항(Polynomial)회귀

- 직선이 아닌 곡선으로 회귀 문제를 해결
- 차수가 높아질수록 모델 복잡도도 높아지기 때문에, 학습 데이터에 과대적합의 위험성 존재



$$y' = w_1 x^2 + w_2 x + c$$

$$y' = w_1 x + b$$

다항회귀 정확도

```python
import numpy as np
train_poly = np.column_stack((train_data ** 2, train_data))
test_poly = np.column_stack((test_data ** 2, test_data))
# print(train_poly.shape, test_poly.shape)

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(train_poly, train_target)
print("Train-Eval:", lr.score(train_poly, train_target))
print("Test-Eval :", lr.score(test_poly, test_target))
```

이웃회귀

```
Train-Eval: 0.7913472691260918
Test-Eval : 0.5839169880624423
```

선형회귀

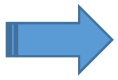
```
Train-Eval: 0.9470530489315999
Test-Eval : 0.7265384110806861
```

다항회귀

```
Train-Eval: 0.947167928120013
Test-Eval : 0.740634032634665
```

	H
5	152.8
0	130.1
7	158.5
2	127.3
9	150.1
4	126.0
3	122.9
6	155.9

	H
8	156.6
1	120.5



```
(8, 2) (2, 2)
```

```
[[23347.84   152.8 ]
 [16926.01   130.1 ]
 [25122.25   158.5 ]
 [16205.29   127.3 ]
 [22530.01   150.1 ]
 [15876.     126.  ]
 [15104.41   122.9 ]
 [24304.81   155.9 ]]
```

```
[[24523.56   156.6 ]
 [14520.25   120.5 ]]
```

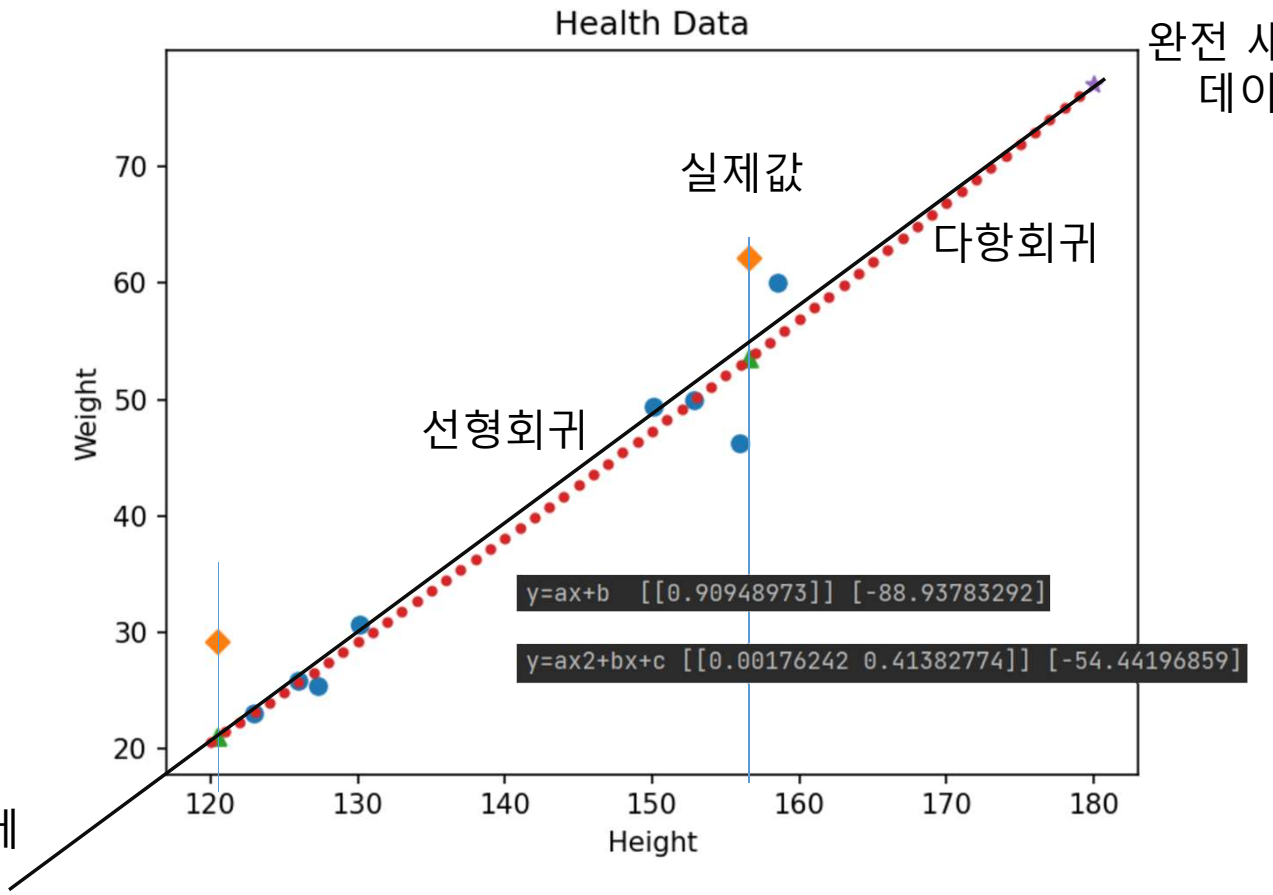
다항회귀 예측

```
test_pred = lr.predict(test_poly)
```

완전 새로운
데이터

```
        H
8   156.6
1   120.5
```

테스트 데이터



음수
몸무게

다중(Multiple)회귀

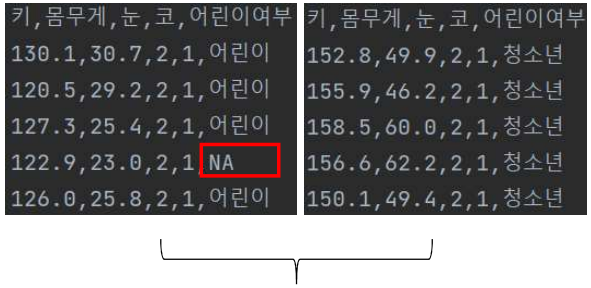
- 독립변수가 아닌 다중변수를 이용 (여러 개의 특징을 사용하여 회귀문제를 해결)
- 의미없는 특징을 추가할 경우, 모델 복잡도만 높아짐
- 기존의 특징들의 조합으로 새로운 특징 생성 가능
- 특성의 개수를 크게 늘리면, 학습 데이터에 과대적합의 위험성 존재

선형회귀 $y' = wx + b$

다항회귀 $y' = w_1 x^2 + w_2 x + b$

다중회귀 $y' = w_1 x_1 + w_2 x_2 + b$

특징: 키, 몸무게, 눈, 코



전체 데이터

규제(Regularization)

- 규제를 통해서 과대적합 방지 가능: 릿지회귀, 라쏘회귀
- 모델 파라미터(계수)의 크기를 작게 만들어 전체적인 영향력을 줄이는 역할
- 릿지회귀의 규제항은 L2-Norm, 라쏘회귀의 규제항 L1-Norm으로 설정
- 규제강도(α) 값이 크면 계수의 크기가 더 줄어들어 과소적합이 되도록 유도됨
- 규제강도(α) 값이 작으면 계수의 크기가 더 커지기 때문에 과대적합의 위험성이 있음

릿지회귀: $L(y, h(x)) = \frac{1}{m}\sum(h(x) - y)^2 + \alpha \sum w^2$

라쏘회귀: $L(y, h(x)) = \frac{1}{m}\sum(h(x) - y)^2 + \alpha \sum |w|$

참고자료

- 지능기전공학부 최유경 교수님 자료, https://github.com/sejongresearch/2021.MachineLearning
- 코랩(Colab), https://colab.research.google.com/
- 파이썬(Python), https://www.python.org/doc/
- 사이킷런(sckit-learn), https://scikit-learn.org/stable/index.html
- 판다스(pandas), https://pandas.pydata.org/
- 맷플롯립(matplotlib), https://matplotlib.org/
- 씨본(seaborn), https://seaborn.pydata.org/
- 캐글(Kaggle), https://www.kaggle.com/
- 넘파이(numpy), https://numpy.org/doc/stable/
- 스택오퍼플러우(stackoverflow), https://stackoverflow.com/