Ch3. 회귀 알고리즘과 모델 규제

동아대 IT 취업동아리 – 머신러닝 스터디

경영정보학과 김예나

시작하기 전에

1장

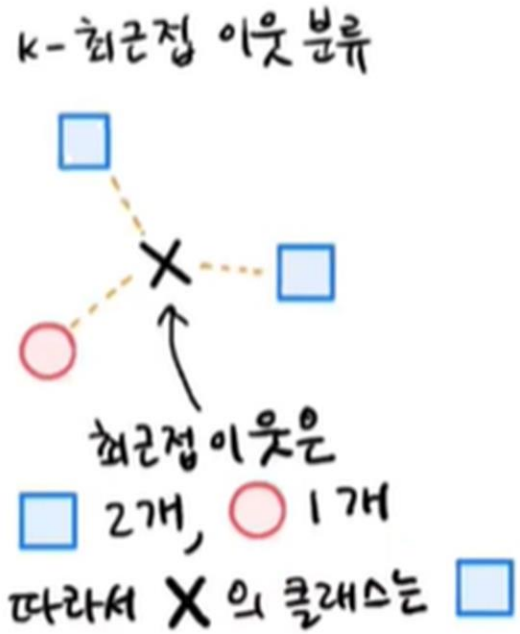
- 30cm 이상은 도미
- 생선 이름을 자동으로 알려주는 머신러닝

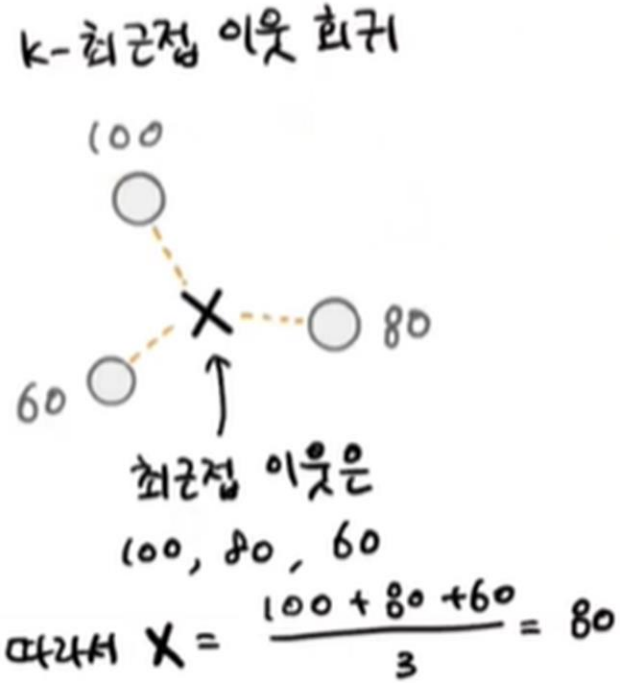
2장

- 답을 모두 외우고 있다면 같은 데이터로 모델을 평가하는 것은 이상하다
- 훈련 세트와 테스트 세트로 나누자 + 전처리

농어 무게 “예측”

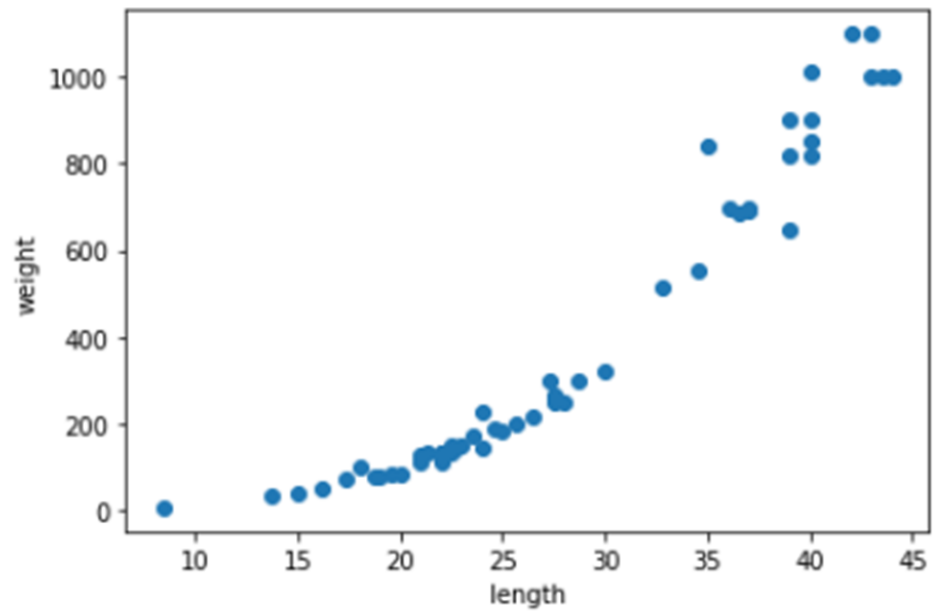
K-최근접 이웃 회귀


k-최근접 이웃 분류
X
최근접 이웃은
2개, 1개
따라서 X의 클래스는


k-최근접 이웃 회귀
100
80
60
X
최근접 이웃은
100, 80, 60
따라서 X = (100 + 80 + 60) / 3 = 80

데이터 준비

```
import matplotlib.pyplot as plt
plt.scatter(perch_length, perch_weight)
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
```



scatter() : 산점도

데이터 준비

```
from sklearn.model_selection import train_test_split
train_input, test_input, train_traget, test_target
    = train_test_split(perch_length, perch_weight, random_state=42)
```

```
train_input = train_input.reshape(-1, 1)
test_input = test_input.reshape(-1, 1)
```

reshape() : 배열 크기 변환

train_input

```
[[19.6]
 [22. ]
 [18.7]
 [17.4]
 [36. ]
 [25. ]
 [ ... ]
 [22.5]
 [19. ]
 [37. ]
 [22. ]
 [25.6]
 [42. ]
 [34.5]]
```

(42,1)

test_input

```
[[ 8.4]
 [18. ]
 [27.5]
 [21.3]
 [22.5]
 [40. ]
 [30. ]
 [24.6]
 [39. ]
 [21. ]
 [43.5]
 [16.2]
 [28. ]
 [27.3]]
```

(14,1)

03-1. K-최근접 이웃 회귀

```
from sklearn.neighbors import KNeighborsRegressor
knr = KNeighborsRegressor()
knr.fit(train_input, train_target)
```

```
knr.score(test_input, test_target)
```

```
0.992809406101064
```

=> 아주 좋은 점수 !!

결정계수(R^2)

- 0 ~ 1
- 예측이 타깃을 어느정도 맞추는지

$$R^2 = 1 - \frac{(\text{타깃} - \text{예측})^2\text{의 합}}{(\text{타깃} - \text{평균})^2\text{의 곱}}$$

평균 절대값 오차 (mean_absolute_error)

- 타깃과 예측의 절대값 오차를 평균하여 반환
- 결정계수보다 직관적인 값

```
from sklearn.metrics import mean_absolute_error

test_prediction = knr.predict(test_input)

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

```
19.157142857142862
```

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

03-1. K-최근접 이웃 회귀

훈련 세트 점수

```
print(knr.score(train_input, train_target))
```

```
0.9698823289099254
```

테스트 세트 점수

```
print(knr.score(test_input,test_target))
```

```
0.992809406101064
```

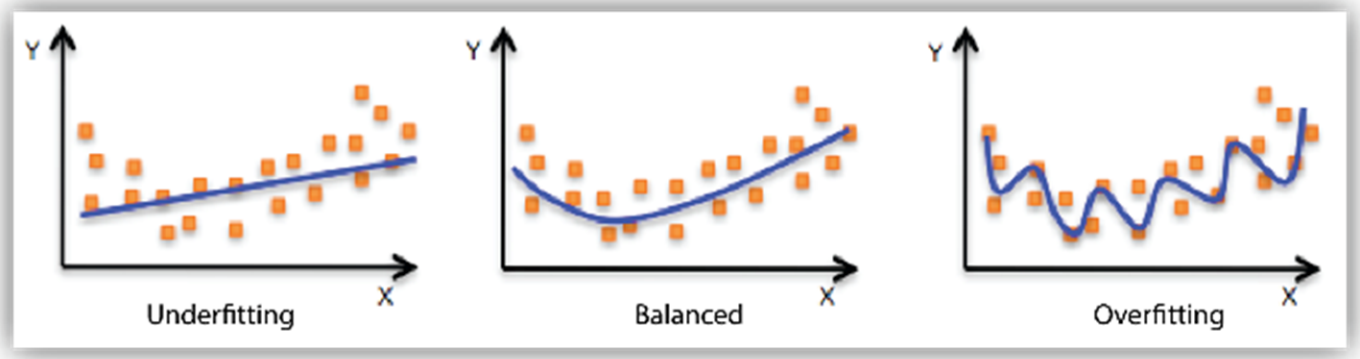
03-1. K-최근접 이웃 회귀

과대적합 (overfitting)

- 훈련세트에서는 good
- 테스트 세트에서는 bad

과소적합 (underfitting)

- 훈련 < 테스트
- 둘 다 너무 낮은 경우



https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

과소적합 문제 해결

```
knr.n_neighbors = 3

knr.fit(train_input, train_target)
```

이웃 개수 줄이기

훈련 세트 점수

```
knr.score(train_input, train_target)
```

```
0.9804899950518966
```

테스트 세트 점수

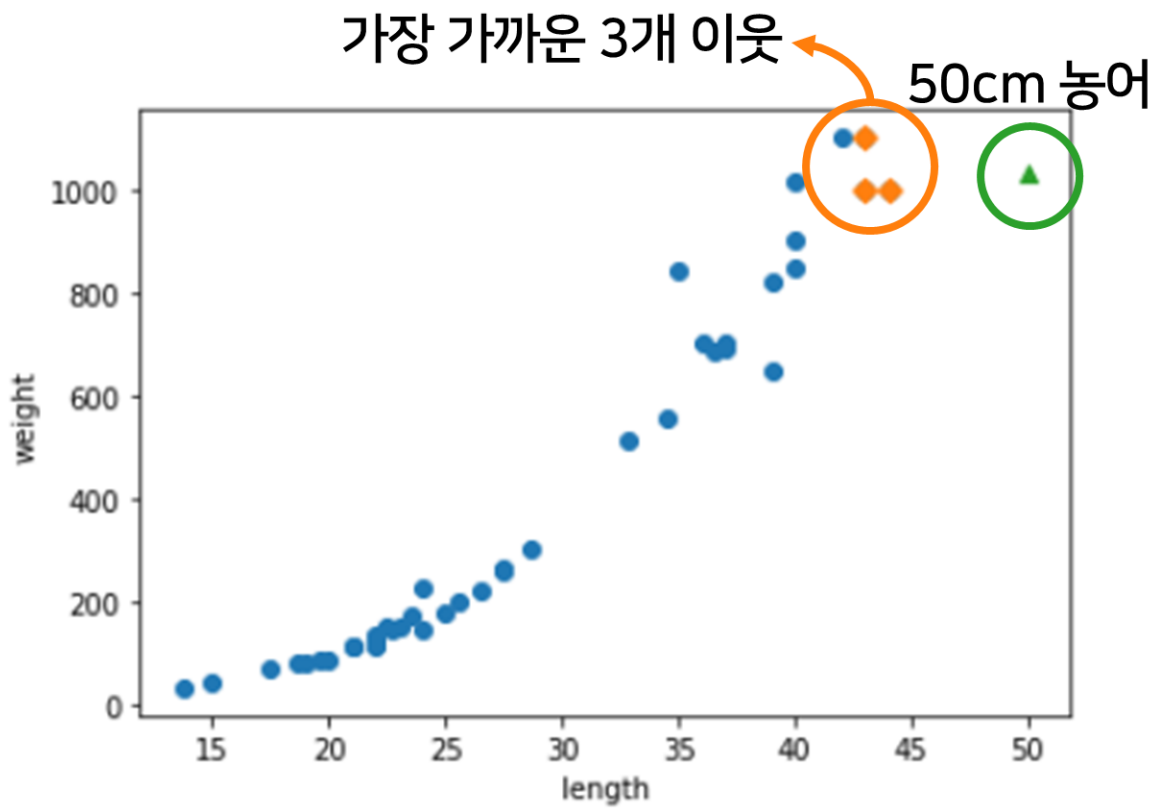
```
knr.score(test_input, test_target)
```

```
0.9746459963987609
```

50cm 농어 무게 예측

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

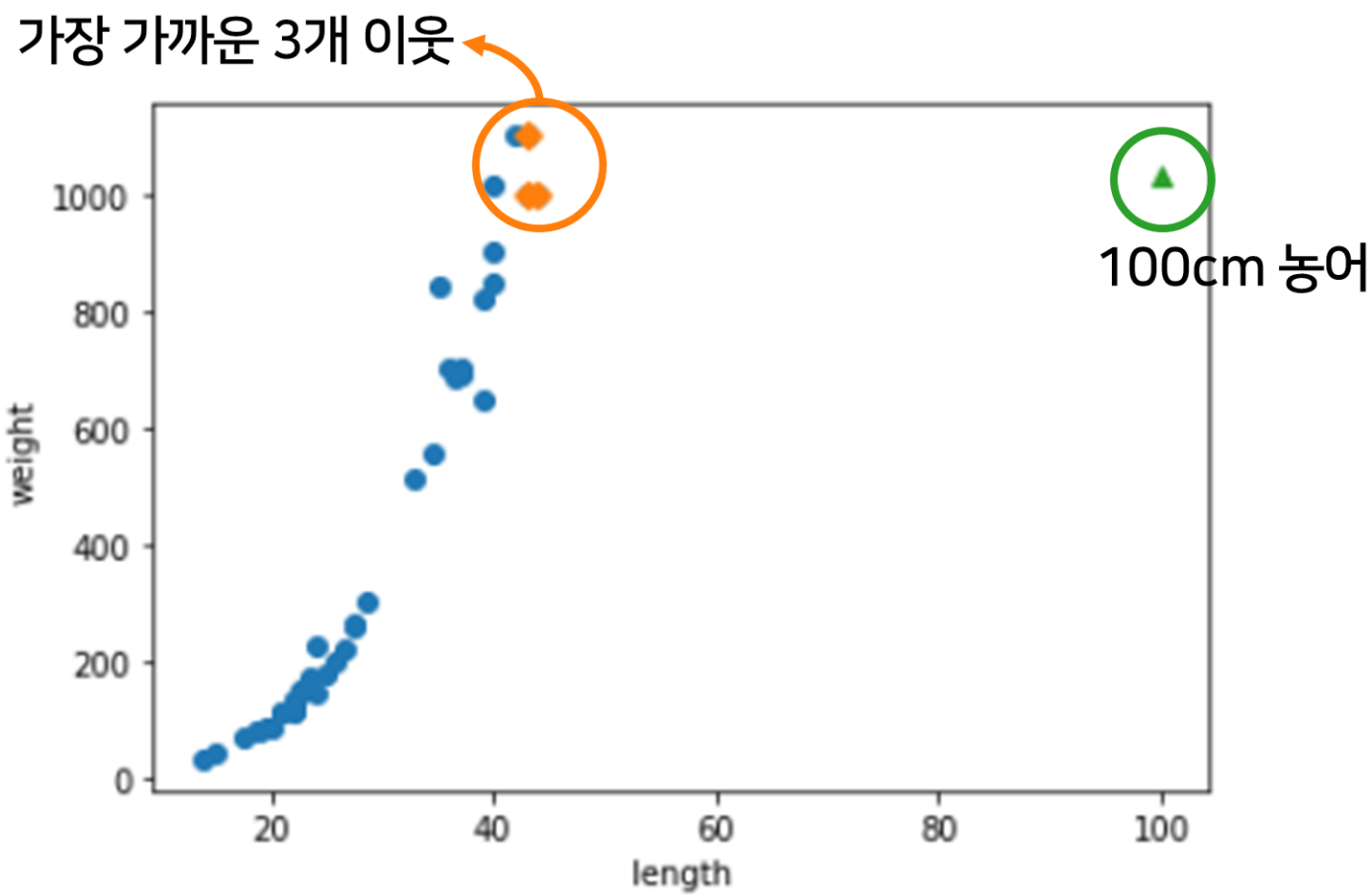
```
[1033.33333333]
```



03-2. 선형 회귀

```
print(knr.predict([[100]]))
```

```
[1033.33333333]
```



아무리 큰 농어라도 무게가 더 늘어나지 않는다 !

선형회귀(Linear Regression)

- 특성과 타깃 사이의 관계를 가장 잘 나타내는 직선
- 특성 1개 : 직선의 방정식
- y=ax+b

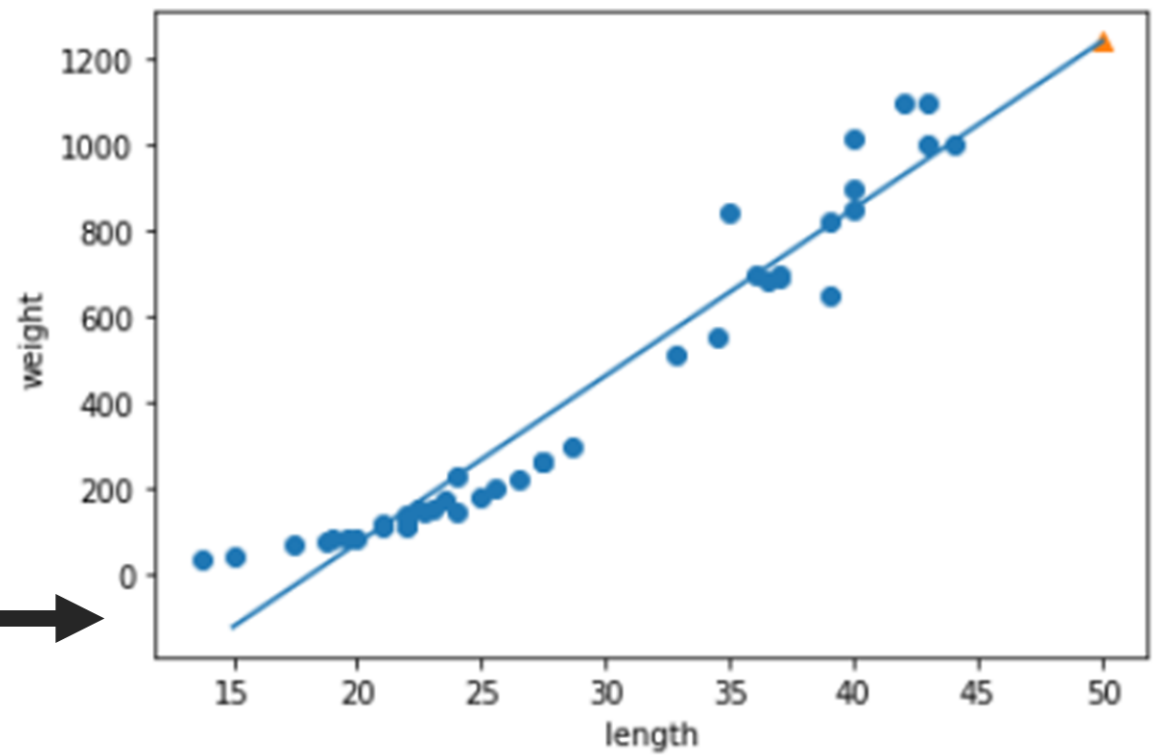
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()

lr.fit(train_input, train_target)
```

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

```
[1241.83860323]
```

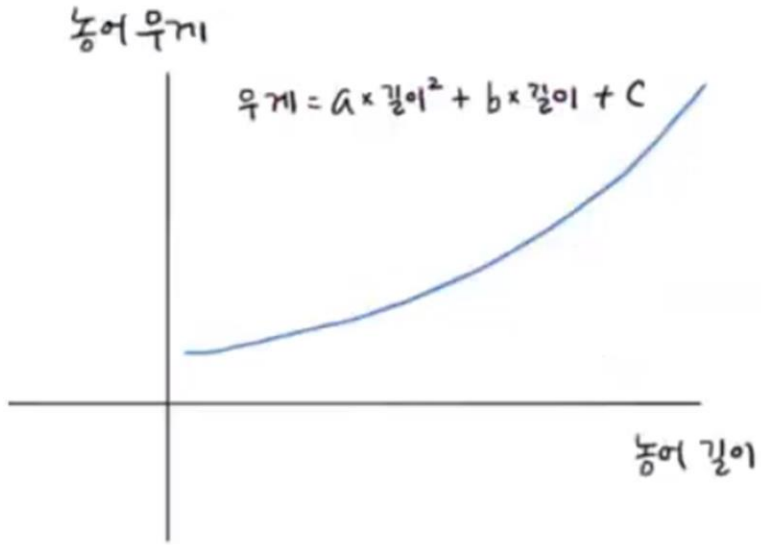
03-2. 선형 회귀

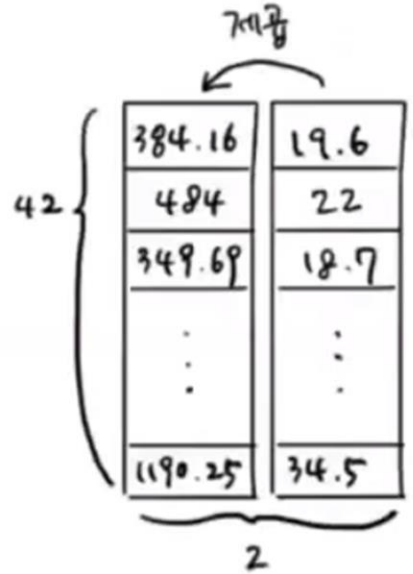
```
plt.scatter(train_input,train_target)

plt.plot([15,50], [15*lr.coef_+lr.intercept_, 50*lr.coef_+lr.intercept_])

plt.scatter(50,1241.8, marker='^')
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
```



무게가 음수 !!

다항 회귀(Polynomial Regression)






```
train_poly = np.column_stack((train_input ** 2, train_input))
test_poly = np.column_stack((test_input ** 2, test_input))
```

다항 회귀(Polynomial Regression)

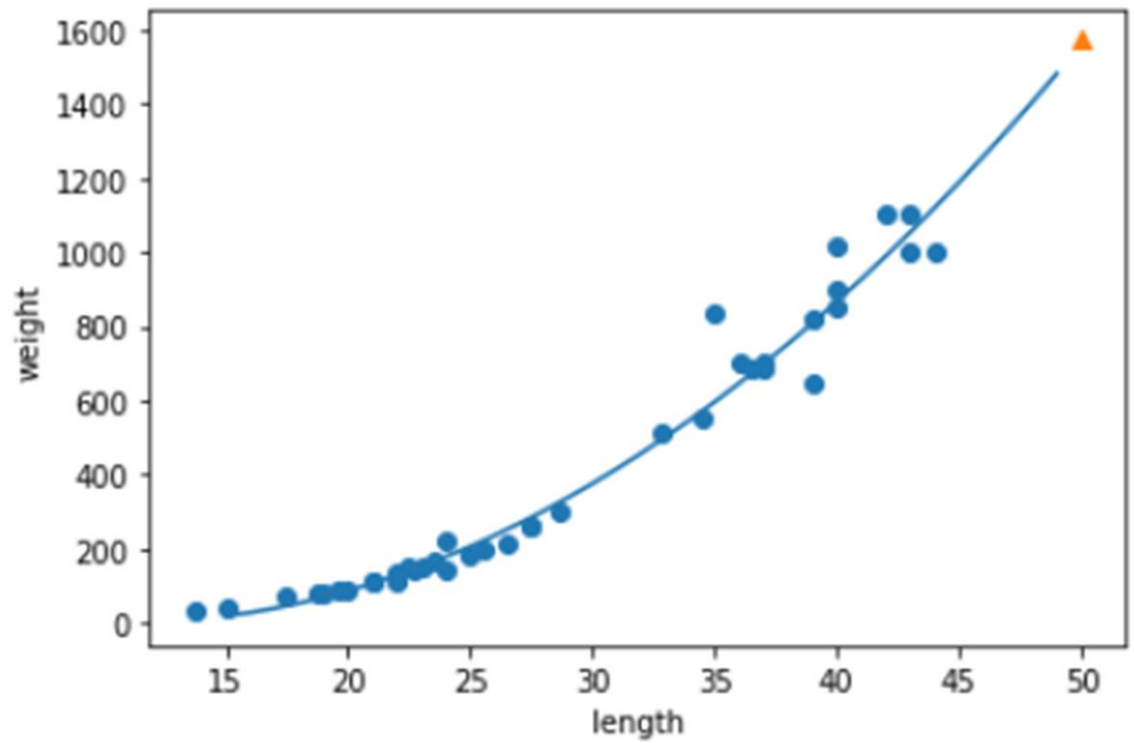
```
lr = LinearRegression()
lr.fit(train_poly,train_target)
```

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

```
[1573.98423528]
```

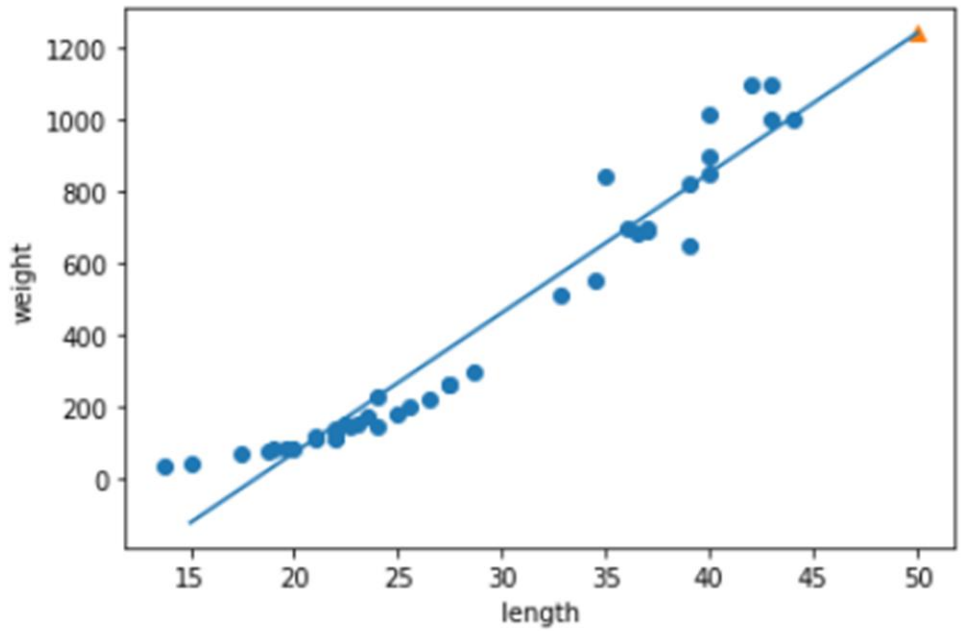
```
print(lr.coef_,lr.intercept_)
```

```
[  1.01433211 -21.55792498] 116.05021078278277
```



03-2. 선형 회귀

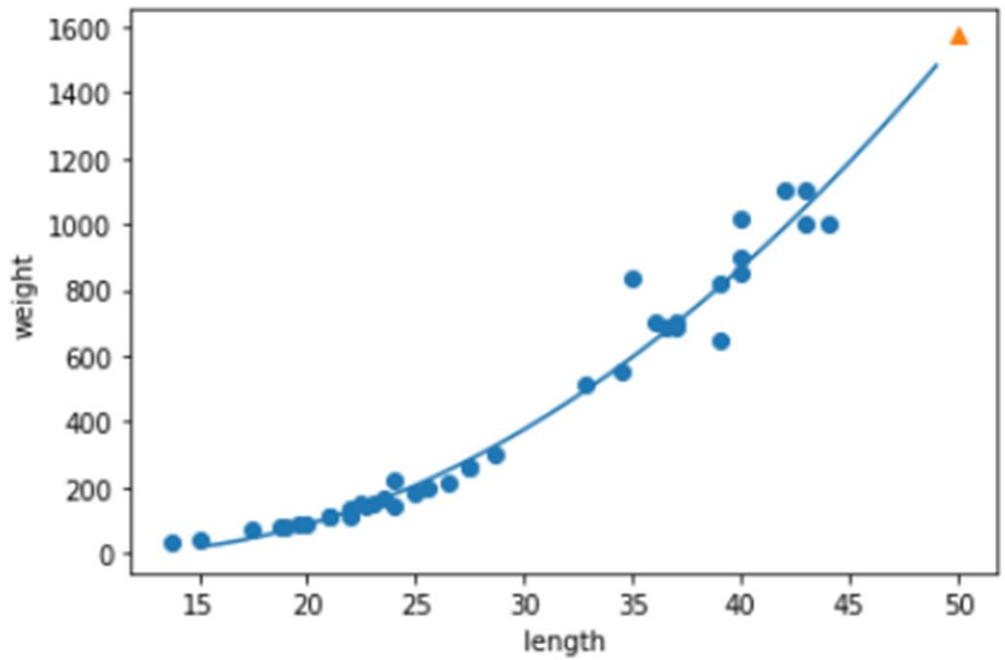
선형 회귀




```
print(lr.score(train_input, train_target))
print(lr.score(test_input, test_target))
```

```
0.939846333997604
0.8247503123313558
```

다항 회귀




```
print(lr.score(train_poly, train_target))
print(lr.score(test_poly, test_target))
```

```
0.9706807451768623
0.9775935108325122
```

감사합니다