1회차

Machine Learning Study

나의 첫 머신러닝, 데이터 다루기

SMARCLE
유시은 | 이정현

목차 LIST

01 나의 첫 머신러닝

인공지능과 머신러닝, 딥러닝

머신러닝이란 규칙을 일일이 프로그래밍 하지 않아도 자동으로 데이터에서 규칙을 학습하는 알고리즘을 연구하는 분야이다.

사이킷런이 머신러닝의 대표적인 라이브러리로 사용된다.

01 나의 첫 머신러닝 코랩과 주피터 노트북

코랩이란 웹 브라우저에서 파이썬 프로그램을 테스트하고 저장할 수 있는 클라우드 기반 주피터 노트북 개발 환경이다.

01 나의 첫 머신러닝 마켓과 머신러닝

01 - 3 에서는 k-최근접 이웃 알고리즘을 이용해 2개의 종류를 분류하는

머신러닝 모델을 훈련한다.

머신러닝에서는 여러 종류(혹은 클래스(class))를 구별하는 문제를 분류(classification)

라고 하고, 2개의 클래스 중 하나를 고르는 문제는 이진 분류(binary classification)라고

한다.

01 나의 첫 머신러닝 마켓과 머신러닝

도미 데이터 준비

```
[2]  bream_length = [25.4, 26.3, 26.5, 29.0, 29.0, 29.7, 29.7, 30.0, 30.0, 30.7, 31.0, 31.0,
                     31.5, 32.0, 32.0, 32.0, 33.0, 33.0, 33.5, 33.5, 34.0, 34.0, 34.5, 35.0,
                     35.0, 35.0, 35.0, 36.0, 36.0, 37.0, 38.5, 38.5, 39.5, 41.0, 41.0] #단위는 cm
     bream_weight = [242.0, 290.0, 340.0, 363.0, 430.0, 450.0, 500.0, 390.0, 450.0, 500.0, 475.0, 500.0,
                     500.0, 340.0, 600.0, 600.0, 700.0, 700.0, 610.0, 650.0, 575.0, 685.0, 620.0, 680.0,
                     700.0, 725.0, 720.0, 714.0, 850.0, 1000.0, 920.0, 955.0, 925.0, 975.0, 950.0] #단위는 g
```

