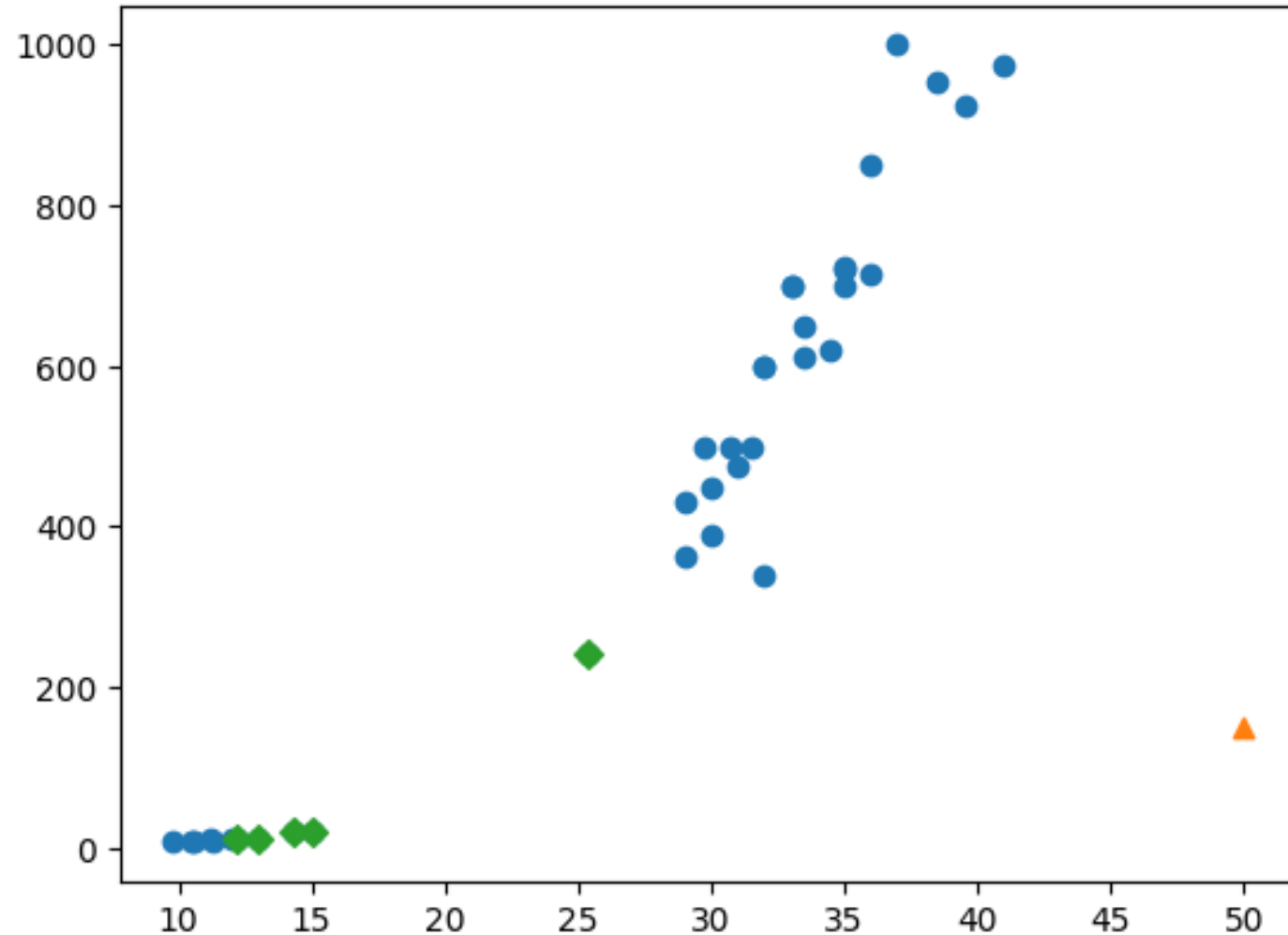


수상한 도미

길이가 50cm, 무게가 150g인 물고기는?



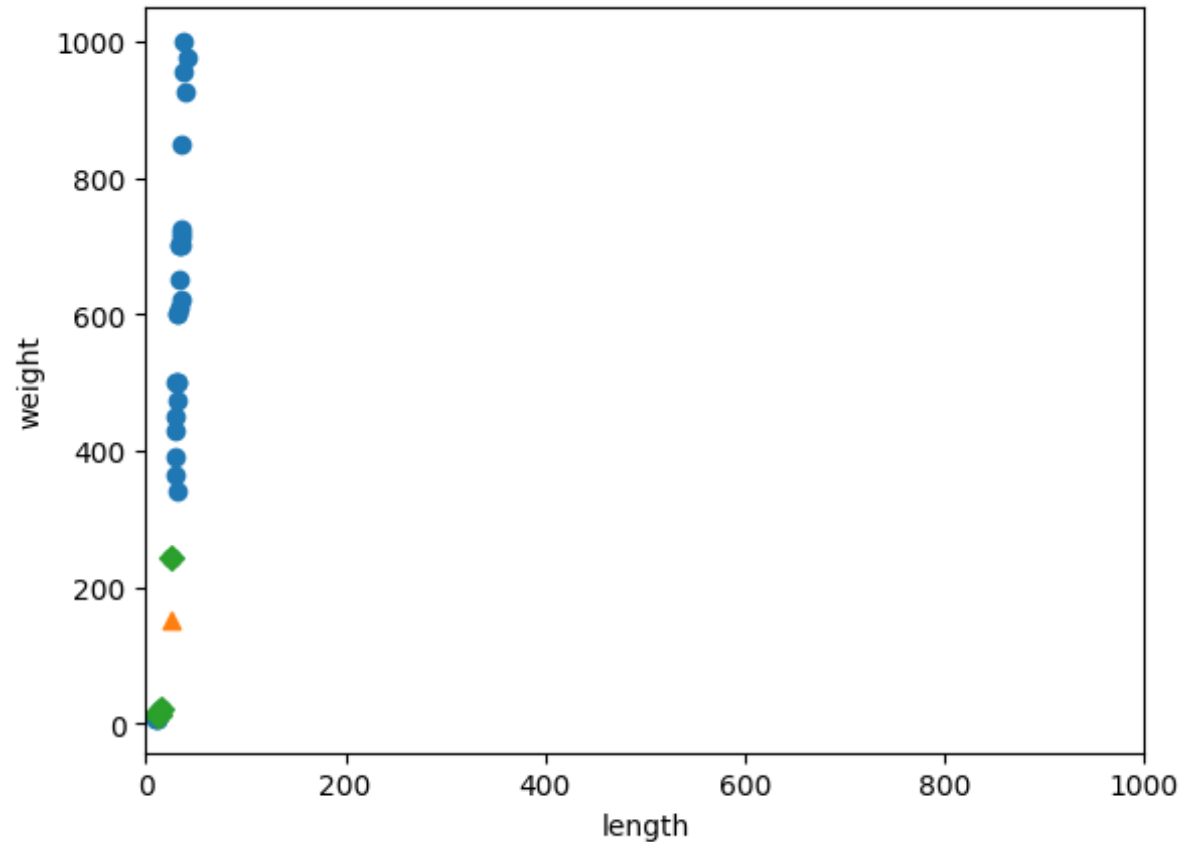
왜 이러한 문제가 난 것인가?

Knn의 작동하는 알고리즘을 생각해보자.

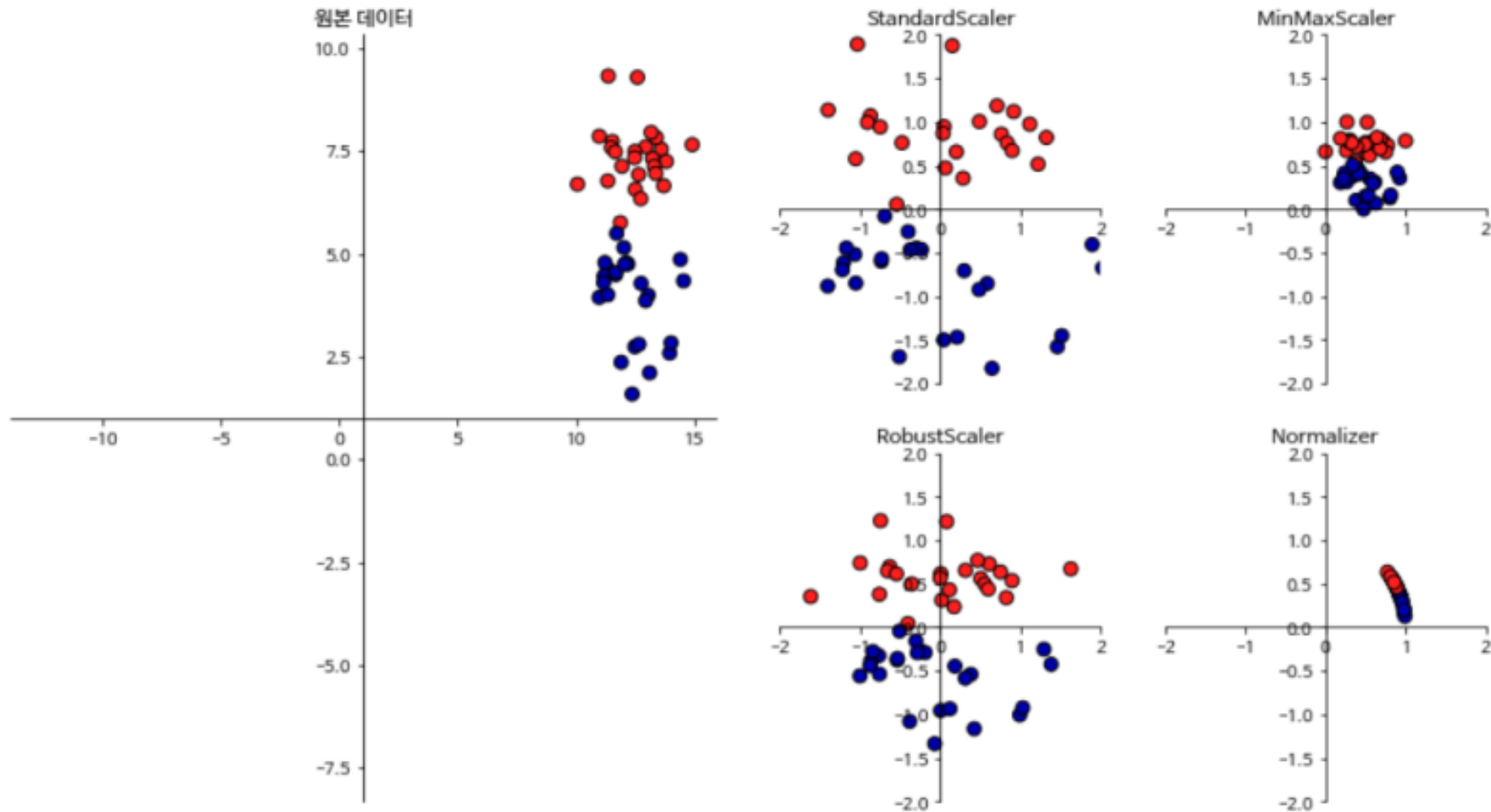
각 변수에 대한
가중치를 고려해야함.

비율을 동일하게 본다면?

```
plt.scatter(train_input[:,0], train_input[:,1])
plt.scatter(25, 150, marker='^')
plt.scatter(train_input[indexes,0], train_input[indexes,1], marker='D')
plt.xlim((0, 1000))
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
```



각 변수의 대한 데이터 범위를 동일하게

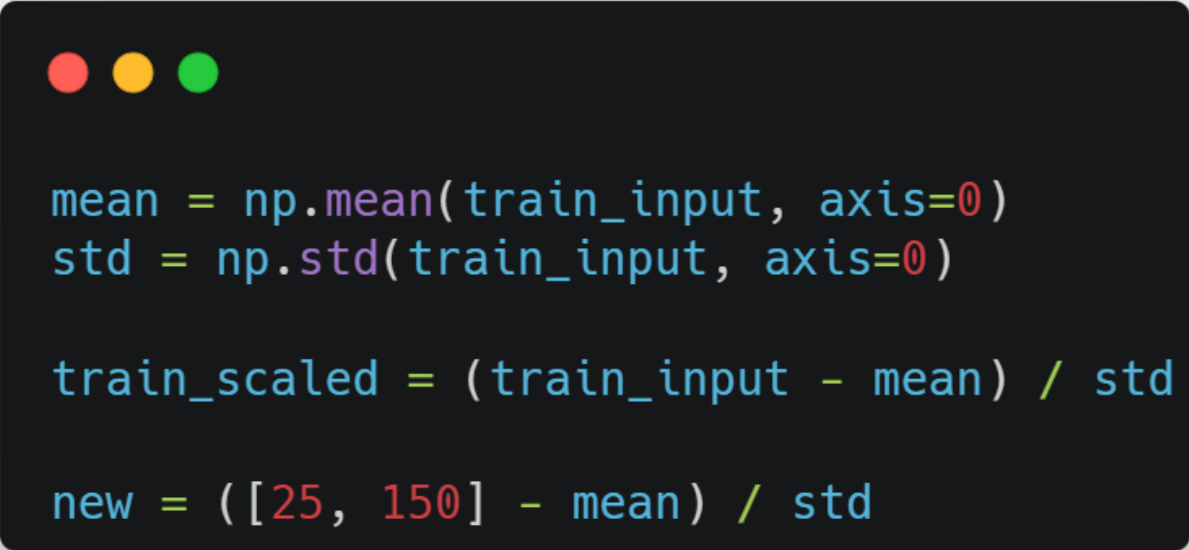


StandardScaler 수식

$$X'_i = \frac{X_i - \mu}{\sigma} = \frac{X_i - X_{\text{mean}}}{X_{\text{std}}}$$

StandardScaler

StandardScaler 수행하기



```
mean = np.mean(train_input, axis=0)
std = np.std(train_input, axis=0)

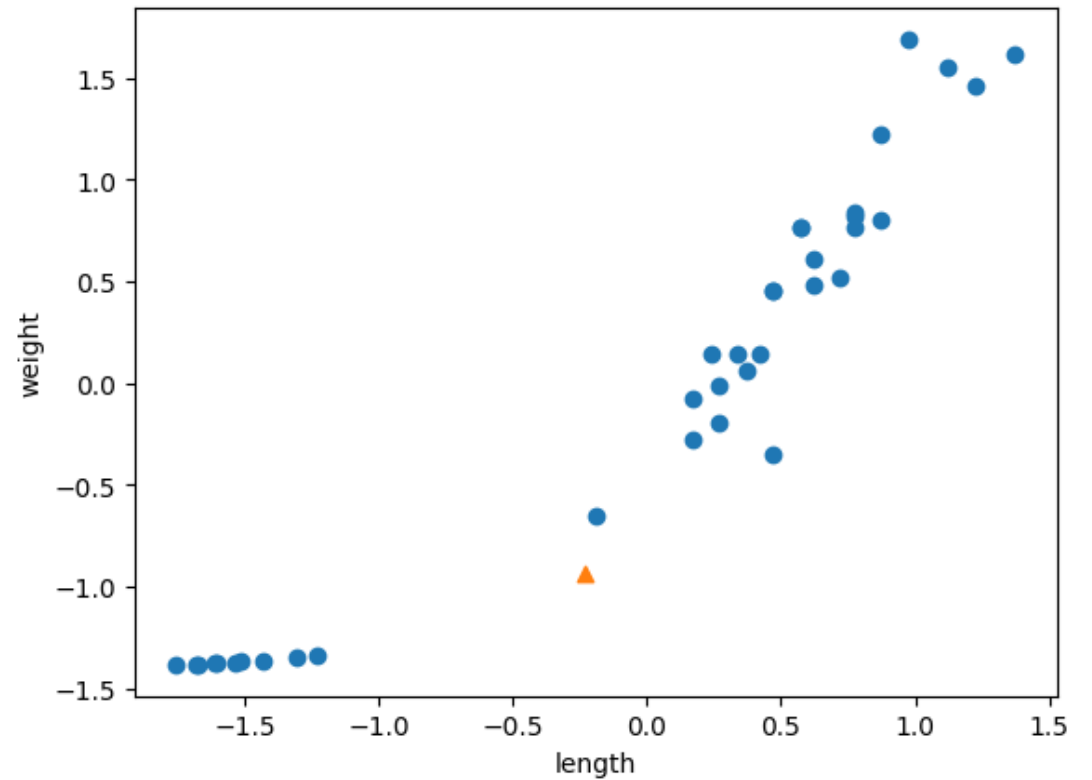
train_scaled = (train_input - mean) / std

new = ([25, 150] - mean) / std
```


전처리 후 데이터 확인



```
plt.scatter(train_scaled[:,0], train_scaled[:,1])  
plt.scatter(new[0], new[1], marker='^')  
plt.xlabel('length')  
plt.ylabel('weight')  
plt.show()
```



테스트 데이터도 동일한 전처리 수행.

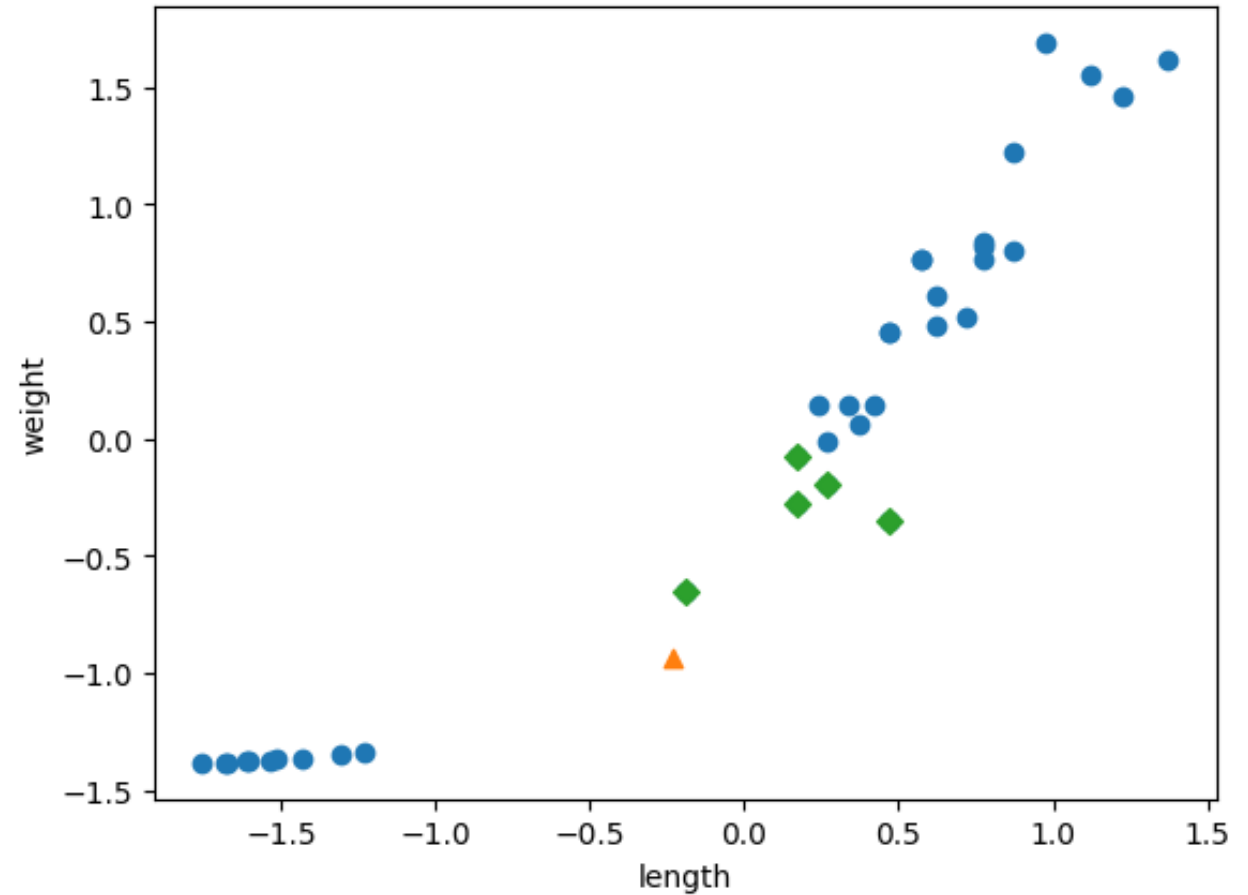


```
plt.scatter(train_scaled[:,0], train_scaled[:,1])  
plt.scatter(new[0], new[1], marker='^')  
plt.xlabel('length')  
plt.ylabel('weight')  
plt.show()
```

데이터 전처리 한 후, KNN 결과 확인



```
plt.scatter(train_scaled[:,0], train_scaled[:,1])  
plt.scatter(new[0], new[1], marker='^')  
plt.scatter(train_scaled[indexes,0], train_scaled[indexes,1], marker='D')  
plt.xlabel('length')  
plt.ylabel('weight')  
plt.show()
```



데이터 준비



```
import pandas as pd  
  
fish = pd.read_csv('https://bit.ly/fish_csv_data')  
fish.head()
```

	Species	Weight	Length	Diagonal	Height	Width
0	Bream	242.0	25.4	30.0	11.5200	4.0200
1	Bream	290.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	26.5	31.1	12.3778	4.6961
3	Bream	363.0	29.0	33.5	12.7300	4.4555
4	Bream	430.0	29.0	34.0	12.4440	5.1340

Input, output 데이터 준비



```
fish_input = fish[['Weight', 'Length', 'Diagonal', 'Height', 'Width']].to_numpy()  
fish_target = fish['Species'].to_numpy()
```

데이터 전처리

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(
    fish_input, fish_target, random_state=42)

from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
ss.fit(train_input)
train_scaled = ss.transform(train_input)
test_scaled = ss.transform(test_input)
```

Knn 수행




```
from sklearn.neighbors import KNeighborsClassifier

kn = KNeighborsClassifier(n_neighbors=3)
kn.fit(train_scaled, train_target)

print(kn.score(train_scaled, train_target))
print(kn.score(test_scaled, test_target))
```

로지스틱 회귀 수행



```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(train_bream_smelt, target_bream_smelt)
```


로지스틱의 파라미터와 결과값

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

decisions = lr.decision_function(train_bream_smelt[:5])
print(decisions)

from scipy.special import expit
print(expit(decisions))
```