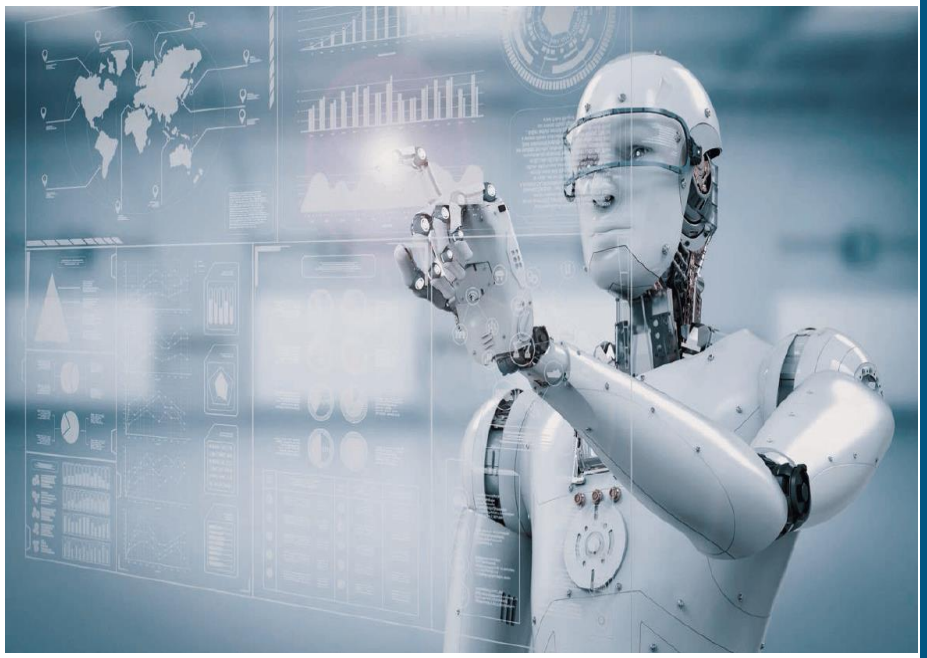


기계학습개론

- 회귀 알고리즘1 -

교수 이홍로

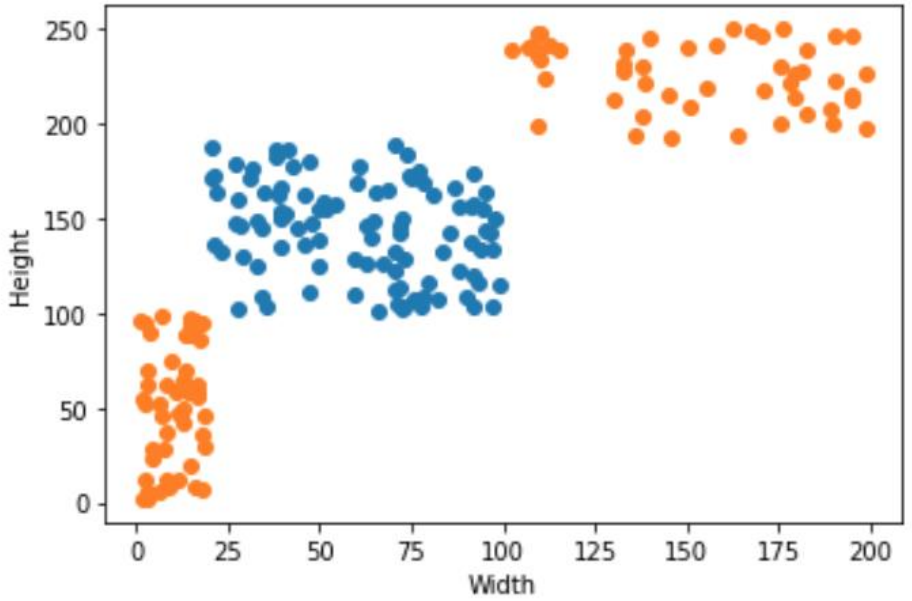
MP : 010-6611-3896
E-mail : hrlee@cnu.ac.kr
강의 홈페이지 : https://cyber.hanbat.ac.kr/

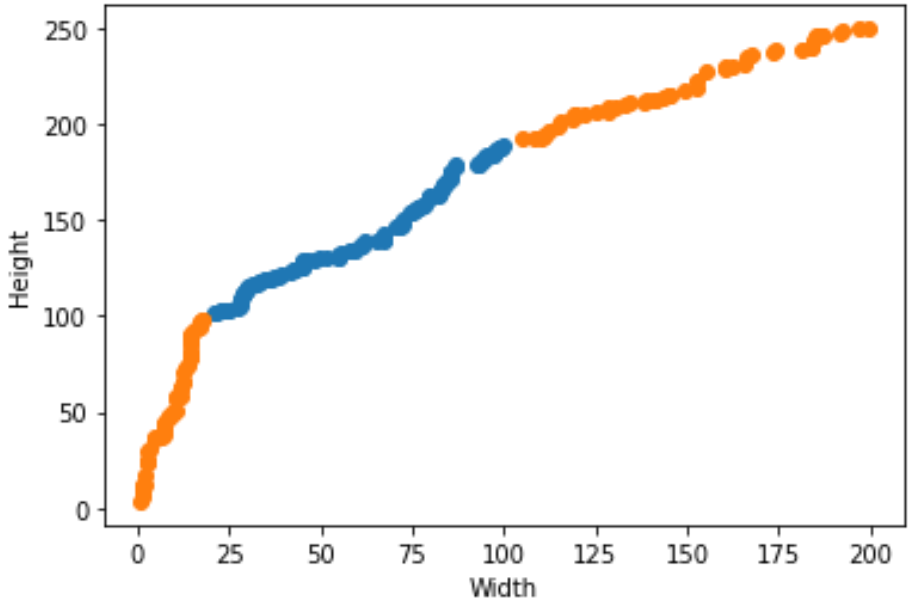
오늘의 강의 목표

- 데이터 정제
- Train Set 및 Test Set 생성하기
- K-최근접 이웃 회귀 알고리즘
- 선형 회귀 알고리즘
- 다항 회귀 알고리즘

데이터 정제

- 좀 더 현실적인 데이터로 수정





데이터 정제

- Sort 적용

```
normal_person_width = np.sort(np.random.uniform(low=20, high=100, size=100))
normal_person_height = np.sort(np.random.uniform(low=100, high=190, size=100))

error_person_width = np.sort(np.append(np.random.uniform(low= 1, high=19, size=50),
                                       np.random.uniform(low=101, high=200, size=50)))
error_person_height= np.sort(np.append(np.random.uniform(low=  1, high=99, size=50),
                                       np.random.uniform(low=191, high=250, size=50)))
```

데이터 정제

- Numpy 데이터로 수정

Numpy 2차원 array

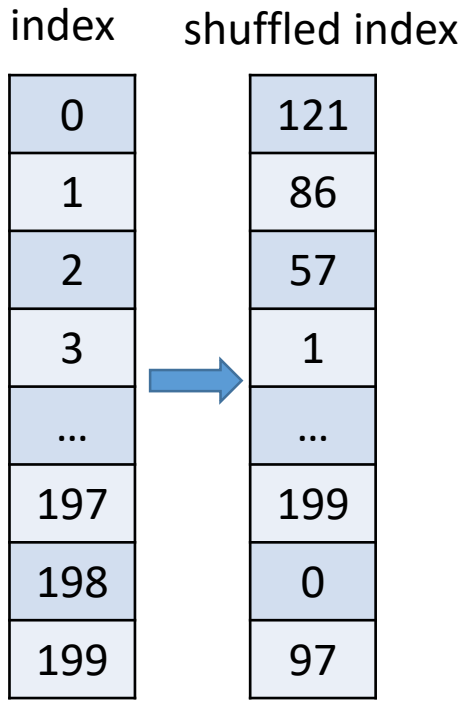
person_data [[1,2],[3,4],[5,6],…,[49,50]] → [[1 2]
[3 4]
[5 6]
…
[49 50]]

```
width = np.append(normal_person_width, error_person_width)
height = np.append(normal_person_height, error_person_height)
answer = [1]*100 + [0]*100
person_data = [[l, w] for l, w in zip(width, height)]

person_data = np.array(person_data)
answer = np.array(answer)
```

Train Set 및 Test Set 생성

- **Train Set(75%), Test Set(25%) 생성**

```
np.random.seed(42)
index = np.arange(200)
np.random.shuffle(index)

train_data = person_data[index[:150]]
train_answer = answer[index[:150]]

test_data = person_data[index[150:]]
test_answer = answer[index[150:]]
```



Height	Width	answer
220	50	0
200	55	0
190	54	0
175	52	1
160	100	1
⋮		⋮
155	45	1
150	50	1
50	10	0
45	44	0
32	31	0
20	53	0

```
from sklearn.model_selection import train_test_split

train_data, test_data, train_answer, test_answer =
train_test_split(person_data, answer, random_state=42)
```

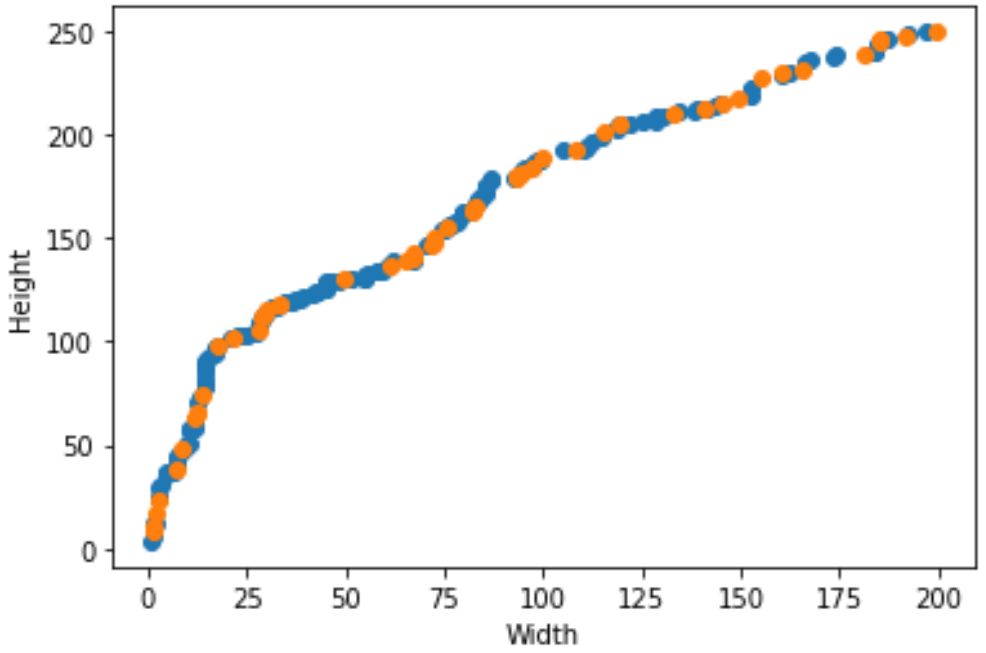
Train Set 75%

Test Set 25%

Train Set 및 Test Set 생성

- **Train Set 및 Test Set 출력**

```
plt.scatter(train_data[:,0], train_data[:,1])
plt.scatter(test_data[:,0], test_data[:,1])
plt.xlabel('Width')
plt.ylabel('Height')
plt.show()
```



Train Set 및 Test Set 생성

- 재 학습 후, 모델 평가하기

```
from sklearn.neighbors import KNeighborsClassifier


kn = KNeighborsClassifier()
kn.fit(train_data, train_answer)
kn.score(test_data, test_answer)
```

