선형 회귀


구글 코랩에서 실행하기

k-최근접 이웃의 한계

In [1]:
```
import numpy as np

perch_length = 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]
     )
perch_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]
     )
```

In [3]:
```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(
    perch_length, perch_weight, random_state=42)
print(train_input.shape, train_target.shape)
print(test_input.shape, test_target.shape)

train_input = train_input.reshape(-1, 1) # 2차원으로 변경
test_input = test_input.reshape(-1, 1)   # 2차원으로 변경
```

```
(42,) (42,)
(14,) (14,)
```

In [4]:
```
train_input.shape
```

Out[4]:
```
(42, 1)
```

In [5]:
```
test_input.shape
```

Out[5]:
```
(14, 1)
```

In [6]:
```
from sklearn.neighbors import KNeighborsRegressor

knr = KNeighborsRegressor(n_neighbors=3)
knr.fit(train_input, train_target)
```

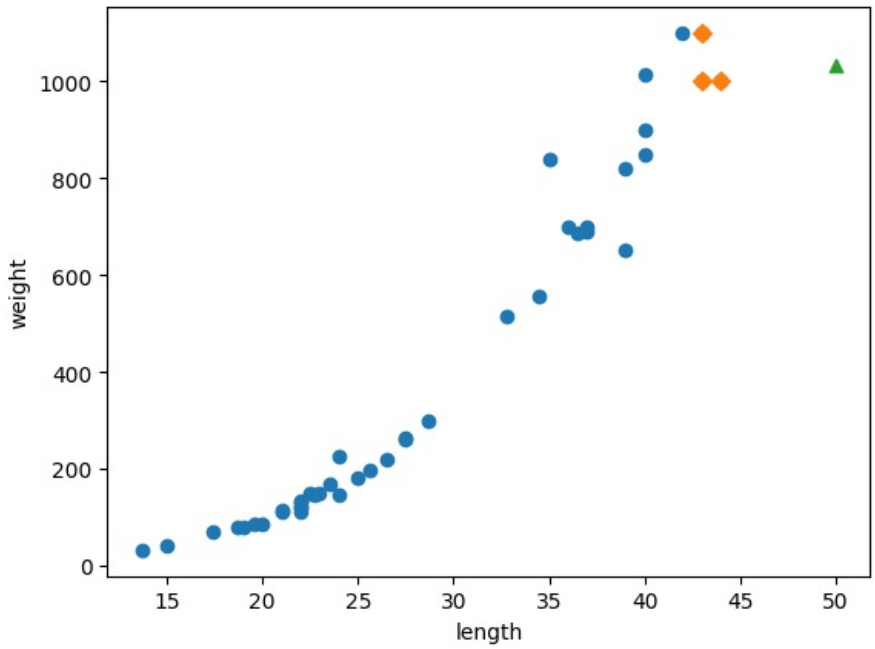
Out[6]:
```
KNeighborsRegressor(n_neighbors=3)
```

In [ ]:
```
print(knr.predict([[50]])) # 제대로 예측이 안됨
```

```
[1033.33333333]
```

In [7]:
```
import matplotlib.pyplot as plt
```

In [8]:
```
distances, indexes = knr.kneighbors([[50]]) # 50 cm 농어의 이웃을 구함

plt.scatter(train_input, train_target)
plt.scatter(train_input[indexes], train_target[indexes], marker='D')
plt.scatter(50, 1033, marker='^')
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
```



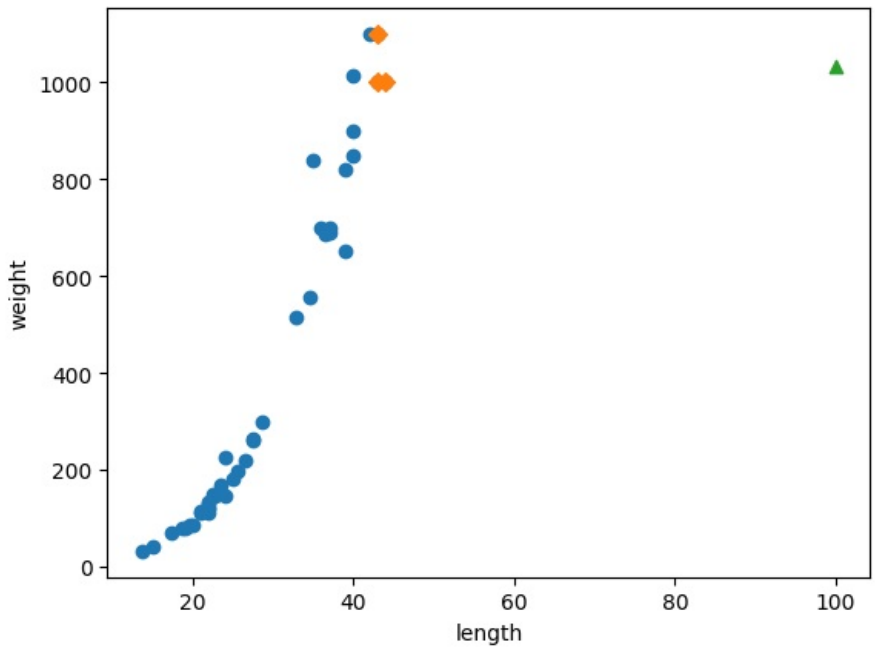
In [ ]:
```
print(np.mean(train_target[indexes]))
```

```
1033.3333333333333
```

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

```
[1033.33333333]
```

In [10]:
```
distances, indexes = knr.kneighbors([[100]]) # 100cm 농어의 이웃을 구함


plt.scatter(train_input, train_target)
plt.scatter(train_input[indexes], train_target[indexes], marker='D')
plt.scatter(100, 1033, marker='^')
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
# 농어가 아무리 커도 무게가 더 늘어나지 않는 문제 발생 -> outlier 처리를 잘 못함
```



선형 회귀

In [11]:
```
from sklearn.linear_model import LinearRegression
```

In [12]:
```
lr = LinearRegression()
lr.fit(train_input, train_target)
```

Out[12]:
```
▾ LinearRegression
LinearRegression()
```

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

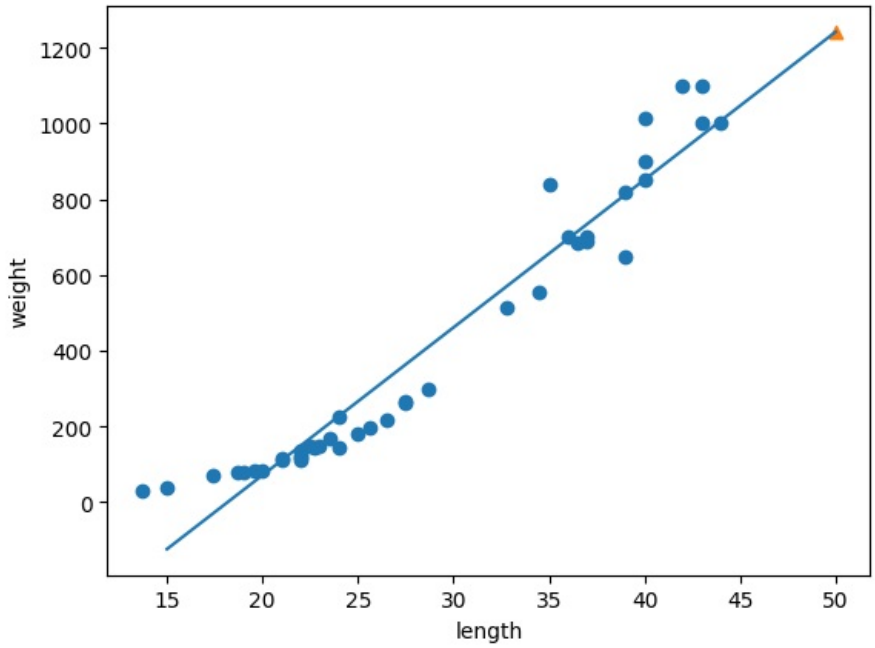
In [13]:
```
print(lr.predict([[50]]))
```

```
[1241.83860323]
```

In [14]:
```
print(lr.coef_, lr.intercept_) # 기울기와 절편
```

```
[39.01714496] -709.0186449535477
```

In [15]:
```
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()
```



In [ ]:
```
# R^2 점수 확인
print(lr.score(train_input, train_target))
print(lr.score(test_input, test_target))
```

```
0.939846333997604
0.8247503123313558
```

다항 회귀

In [16]:
```
train_poly = np.column_stack((train_input ** 2, train_input))
test_poly = np.column_stack((test_input ** 2, test_input))
```

In [17]:
```
print(train_input.shape)
x = train_input**2
print(x.shape)
```

```
(42, 1)
(42, 1)
```

In [18]:
```
print(train_poly.shape, test_poly.shape)
```

```
(42, 2) (14, 2)
```

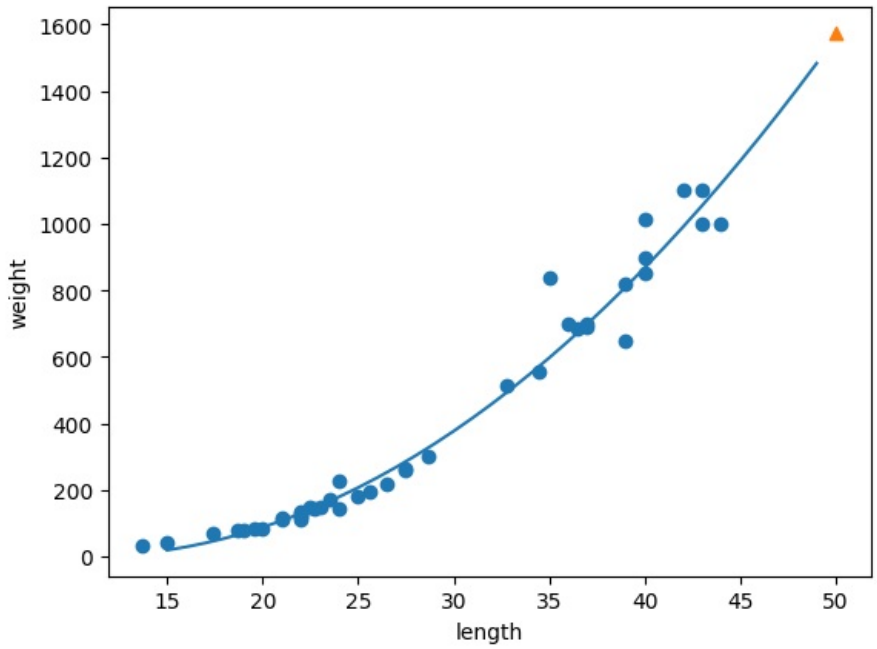
In [19]:
```
lr = LinearRegression()
lr.fit(train_poly, train_target)

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

```
[1573.98423528]
```

In [20]:
```
print(lr.coef_, lr.intercept_)
```

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

In [21]:
```
point = np.arange(15, 50)
plt.scatter(train_input, train_target)
plt.plot(point, 1.01*point**2 - 21.6*point + 116.05)
plt.scatter([50], [1574], marker='^')
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
```



In [22]:
```
print(lr.score(train_poly, train_target))
print(lr.score(test_poly, test_target))
# Polynomial Regression을 진행하니 성능 향상
```

```
0.9706807451768623
0.9775935108325122
```

In [ ]: