선형 회귀

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, 23.9, 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 [2]:
```
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 [ ]:
```
train_input.shape
```

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

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

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

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

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

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

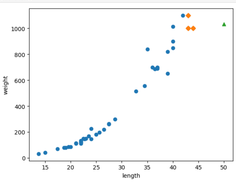
In [ ]:
```
print(knr.predict([[50]])) # 제대로 예측이 안됨
```

```
[1033.33333333]
```

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

In [ ]:
```
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]))
```

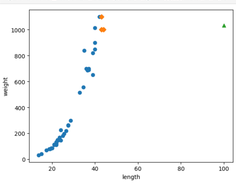
```
1033.3333333333333
```

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

```
[1033.33333333]
```

In [ ]:
```
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 [13]:
```
from sklearn.linear_model import LinearRegression
```

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

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

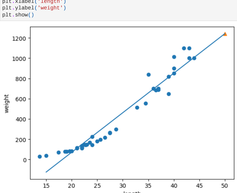
In [ ]:
```
print(lr.predict([[50]]))
```

```
[1241.83860323]
```

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

```
[39.01714496] -709.0186449535477
```

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

```
0.9398463339976604
0.8247503123313558
```

다항 회귀

In [18]:
```
# 다항식을 사용한 linear regression -> 곡선
# 그래서 아이어를 위공한 것과 train_input 두 배열을 나란히 붙힘
train_poly = np.column_stack((train_input ** 2, train_input))
test_poly = np.column_stack((test_input ** 2, test_input))
print(train_poly.shape)
```

```
(42, 2)
```

In [9]:
```
print(train_input.shape)
x = train_input**2
print(x.shape)
print(x)
```

```
(42, 1)
(42, 1)
[[ 384.16]
 [ 484.  ]
 [ 349.69]
 [ 302.76]
 [1296.  ]
 [ 625.  ]
 [1600.  ]
 [1521.  ]
 [1849.  ]
 [ 484.  ]
 [ 400.  ]
 [ 484.  ]
 [ 576.  ]
 [ 756.25]
 [1849.  ]
 [1444.  ]
 [ 576.  ]
 [ 441.  ]
 [ 756.25]
 [1600.  ]
 [1075.84]
 [ 702.25]
 [1332.25]
 [ 187.69]
 [ 515.29]
 [ 225.  ]
 [1369.  ]
 [1225.  ]
 [ 823.69]
 [ 552.25]
 [1521.  ]
 [ 441.  ]
 [ 529.  ]
 [ 484.  ]
 [1936.  ]
 [ 506.25]
 [ 361.  ]
 [1369.  ]
 [ 484.  ]
 [ 655.36]
 [1764.  ]
 [1190.25]]
```

In [10]:
```
print(train_poly.shape, test_poly.shape)
```

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

In [11]:
```
print(train_poly)
```

```
[[ 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 ]
 [1600.    40.  ]
 [1075.84  32.8 ]
 [ 702.25  26.5 ]
 [1332.25  36.5 ]
 [ 187.69  13.7 ]
 [ 515.29  22.7 ]
 [ 225.    15.  ]
 [1369.    37.  ]
 [1225.    35.  ]
 [ 823.69  28.7 ]
 [ 552.25  23.5 ]
 [1521.    39.  ]
 [ 441.    21.  ]
 [ 529.    23.  ]
 [ 484.    22.  ]
 [1936.    44.  ]
 [ 506.25  22.5 ]
 [ 361.    19.  ]
 [1369.    37.  ]
 [ 484.    22.  ]
 [ 655.36  25.6 ]
 [1764.    42.  ]
 [1190.25  34.5 ]]
```

In [14]:
```
lr = LinearRegression()
lr.fit(train_poly, train_target)

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

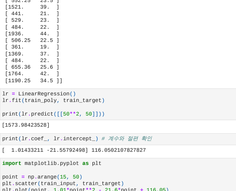
```
[1573.98423528]
```

In [15]:
```
print(lr.coef_, lr.intercept_) # 계수와 절편 확인
```

```
[  1.01433211 -21.55792498] 116.05021078278927
```

In [17]:
```
import matplotlib.pyplot as plt

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 [ ]:
```
# r^2 점수 평가
print(lr.score(train_poly, train_target))
print(lr.score(test_poly, test_target))
# Polynomial Regression을 진행하니 상능 향상
```

```
0.9706807451768623
0.9775935108325122
```

In [ ]: