

머신러닝 데이터 다루기: 훈련 세트와 테스트 세트

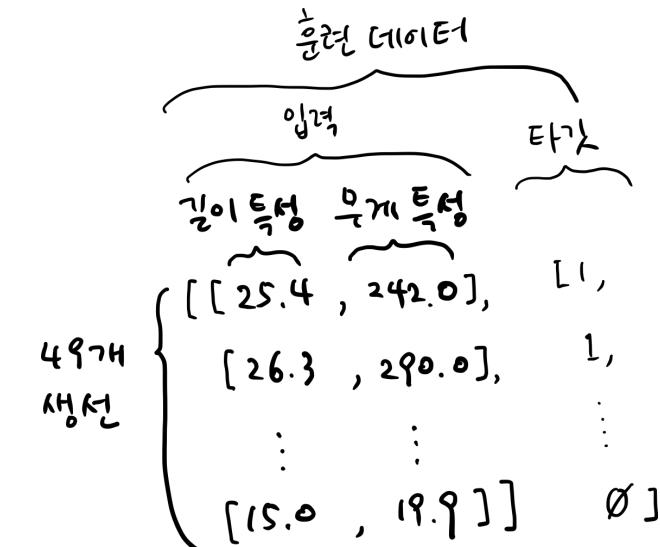
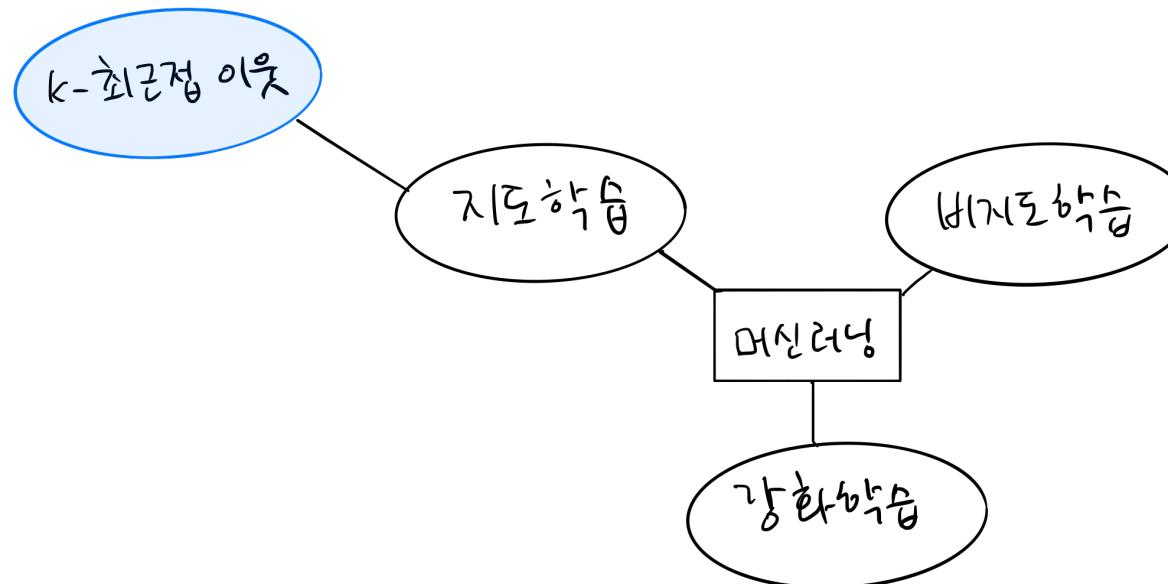
임 경 태

CONTENTS

- 1 지난 강의 복습**
- 2 학습/검증 데이터**
- 3 표준화**

❶ K-최근접 알고리즘을 이용한 도미/빙어 분류

- 기계학습 방법 중 **지도 학습**은 **입력**(데이터)과 **타깃**(정답)으로 이루어진 훈련 데이터를 이용.
- 도미, 빙어 구분 문제의 경우 입력으로 사용된 길이와 무게를 **특성(feature)**이라고 한다.
- 기계학습 **모델**은 K-최근접 이웃 모델을 활용함.



CONTENTS

1

복습

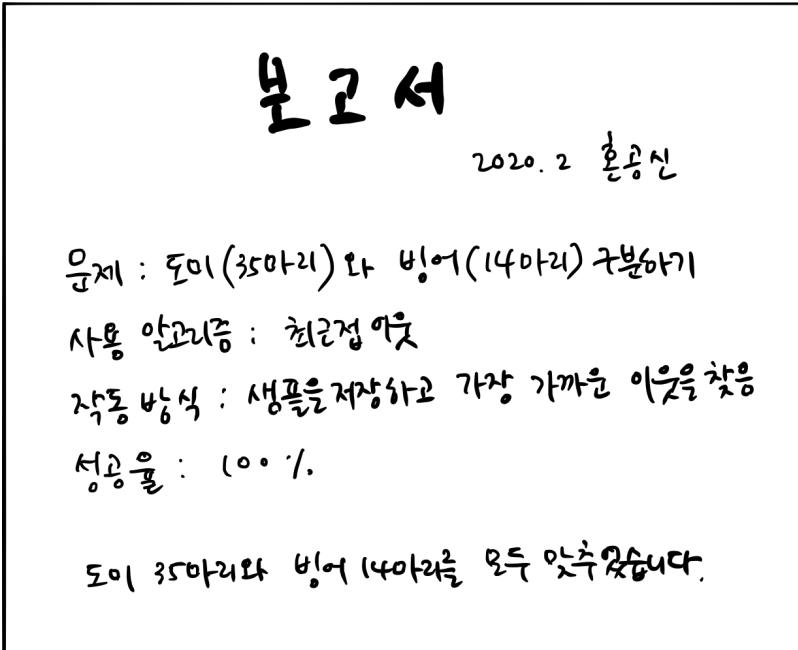
2

학습/검증 데이터

3

표준화

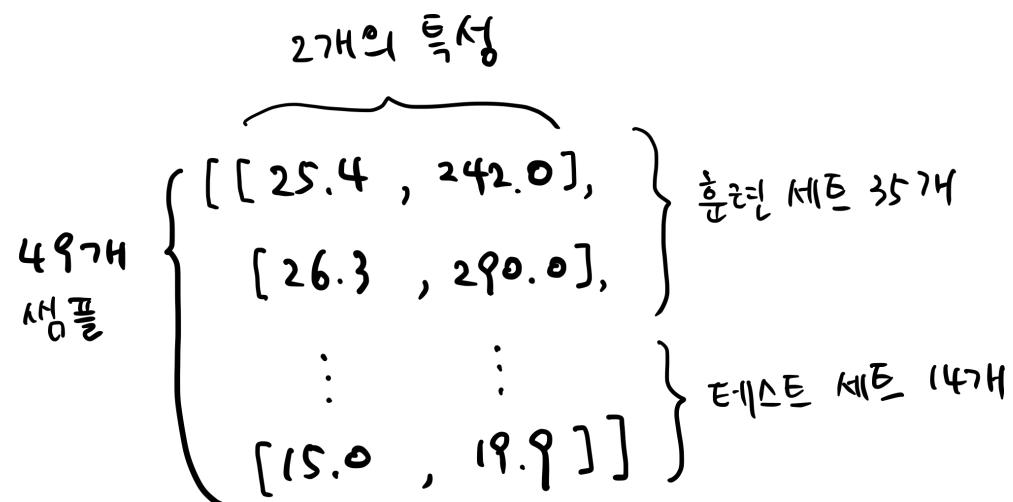
의문점



- **의문점:** 학습에 사용된 샘플 데이터는 이미 기계가 알고 있는 데이터.. 모두 맞추는게 당연한 것 아닌가?
 - 중간고사 시험문제와 답을 알려주고 시험 보는 격
- **가설:** 기계가 학습을 마친 후 처음 보는 데이터로 평가를 해야해야 공정할 듯
 - 연습문제와 시험문제는 달라야 한다!

📝 Train / Test 데이터는 어떻게 나누면 될까?

- 일단 훈련 데이터 일부를 평가 데이터로 떼어낸다.
- 간단하게 처음부터 35개를 train으로, 이 후 14개를 test로 나누면 어떻게 될까?



```
train_input = fish_data[ :35 ]
train_target = fish_target[ :35 ]

test_input = fish_data[ 35 : ]
test_target = fish_target[ 35 : ]
```

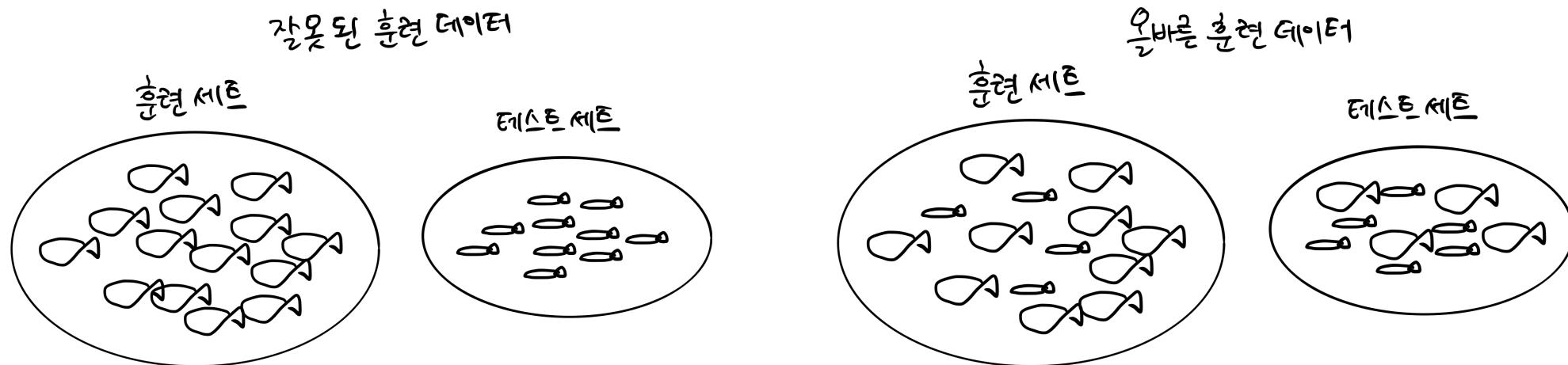
가설 실험

```
from sklearn.neighbors import KNeighborsClassifier  
  
kn = KNeighborsClassifier()  
kn = kn.fit(train_input, train_target)  
  
kn.score(test_input, test_target)  
0.0
```

- 0.0?? 0점?? why??

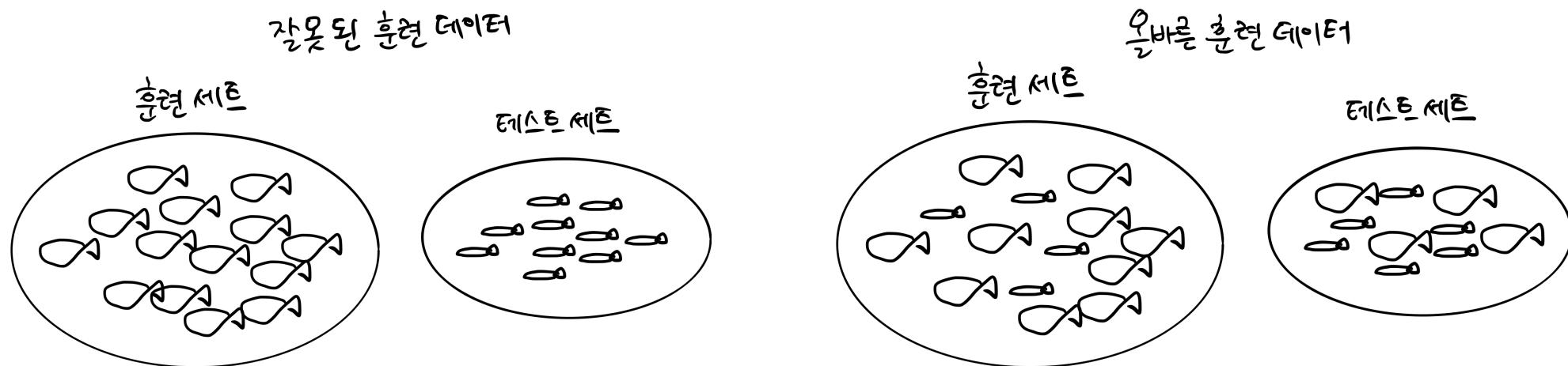
데이터 편향

- 데이터 편향이란?: 학습/평가 데이터가 각기 이질적인 데이터로 분포되어 있는 상황
 - 전체를 대변하지 않는 데이터가 표본에 많이 포함되는 경우 (exceptional 한 데이터가 많이 들어감)
- ~~간단하게 처음부터 35개를 train으로, 이후 14개를 test로 나누면 어떻게 될까?~~ (편향 발생 가능)



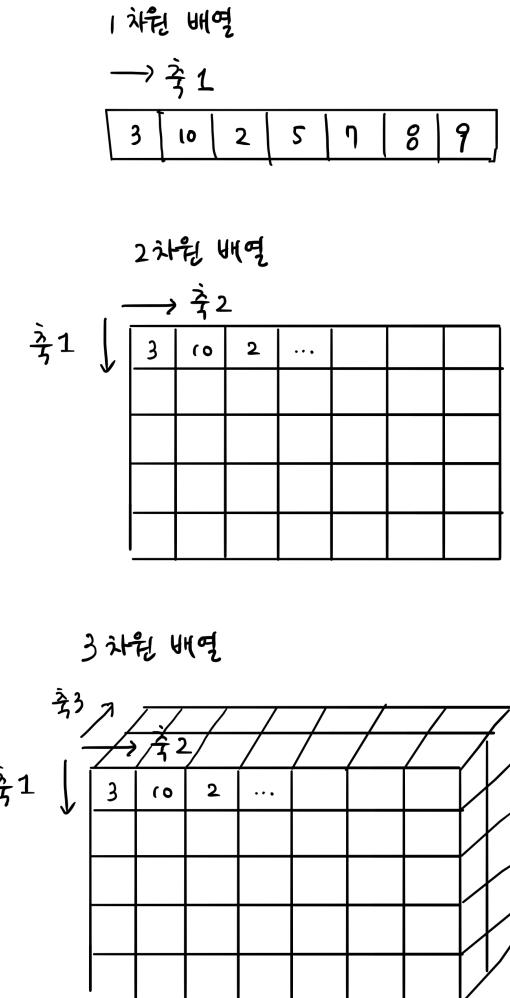
Random Sampling 방법

- 데이터 편향을 방지하기 위해 학습/평가 데이터를 무작위로 섞어 데이터를 제작



Random Sampling 방법

- numpy를 활용한 Random Sampling



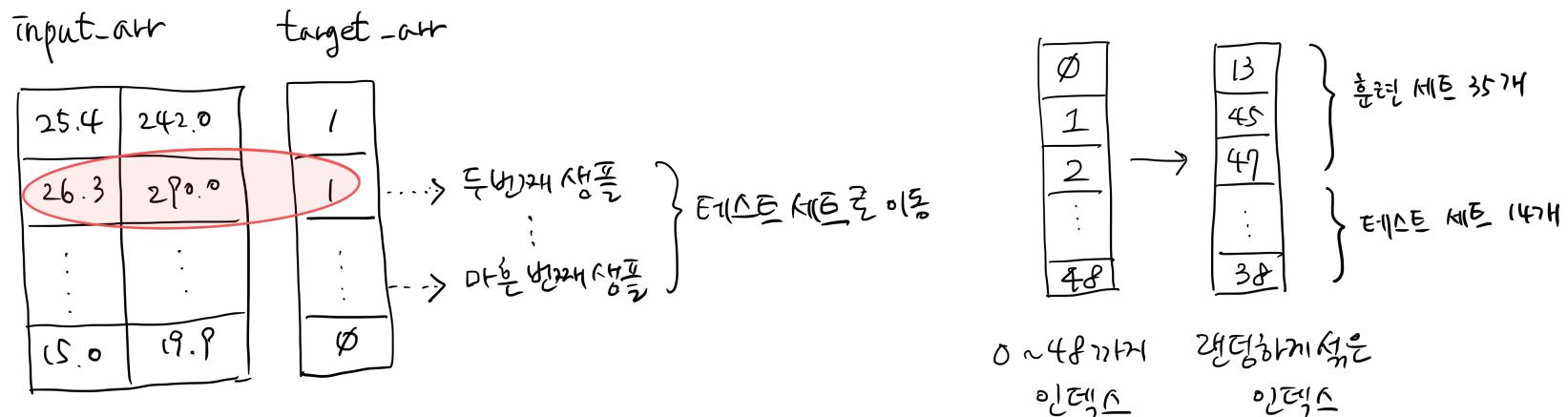
```
import numpy as np  
  
input_arr = np.array(fish_data)  
target_arr = np.array(fish_target)  
  
print(input_arr)
```

2개의 열(특성)

행(샘플) { [[25.4 , 242.0],
[26.3 , 290.0],
: :
[15.0 , 19.9]]



Random Sampling 방법



np.random.seed(42)

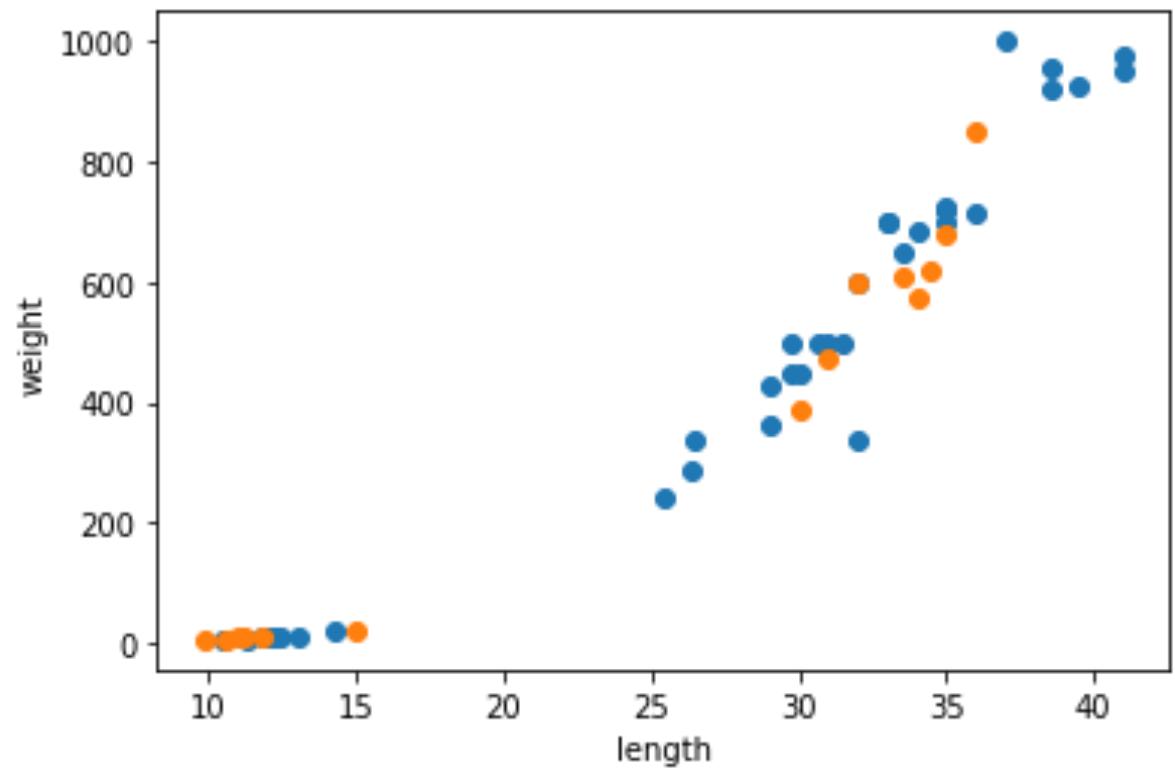
```
index = np.arange(49)  
np.random.shuffle(index)
```

```
[ 13  45  47  44  17  27  26  25  31  19  12  4  34  8  3  6  40  41  46  15  9  16  24  33  
 30   0  43  32   5  29  11  36   1  21   2  37  35  23  39  10  22  18  48  20   7  42  14  28  
 38 ]
```

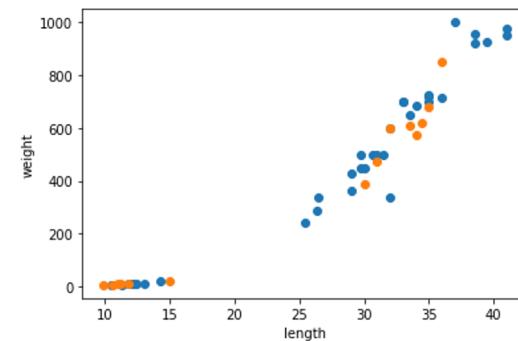
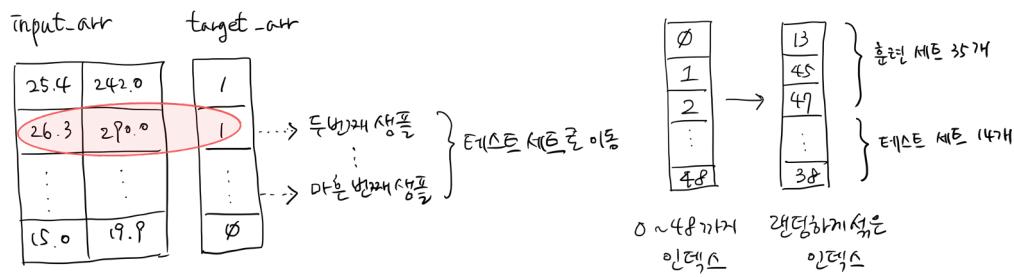
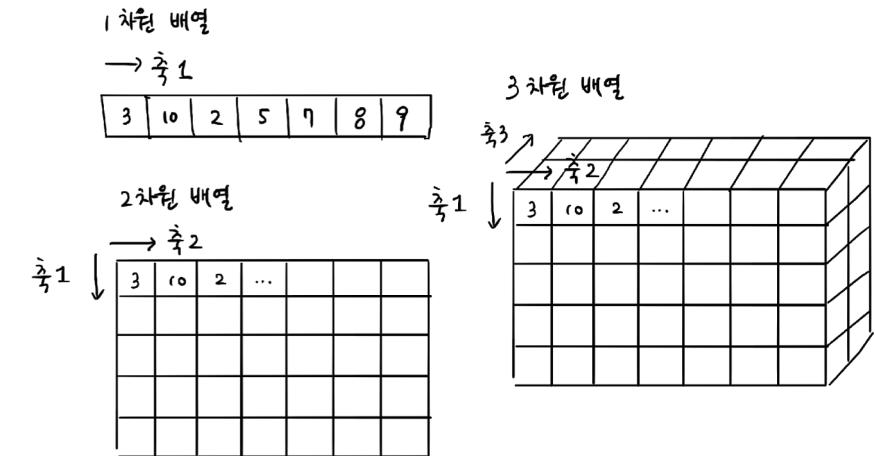
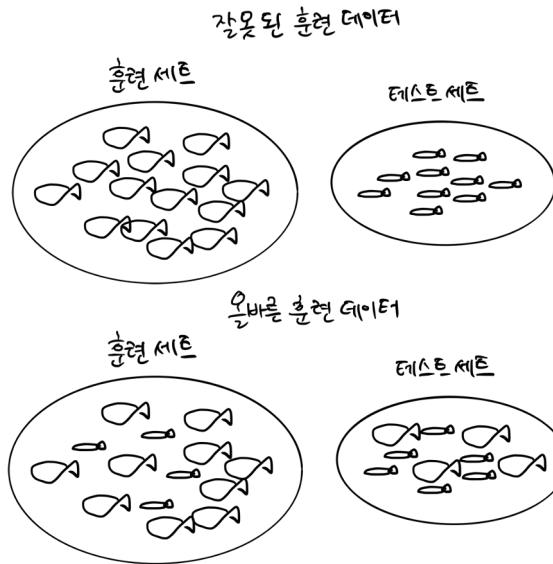
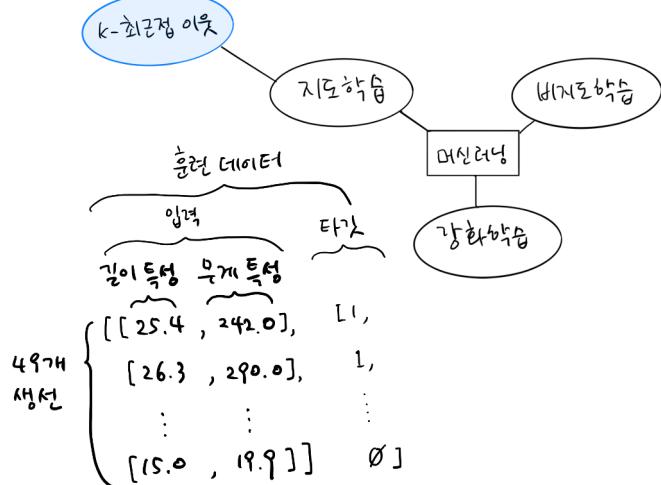
```
train_input = input_arr[index[:35]]  
train_target = target_arr[index[:35]]
```

Random Sampling 방법

```
test_input = input_arr[index[35:]]  
test_target = target_arr[index[35:]]  
  
import matplotlib.pyplot as plt  
  
plt.scatter(train_input[:, 0], train_input[:, 1])  
plt.scatter(test_input[:, 0], test_input[:, 1])  
plt.xlabel('length')  
plt.ylabel('weight')  
plt.show()
```



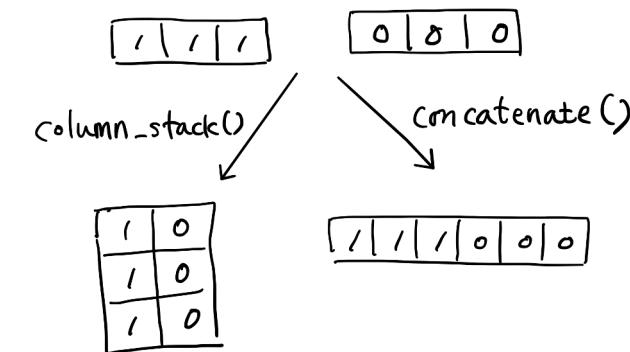
정리



scikit-learn을 활용한 Random Sampling 방법

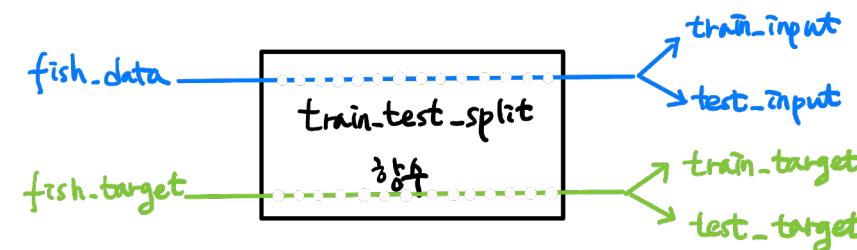
```
fish_data = np.column_stack((fish_length, fish_weight))

[[ 25.4 242. ]
 [ 26.3 290. ]
 [ 26.5 340. ]
 [ 29.  363. ]
 [ 29.  430. ]]
```



scikit-learn을 활용한 Random Sampling 방법

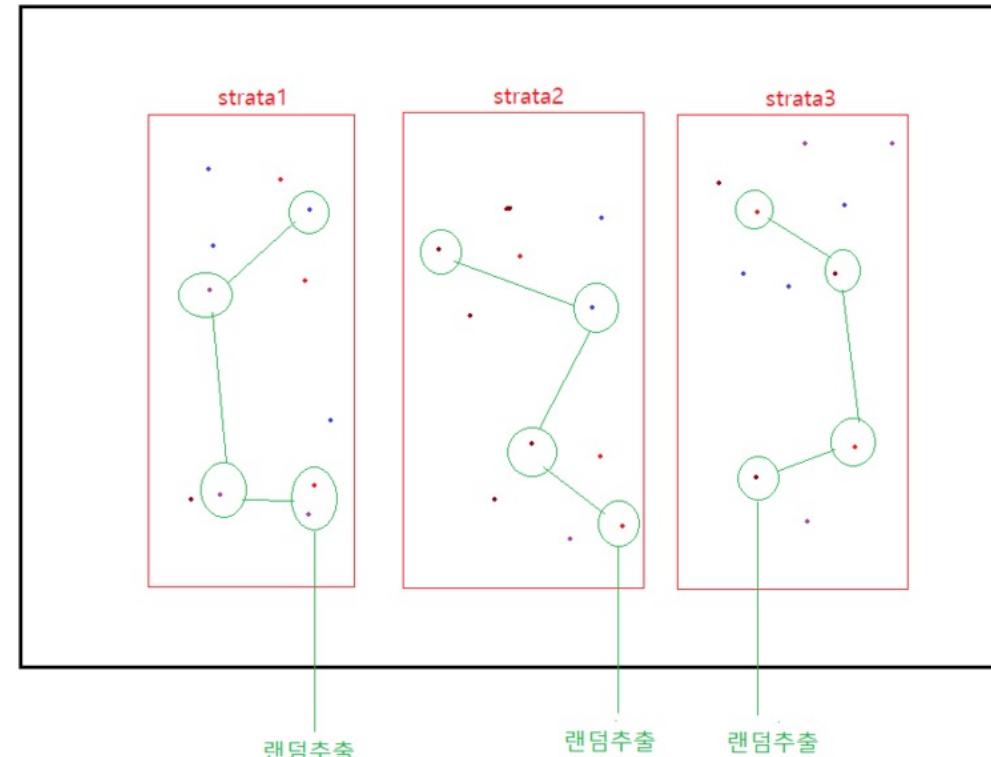
```
from sklearn.model_selection import train_test_split  
  
train_input, test_input, train_target, test_target = train_test_split(  
    fish_data, fish_target, stratify=fish_target, random_state=42)
```



scikit-learn을 활용한 Random Sampling 방법

- 랜덤샘플링의 경우에도 샘플링 편향이 발생할 수 있지 않을까?
 - 예를 들어, 전체 데이터 중 도미(35) 빙어(14) 이면 2.5:1의 비율인데 13개의 테스트 데이터 중 도미 10 빙어 3이면 3.3:1 이다. (샘플링 편향이 약간 발생함) 이를 해결하기 위해 stratify 활용

도미와 빙어를 strata그룹으로 나누고
각 그룹에서 무작위로 샘플링



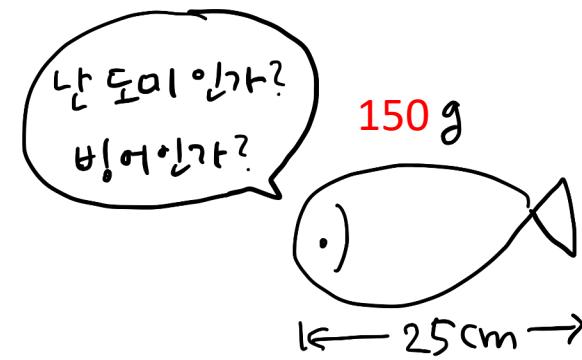
CONTENTS

1 복습

2 학습/검증 데이터

3 데이터 표준화

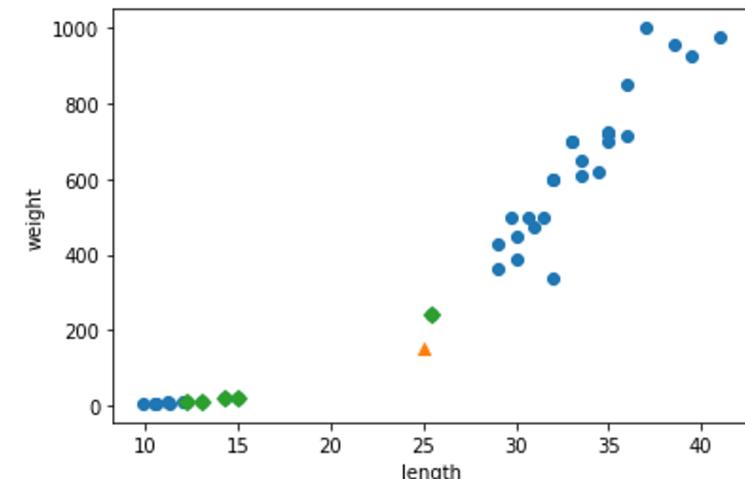
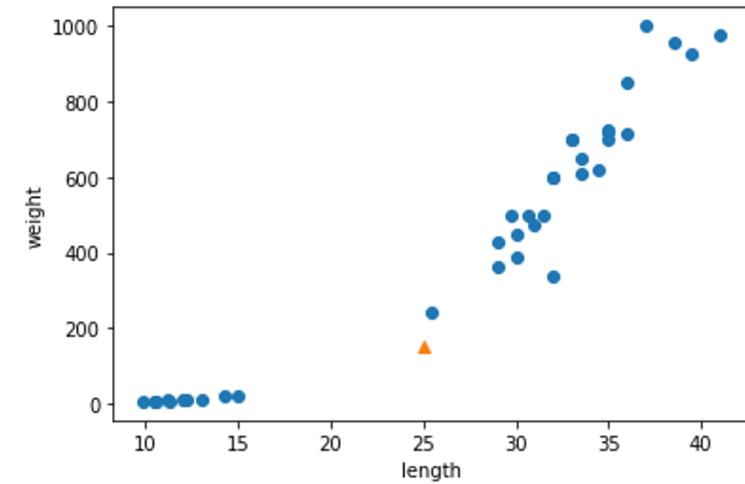
처음 보는 데이터의 예측



처음 보는 데이터의 예측

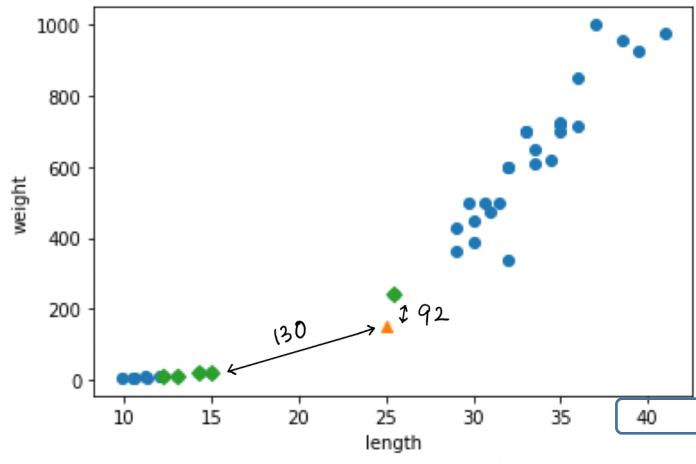
```
from sklearn.neighbors import KNeighborsClassifier  
  
kn = KNeighborsClassifier()  
kn.fit(train_input, train_target)  
kn.score(test_input, test_target)  
1.0  
  
print(kn.predict([[25, 150]]))  
[0.] • 25 cm인데 빙어라고?
```

```
distances, indexes = kn.kneighbors([[25, 150]])  
                         X축(길이)   Y축(무게)  
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.xlabel('length')  
plt.ylabel('weight')  
plt.show()
```

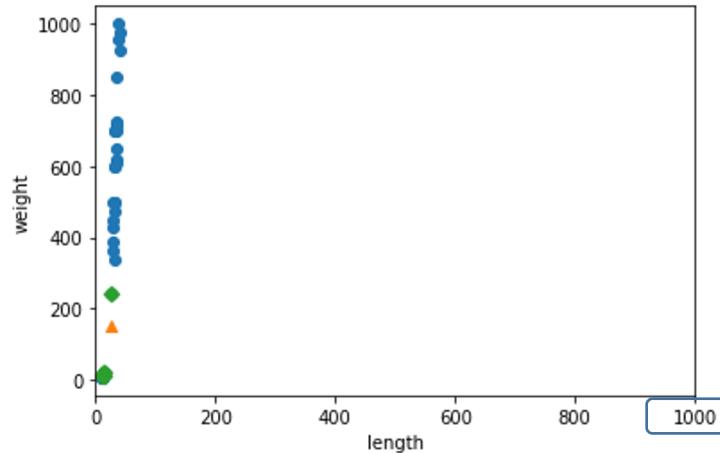


처음 보는 데이터의 예측

✎ X축과 Y축의 값의 범위를 같게 그려보면



```
distances, indexes = kn.kneighbors([[25, 150]])  
  
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.xlabel('length')  
plt.ylabel('weight')  
plt.show()
```



```
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()
```

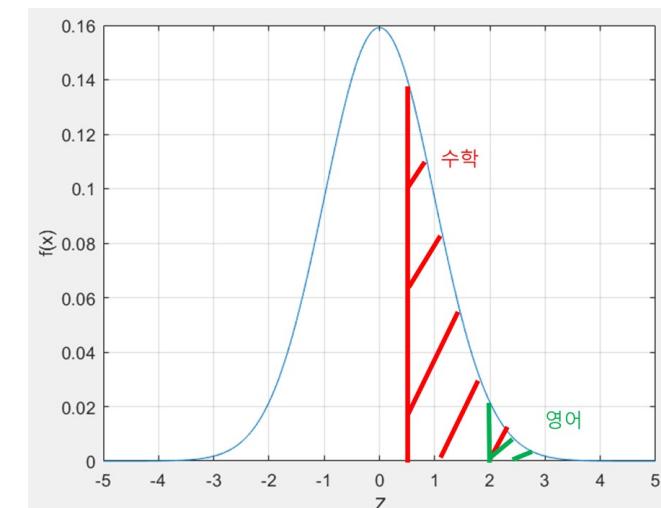


정규분포 표준화

- 정규 분포의 표준화는 평균이 m 이고 표준편차가 σ 인 정규 분포를 따르는 확률변수 X 를 평균이 0이고 표준편차가 1인 표준정규분포를 따르는 확률변수 Z 로 바꾸는 것을 의미한다.
$$Z = \frac{X - m}{\sigma}$$
- 수학 영어 시험 점수가, 수학 점수는 평균이 80 표준편차가 20, 영어 점수는 평균이 60 표준편차 10일 때 수학 90, 영어 80 받았을 때 상대적으로 어떤 점수를 잘 본 걸까?

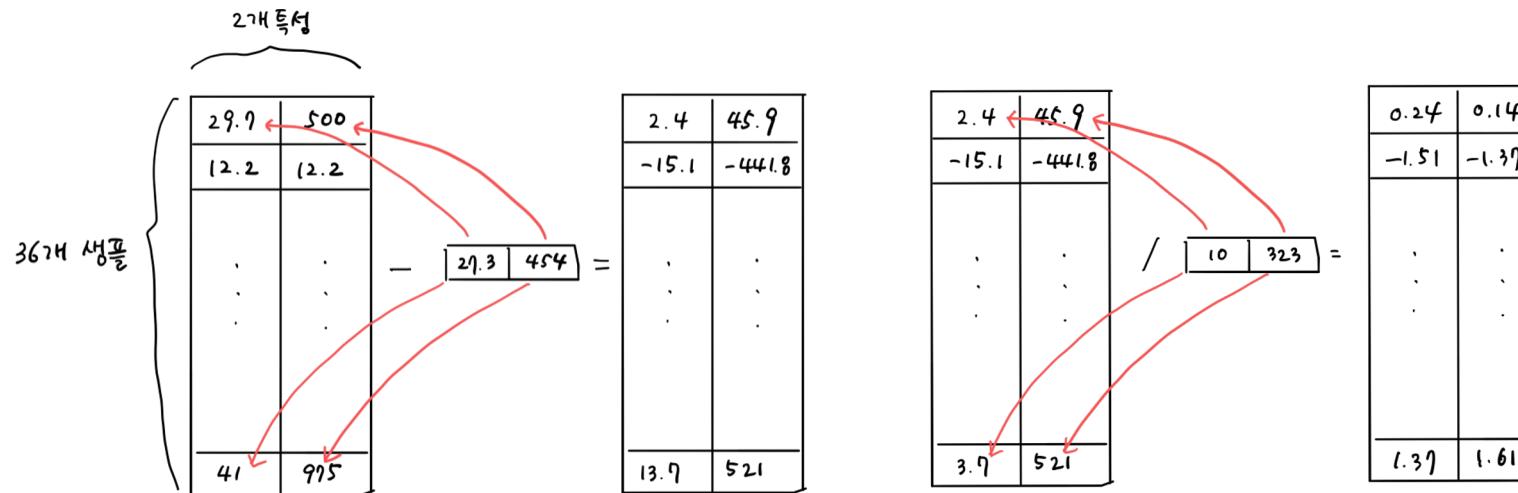
- 공정한 상대 평가를 위해 정규 분포의 표준화가 필요하다.

각각 점수가 상위 몇 %에 위치하나?



정규분포 표준화

- 길이와 무게를 공정한 기준에서 평가해볼 수 있도록 표준화 해보자
 - 표준화 할 때 36개의 샘플만 가지고 하는 이유는?



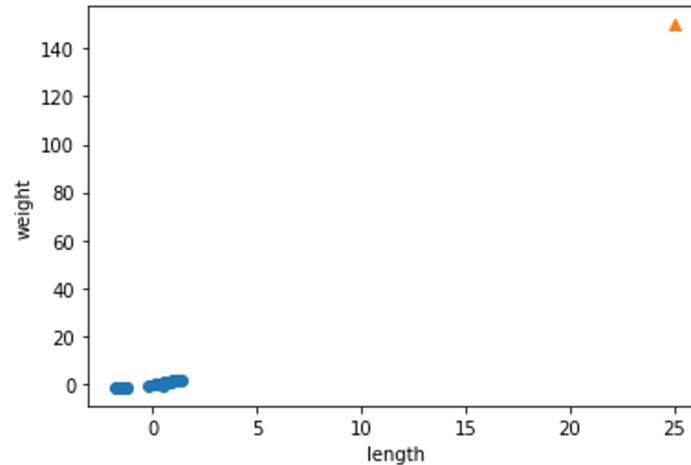
```
mean = np.mean(train_input, axis=0)
std = np.std(train_input, axis=0)

print(mean, std)
[ 27.29722222 454.09722222] [ 9.98244253 323.29893931]

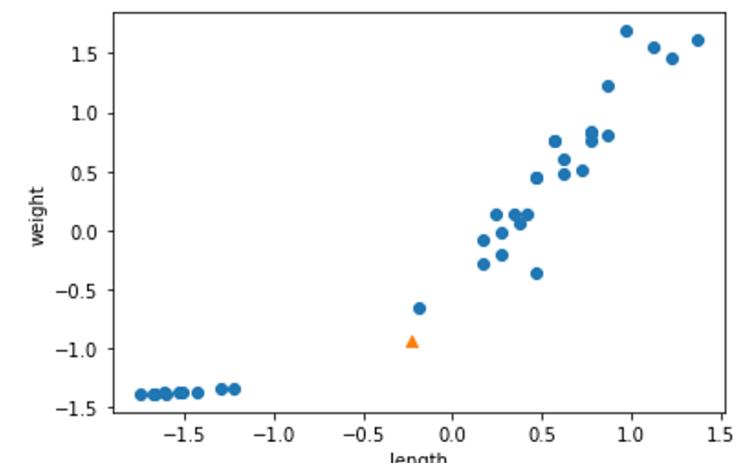
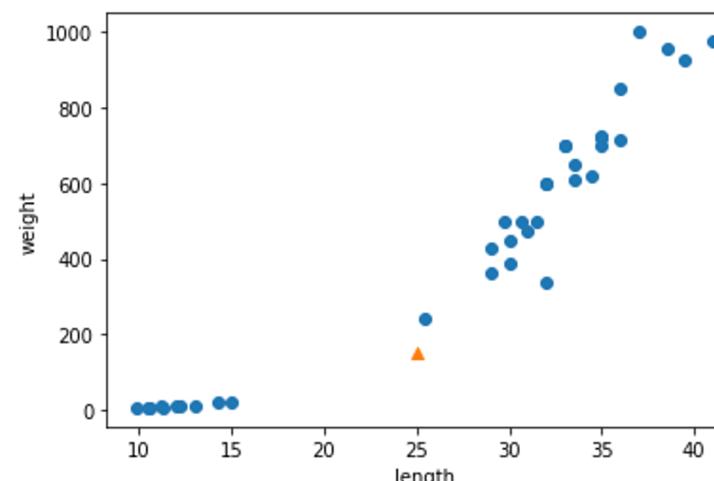
train_scaled = (train_input - mean) / std
```

표준화 이후 시각화

- 학습데이터는 표준화 하고 Test데이터를 표준화 하지 않을 경우?



```
new = ([25, 150] - mean) / std  
  
plt.scatter(train_scaled[:,0], train_scaled[:,1])  
plt.scatter(new[0], new[1], marker='^')  
plt.xlabel('length')  
plt.ylabel('weight')  
plt.show()
```



전처리 데이터를 이용한 실험

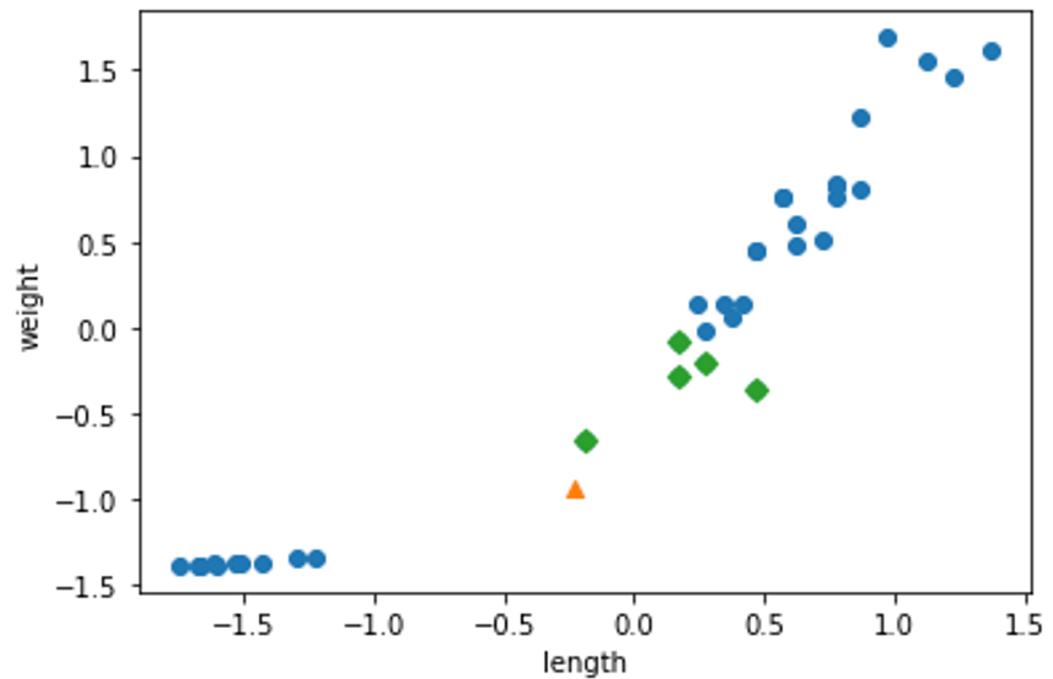
```
kn.fit(train_scaled, train_target)

test_scaled = (test_input - mean) / std
kn.score(test_scaled, test_target)
1.0

print(kn.predict([new]))
[1.]

distances, indexes = kn.kneighbors([new])

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()
```



오늘 핵심 한 줄

```
from sklearn.neighbors import KNeighborsClassifier  
  
kn = KNeighborsClassifier()  
  
kn.fit(fish_data, fish_target)  
  
kn.score(fish_data, fish_target)  
1.0
```

V.S.

```
kn = kn.fit(train_input, train_target)  
  
kn.score(test_input, test_target)  
1.0
```

CONTENTS

- 1 복습
- 2 학습/검증 데이터
- 3 표준화
- 4 더 생각해볼 만한 문제

학습(Train), 검증(Valid), 평가(Test) 데이터

- 학습 데이터: 우리가 선택한 알고리즘을 위해 학습에 활용할 데이터. (수능 문제집)

- 검증 데이터: 모델의 예측/분류 정확도를 계산할 수 있다. (수능 모의고사)

사실 모든 검증 세트에 대한 실제 레이블, 즉 정답을 알고 있지만 그렇지 않은 척 하는 셈이다

- 평가 데이터: 검증 데이터 와 비슷하지만, 모델을 구축하거나 튜닝할 때 포함된 적 없다는 점에서 차이가 있다 (수능)

분산 VS. 편향

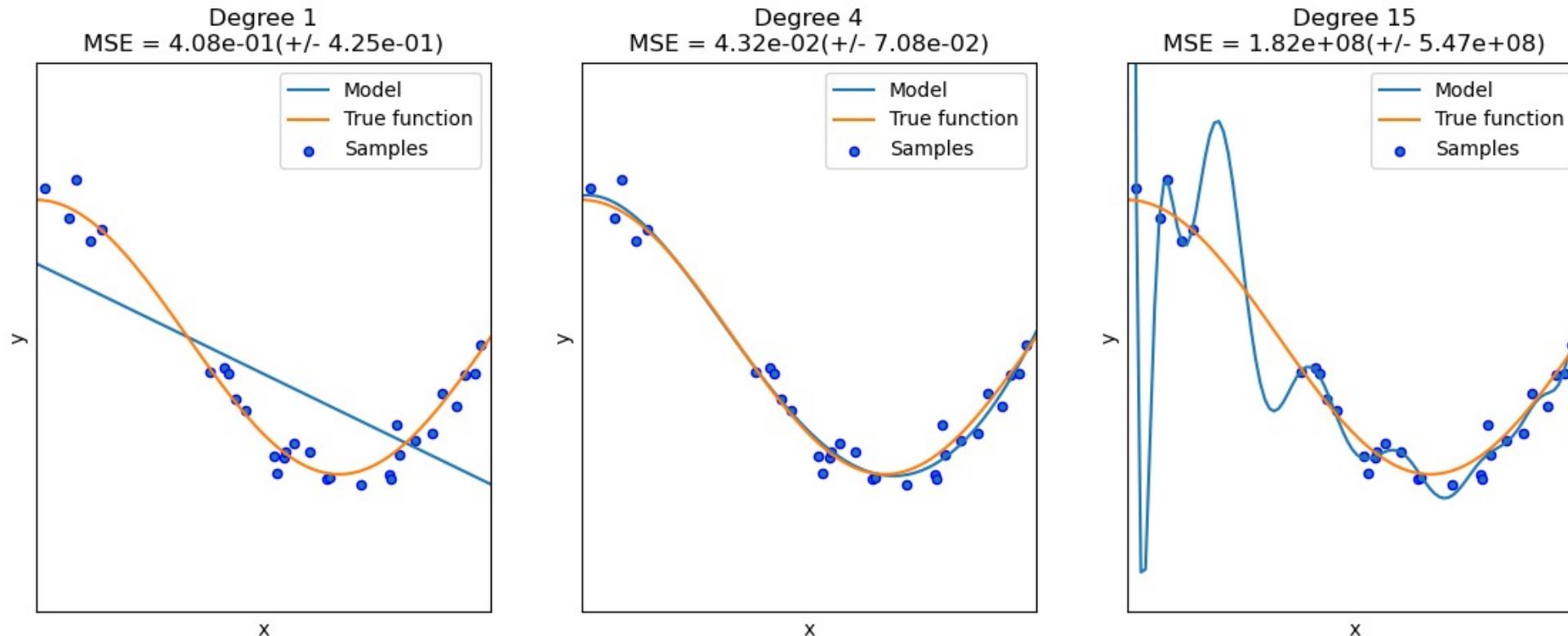
- **편향(Bias)** : 모델이 학습데이터를 충분히 표현할 수 없어 발생. bias가 높다는 건 학습을 덜했다 **underfitting** 되었다고 함. 학습 더 시키거나, 복잡한 모델을 사용하거나. 데이터 추가하여 해결.
- **분산(Variance)** : 트레이닝 데이터에 너무 민감하게 반응하여 발생함. variance가 굉장히 높아서 **overfitting** 되어 있다는 얘기. 데이터 자체에 의존하고 집착하고 일반화가 되어 있지 않다. 학습 데이터에 성능이 좋으나 테스트 데이터에 성능이 낮다.
- 편향과 분산은 서로 다르게 움직인다.
편향을 낮추면 분산이 올라가고 분산을 낮추면 편향성이 올라감 이를 편향-분산 트레이드 오프라고 함

Underfitting VS. Overfitting

- 편향(Bias) : 모델이 학습데이터를 충분히 표현할 수 없어 발생.
- 분산(Variance) : 트레이닝 데이터에 너무 민감하게 반응하여 발생함.

데이터에 분산이 크면 예측이 어려움. 분산을 낮춰서 퍼져있는 데이터를 impact있게 모으기 위해 정규화(Normalize)하는 것.

*분산이 클수록 평균을 중심으로 광범위하게 분포하고 작을 수록 평균에 몰려있다 (즉 데이터가 개판 임)



감사합니다.