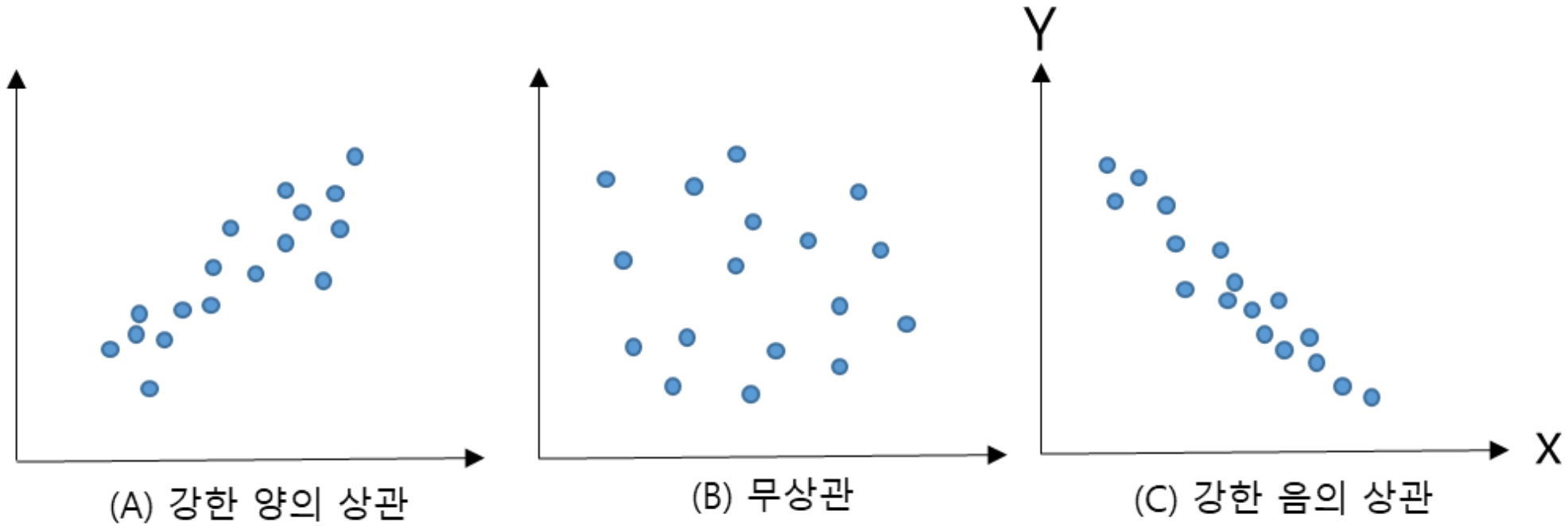
Train Set 및 Test Set 생성

- K 파라미터 찾기 알고리즘 수정

```
kn = KNeighborsClassifier()
kn.fit(train_data, train_answer)

for n in range(1, 100):
    kn.n_neighbors = n
    score = kn.score(test_data, test_answer)
    print(str(n)+":"+str(score))
```

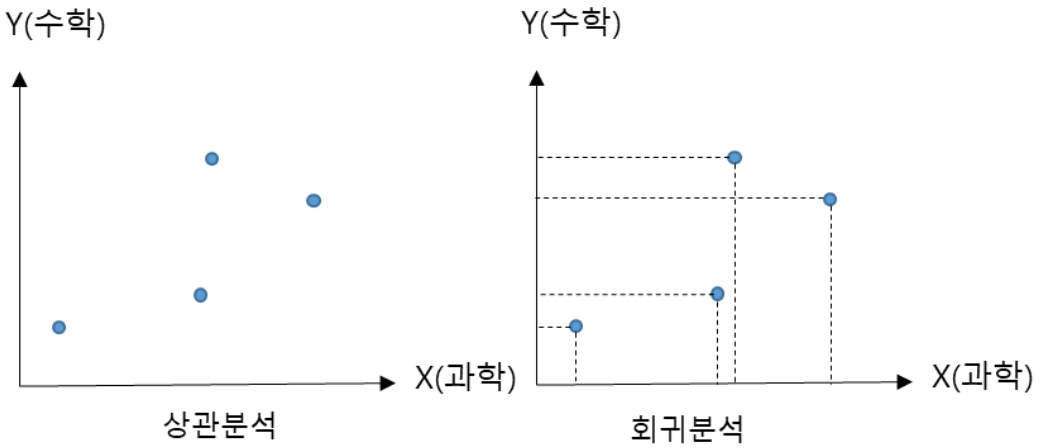
회귀 알고리즘

- **데이터의 상관 분석**



회귀 알고리즘

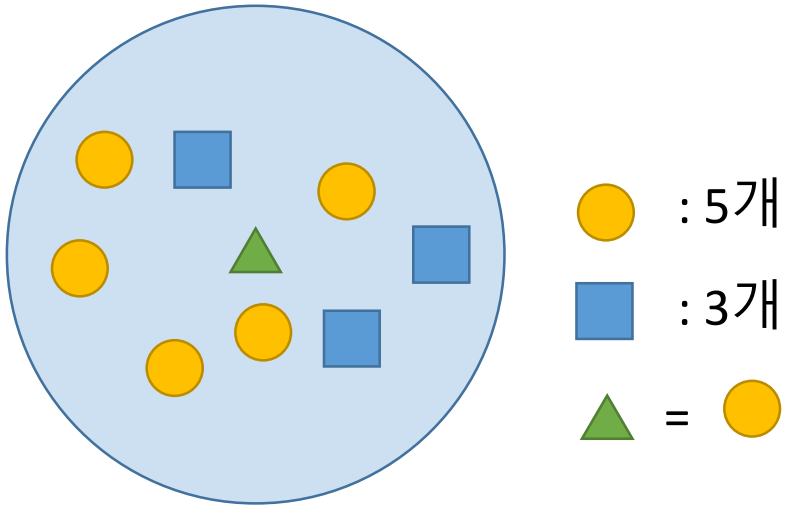
- **회귀 알고리즘이란?**
  - 두 연속형 변수 X와 Y를 독립변수와 종속변수라고 하는 인과관계로 설명할 수 있는 분석법
  - '과학 점수가 좋으면 수학점수가 좋을까요?' 와 같이 간단하지만 미래를 예측할 수 있는 머신러닝의 초기 모델이 됨



X	Y
독립변수, 설명변수, 원인변수	종속변수, 반응변수, 결과변수 머신러닝(클래스,라벨)
다른 변수에 영향을 주는 원인	다른 변수에 영향을 받는 결과
Dependent Variable Response	Independent Variable, Predictor Factor

K-최근접 이웃 회귀 알고리즘

- 분류와 예측의 차이점

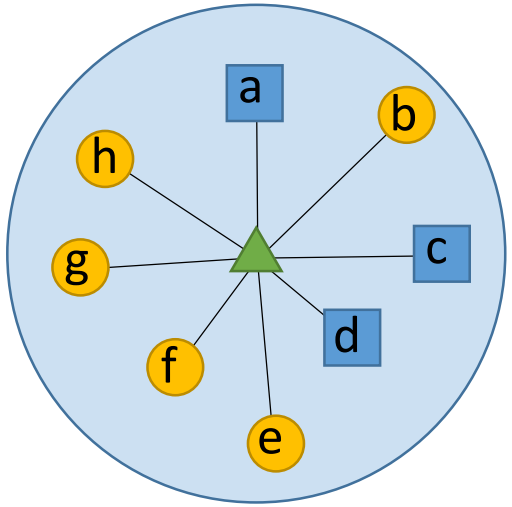


: 5개

: 3개

=

[분류]

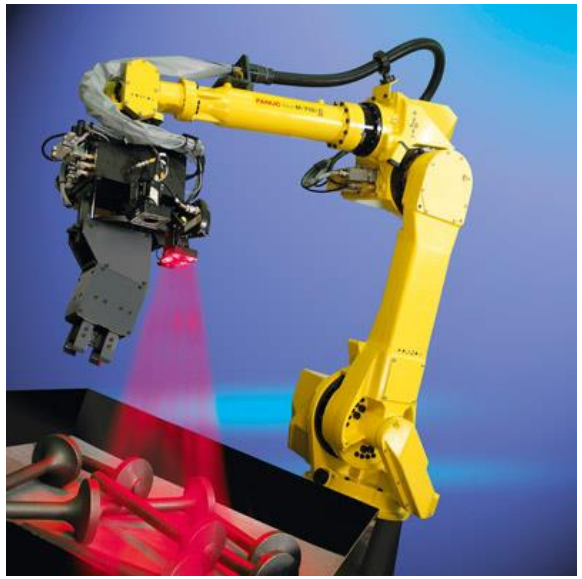


$$\triangle = \frac{a+b+c+d+e+f+g+h}{8}$$

[예측]

K-최근접 이웃 회귀 알고리즘

- **새로운 요구사항 : 분류가 아닌 예측**
  - 부품을 생산하는 공장에서 머신러닝을 활용하여 생산된 부품의 무게를 예측해야 하는 문제 발생



<요구사항>

- 30% 제품의 무게만 측정되었음
- 모든 제품의 무게를 측정하는 것은 시간적으로 불가능
- 크기에 따른 제품 무게를 예측하는 것이 필요

K-최근접 이웃 회귀 알고리즘

- 데이터 준비

```
import numpy as np
import matplotlib.pyplot as plt

product_width = np.array(
    [8.4, 13.7, 15.0, 16.2, 17.4, 18.0, 18.7, 19.0, 19.6, 20.0,
     21.0, 21.0, 21.0, 21.3, 22.0, 22.0, 22.0, 22.0, 22.0, 22.5,
     22.5, 22.7, 23.0, 23.5, 24.0, 24.0, 24.6, 25.0, 25.6, 26.5,
     27.3, 27.5, 27.5, 27.5, 28.0, 28.7, 30.0, 32.8, 34.5, 35.0,
     36.5, 36.0, 37.0, 37.0, 39.0, 39.0, 39.0, 40.0, 40.0, 40.0,
     40.0, 42.0, 43.0, 43.0, 43.5, 44.0]
     )
```

K-최근접 이웃 회귀 알고리즘

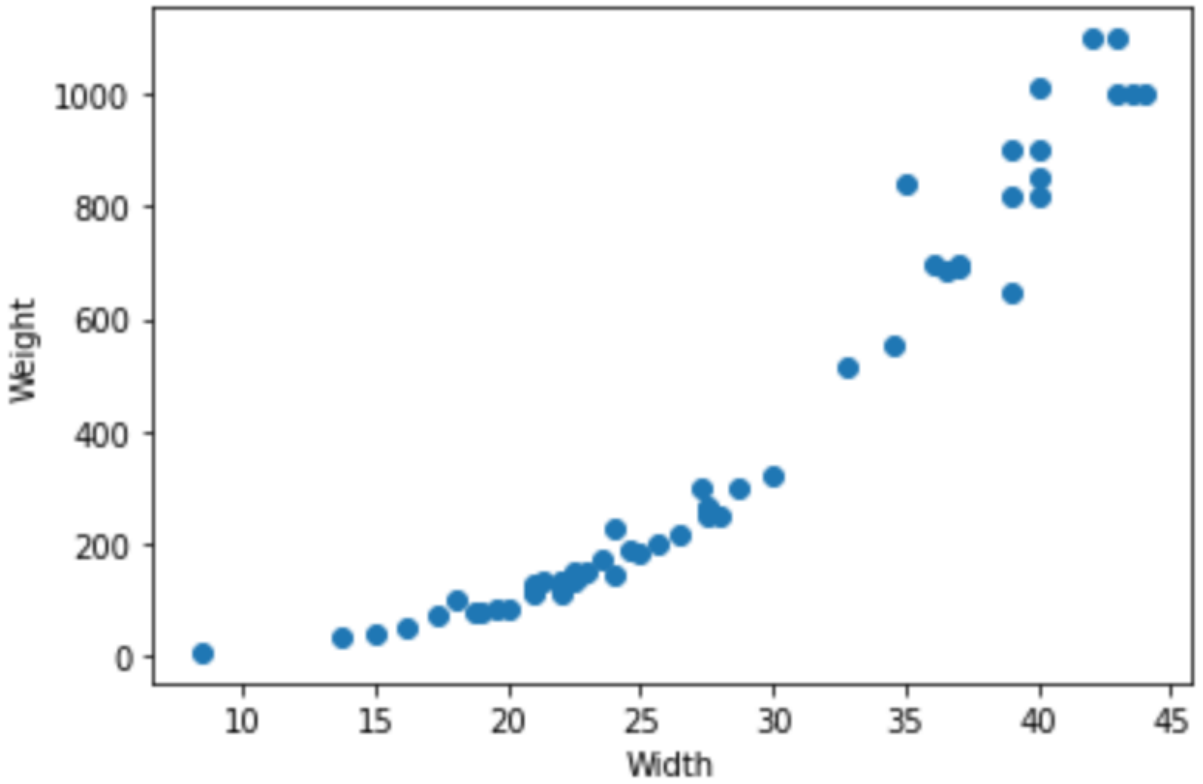
- 데이터 준비

```
product_weight = np.array(
    [5.9, 32.0, 40.0, 51.5, 70.0, 100.0, 78.0, 80.0, 85.0, 85.0,
     110.0, 115.0, 125.0, 130.0, 120.0, 120.0, 130.0, 135.0, 110.0,
     130.0, 150.0, 145.0, 150.0, 170.0, 225.0, 145.0, 188.0, 180.0,
     197.0, 218.0, 300.0, 260.0, 265.0, 250.0, 250.0, 300.0, 320.0,
     514.0, 556.0, 840.0, 685.0, 700.0, 700.0, 690.0, 900.0, 650.0,
     820.0, 850.0, 900.0, 1015.0, 820.0, 1100.0, 1000.0, 1100.0,
     1000.0, 1000.0]
     )
```

K-최근접 이웃 회귀 알고리즘

- 데이터 산점도 출력

```
plt.scatter(product_width, product_weight)
plt.xlabel('Width')
plt.ylabel('Weight')
plt.show()
```



K-최근접 이웃 회귀 알고리즘

- 사이킷런을 이용하여 훈련 및 테스트 데이터 생성하기

```
from sklearn.model_selection import train_test_split

train_data, test_data, train_answer, test_answer =
train_test_split(product_width, product_weight,
random_state=42)

train_data = train_data.reshape(-1,1)
test_data = test_data.reshape(-1,1)
```

train 75% (42)

Width	answer
50	0
55	0
54	0
52	1
100	1
⋮	⋮
45	1
50	1
10	0
44	0
31	0
53	0

test 25% (14)

K-최근접 이웃 회귀 알고리즘

- K-최근접 이웃 회귀 알고리즘을 적용하여 학습시키기(모델생성)

```
from sklearn.neighbors import KNeighborsRegressor


knr = KNeighborsRegressor()
knr.fit(train_data, train_answer)
knr.score(test_data, test_answer)
```

0.992809406101064

K-최근접 이웃 회귀 알고리즘

- 정답과 예측결과의 오차 계산하기

```
from sklearn.metrics import mean_absolute_error


test_prediction = knr.predict(test_data)


mae = mean_absolute_error(test_answer, test_prediction)
print(mae)
```

```
19.157142857142862
```

K-최근접 이웃 회귀 알고리즘

- **Training Set로 모델 평가해 보기 (과대적합 및 과소적합 분석)**

```
print(knr.score(train_data, train_answer))     0.9698823289099254
```

- **모델 평가 : Training Set vs. Test Set**

Test Set `knr.score(test_data, test_answer)` `0.992809406101064`

Training Set `knr.score(train_data, train_answer)` `0.9698823289099254`

- **K파라미터 수정**

knr.n_neighbors = 3

Test Set `knr.score(test_data, test_answer)` `0.9804899950518966`

Training Set `knr.score(train_data, train_answer)` `0.9746459963987609`

K-최근접 이웃 회귀 알고리즘

- **Width 50cm를 넣어 Weight 예측해 보기**

```
print(knr.predict([[50]]))
```

[1033.33333333] 1033g으로 예측, 그러나 실제적인 무게는 1100g

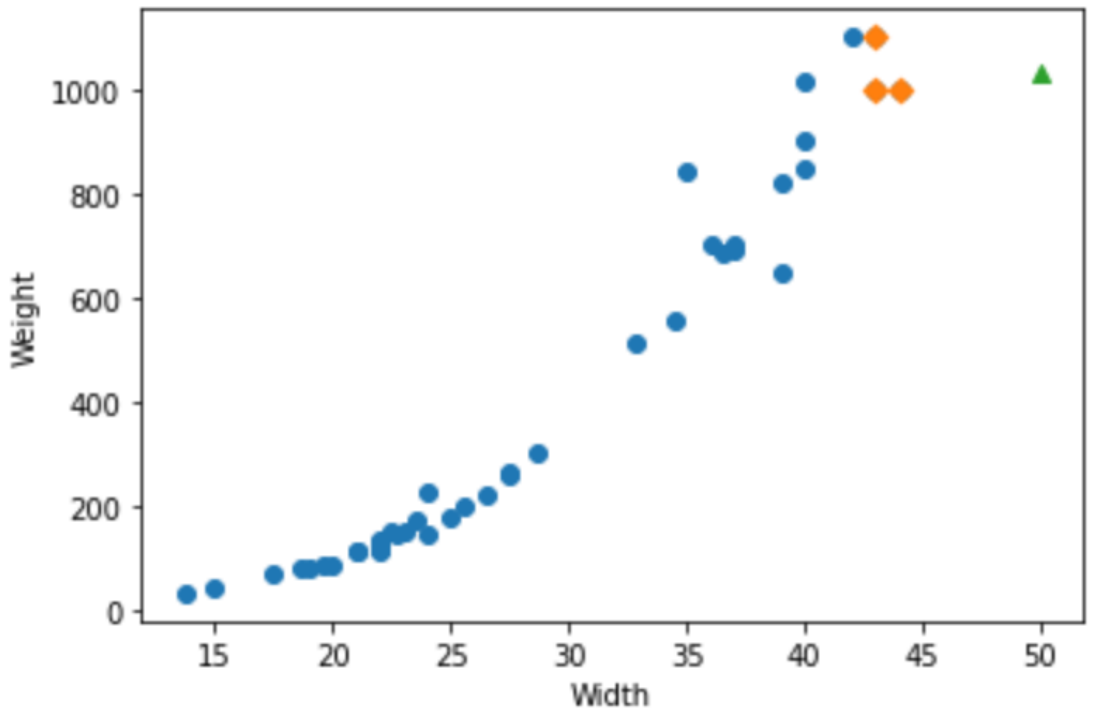
- **50cm 제품의 최근접 이웃들을 산점도에 표시해 보기**

```
import matplotlib.pyplot as plt

distances, indexes = knr.kneighbors([[50]])

plt.scatter(train_data, train_answer)
plt.scatter(train_data[indexes], train_answer[indexes], marker='D')

plt.scatter(50, 1033, marker='^')
plt.xlabel('Width')
plt.ylabel('Weight')
plt.show()
```

K-최근접 이웃 회귀 알고리즘

- **50cm 제품의 최근접 이웃들을 산점도에 표시해 보기**

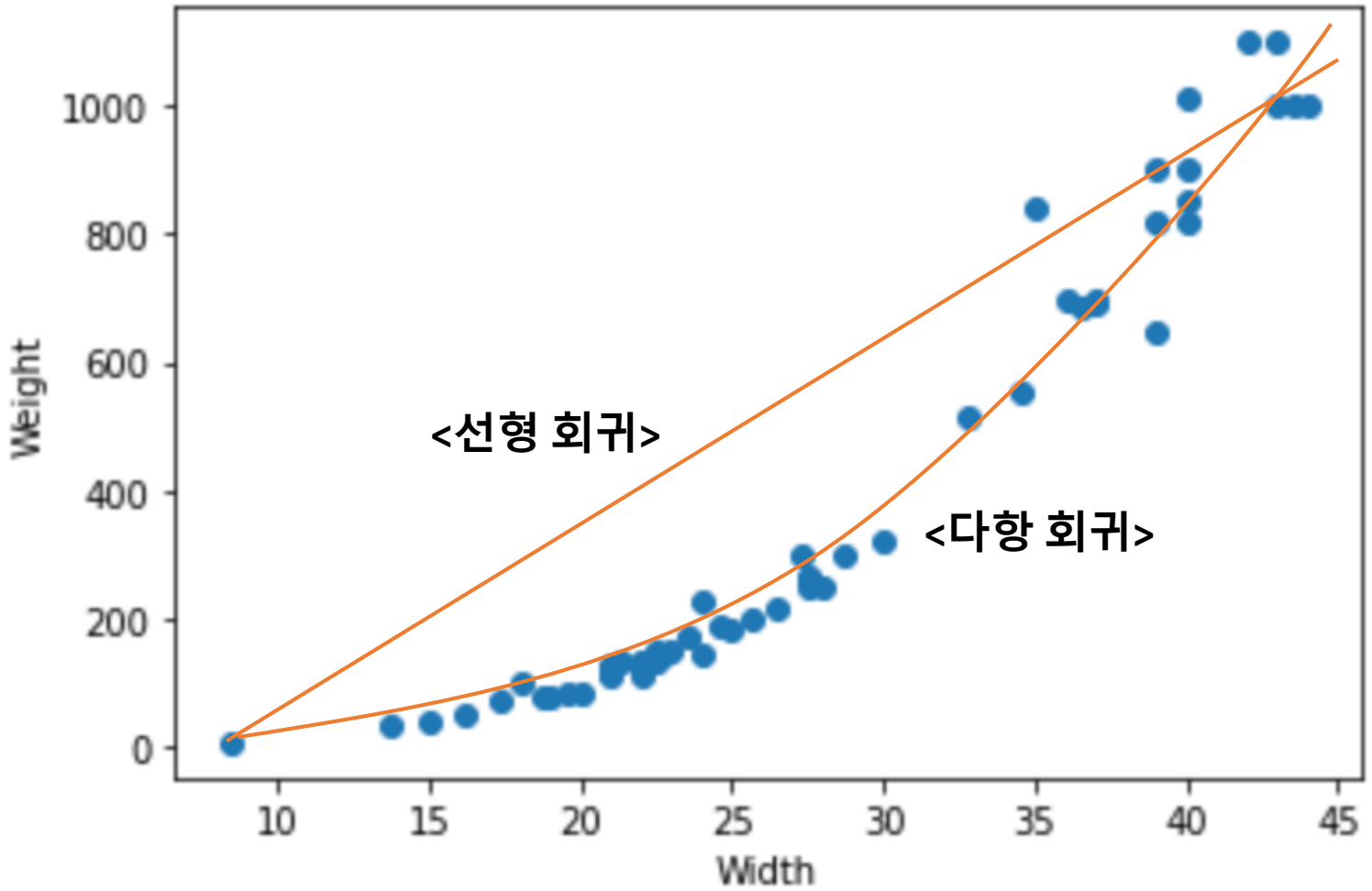
```
import matplotlib.pyplot as plt


distances, indexes = knr.kneighbors([[50]])


plt.scatter(train_data, train_answer)
plt.scatter(train_data[indexes],
train_answer[indexes], marker='D')


plt.scatter(50, 1033, marker='^')
plt.xlabel('Width')
plt.ylabel('Weight')
plt.show()
```



아무리 커져도 무게가 늘지 않음
K-최근접 이웃 회귀 알고리즘으로 예측하기에는 부적합

선형 회귀 알고리즘

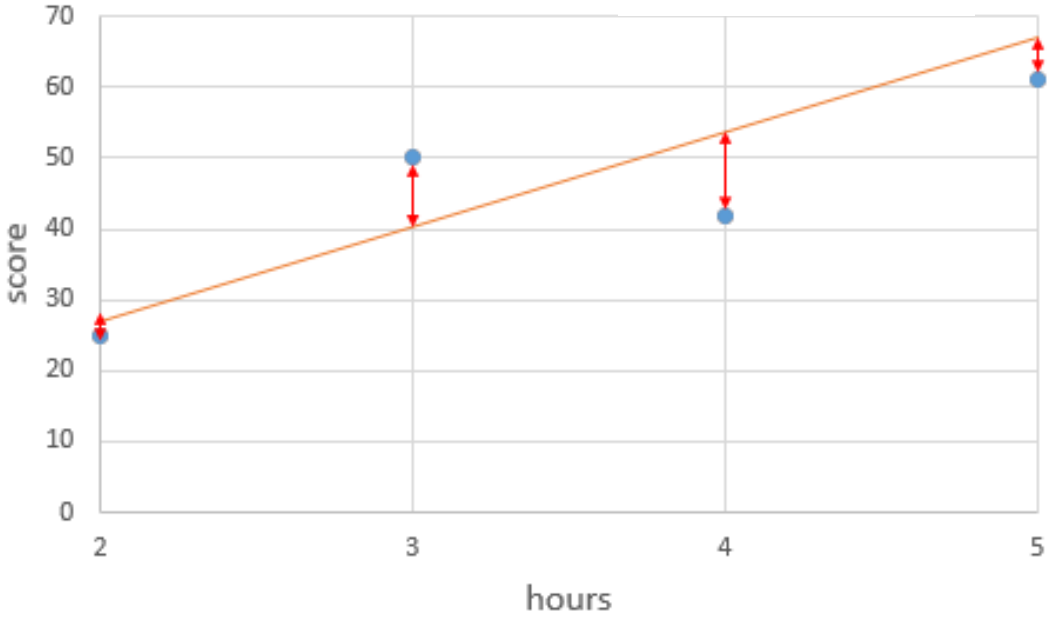
- 다양한 회귀 알고리즘



선형 회귀 알고리즘

- **원리**
  - 주어진 데이터를 대표하는 하나의 직선을 찾는 것
  - 회귀선, 회귀식으로 구성

$$y = wx + b \qquad H(x) = wx + b$$



hours(x)	2	3	4	5
실제값	25	50	42	61
예측값	27	40	53	66
오차	-2	10	-9	-5

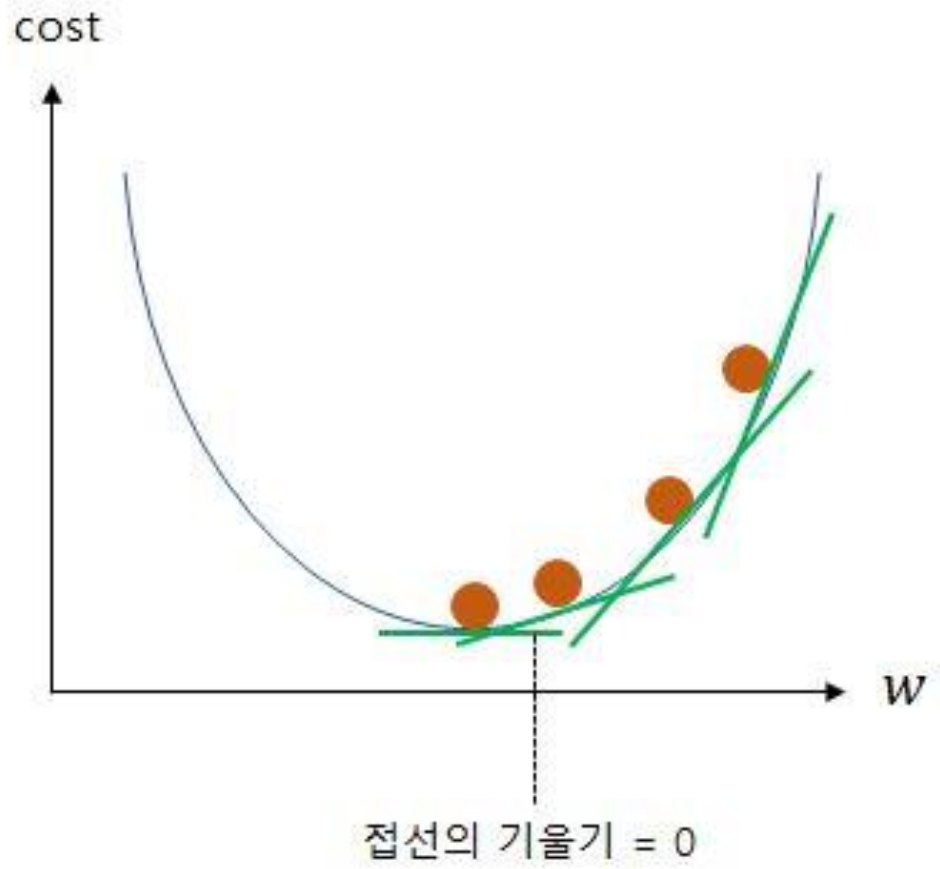
$$\sum_{i=1}^{n} \left[y^{(i)} - H(x^{(i)}) \right]^2 = (-2)^2 + 10^2 + (-9)^2 + (-5)^2 = 210$$

$$\frac{1}{n} \sum_{i=1}^{n} \left[y^{(i)} - H(x^{(i)}) \right]^2 = 210/4 = 52.5$$

선형 회귀 알고리즘

- **회귀 알고리즘의 궁극적인 목표**

$$cost(w,b) = \frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)} - H(x^{(i)})\right]^2$$



선형 회귀 알고리즘

- 선형 회귀 알고리즘을 적용하여 학습하기(모델생성)

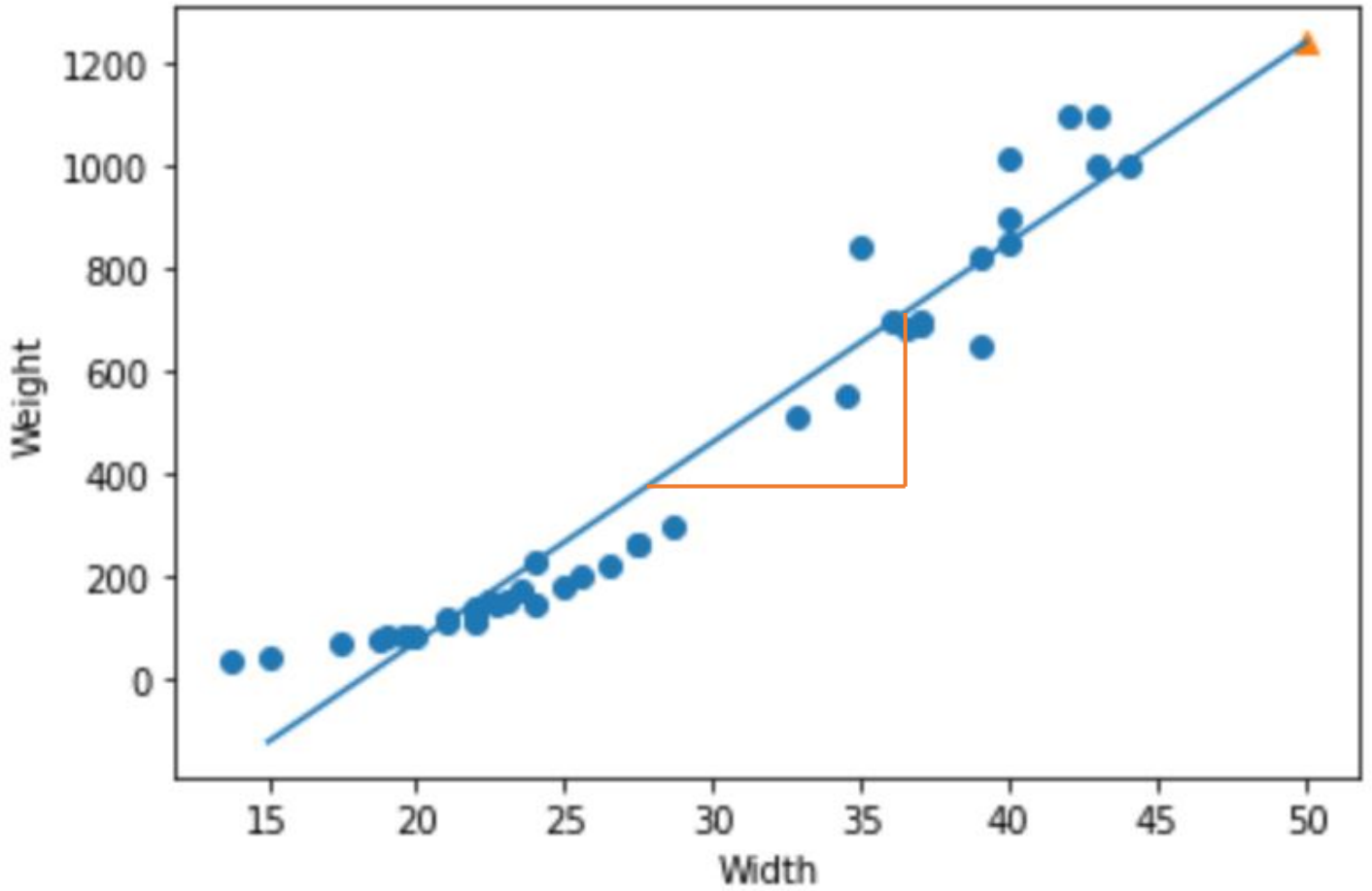
```
from sklearn.linear_model import LinearRegression


model = LinearRegression()
model.fit(train_data, train_answer)
```

- Width 50cm를 넣어 무게 예측해 보기

```
print(model.predict([[50]]))
```

선형 회귀 알고리즘

- **50cm 제품의 최근접 이웃들을 산점도에 표시하기**



Weight = a x Width + b

a : 기울기
b : 절편

선형 회귀 알고리즘

- **모델 평가 : Training Set vs. Test Set**

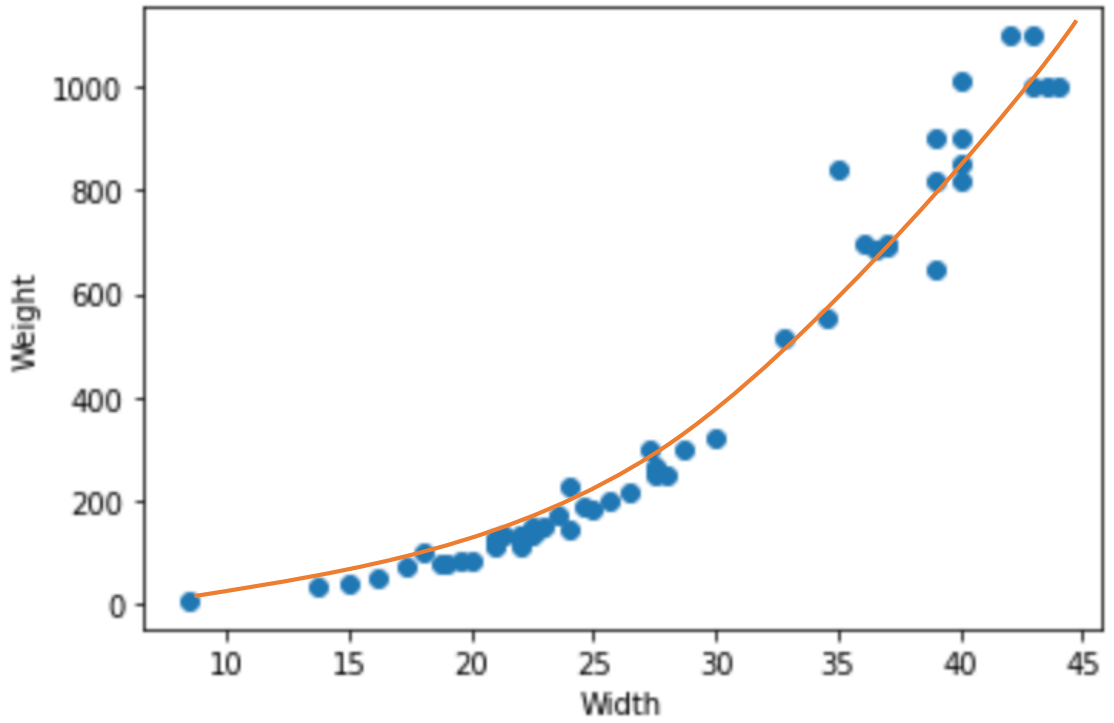
Test Set `model.score(test_data, test_answer)` `0.8247503123313558`

Training Set `model.score(train_data, train_answer)` `0.939846333997604`

선형 회귀 알고리즘 또한 해당 데이터에는 부적합

(데이터에 정확히 Fitting시킬 수 있는 모델이 필요)

다항 회귀 알고리즘



$$y = w_1 x_1 + w_2 x_2 + \ldots w_n x_n + b$$

다항 회귀 알고리즘

- 사이킷런을 이용하여 훈련 및 테스트 데이터 생성하기

```
from sklearn.model_selection import train_test_split
train_data, test_data, train_answer, test_answer =
train_test_split(product_width, product_weight, random_state=42)

train_data = train_data.reshape(-1,1)
test_data = test_data.reshape(-1,1)

train_square = np.column_stack((train_data ** 2, train_data))
test_square = np.column_stack((test_data ** 2, test_data))

print(train_data.shape, test_data.shape)
print(train_square.shape, test_square.shape)
print(train_square)
```

Width²	Width
[[384.16	19.6]
[484.	22.]
[349.69	18.7]
[302.76	17.4]
[1296.	36.]
[625.	25.]
[1600.	40.]
[1521.	39.]
[1849.	43.]
[484.	22.]
[400.	20.]
[484.	22.]
[576.	24.]
[756.25	27.5]
[1849.	43.]
[1600.	40.]
[576.	24.]
[441.	21.]
[756.25	27.5]

다항 회귀 알고리즘

- 선형 회귀 알고리즘을 적용하여 학습하기(모델생성)

```
from sklearn.linear_model import LinearRegression


model = LinearRegression()
model.fit(train_square, train_answer)
```

- Width 50cm를 넣어 무게 예측해 보기

```
print(model.predict([[50**2,50]]))
```

```
[1573.98423528]
```

- 계수와 절편 출력하기

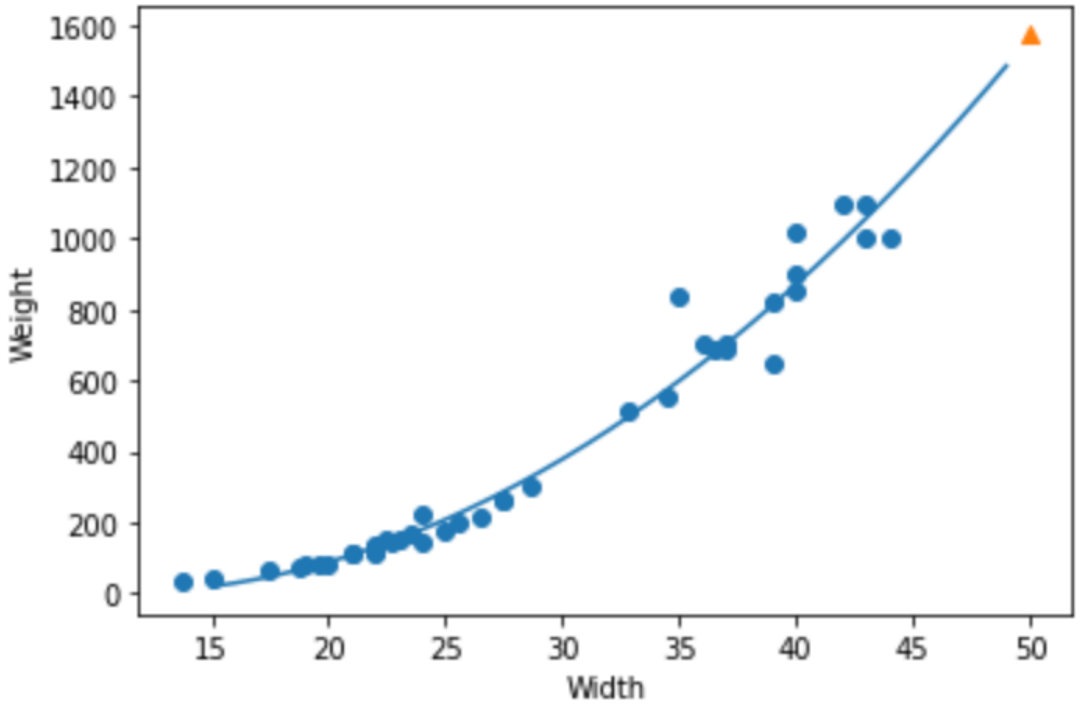
```
print(model.coef_, model.intercept_)
```

```
[ 1.01433211 -21.55792498] 116.0502107827827
```

다항 회귀 알고리즘

- **50cm 제품의 최근접 이웃들을 산점도에 표시하기**

```
import matplotlib.pyplot as plt


width = np.arange(15,50)
plt.scatter(train_data, train_answer)
plt.plot(width, 1.01*width**2 + (-
21.55)*width + 116.05021078278 27)


plt.scatter(50, 1574, marker='^')
plt.xlabel('Width')
plt.ylabel('Weight')
plt.show()
```



다항 회귀 알고리즘

- **Training Set와 Test Set의 모델 평가해 보기**

```
print('Training Set : ' + str(model.score(train_square, train_answer)))
print('Test Set     : ' + str(model.score(test_square, test_answer)))
```

```
Training Set : 0.9706807451768623
Test Set : 0.9775935108325122
```

“지금 이 순간에 최선을 다하면
다음 순간을 위한 최고의 장소에
있게 됩니다.”

-오프라 윈프리-