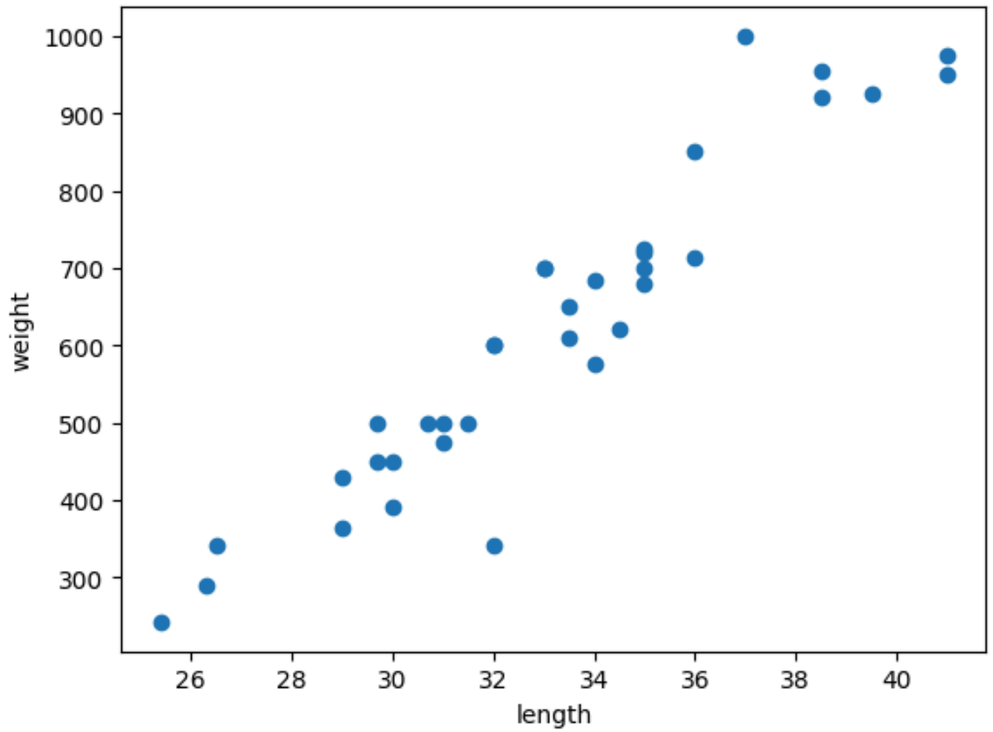
01 나의 첫 머신러닝 마켓과 머신러닝

matplotlib 패키지를 import

데이터의 산점도

```
import matplotlib.pyplot as plt

plt.scatter(bream_length, bream_weight) #x축이 길이, y축이 무게
plt.xlabel('length') #x축 이름 지정
plt.ylabel('weight') #y축 이름 지정
plt.show()
```



선형(linear)적인 그래프 생성

01 나의 첫 머신러닝 마켓과 머신러닝

빙어 데이터 준비

```
[7]  smelt_length = [9.8, 10.5, 10.6, 11.0, 11.2, 11.3, 11.8, 11.8, 12.0, 12.2, 12.4, 13.0, 14.3, 15.0]
     smelt_weight = [6.7, 7.5, 7.0, 9.7, 9.8, 8.7, 10.0, 9.9, 9.8, 12.2, 13.4, 12.2, 19.7, 19.9]
```

빙어 데이터가 추가된 산점도

```
plt.scatter(bream_length, bream_weight)
plt.scatter(smelt_length, smelt_weight)
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
```

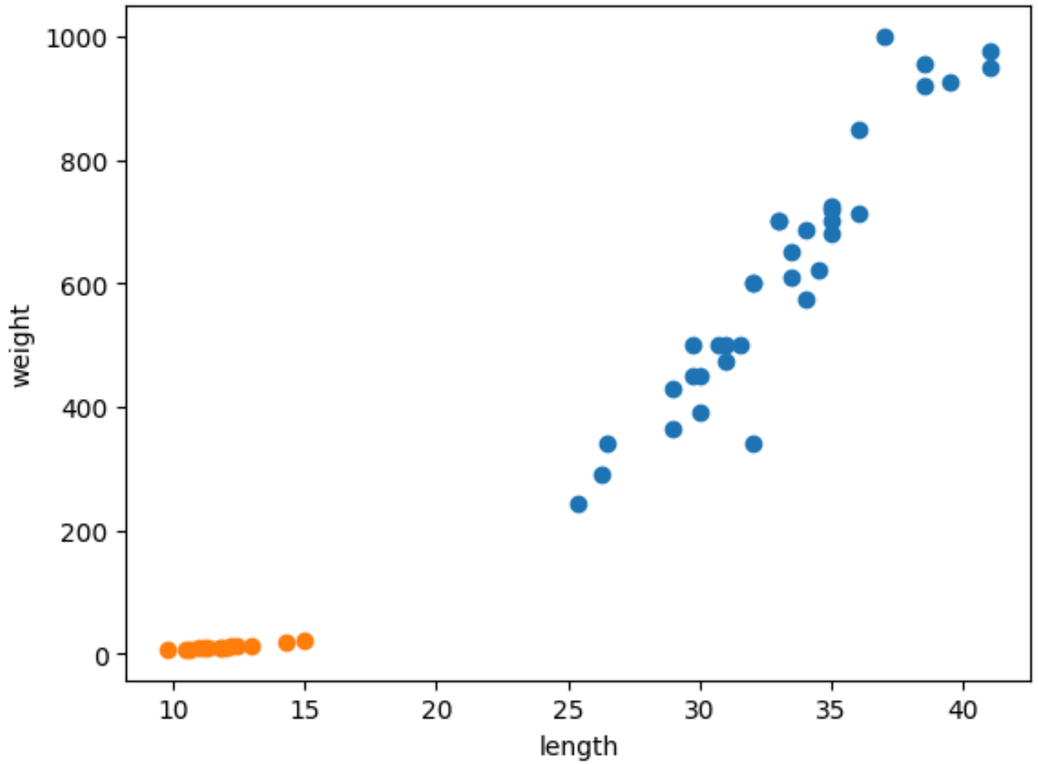
01 나의 첫 머신러닝 마켓과 머신러닝

자동적으로 도미와 빙어의 데이터가

다른 색깔으로 구분됨.

도미와 빙어의 산점도는 모두 선형적이지만

다른 특성을 가짐.



01 나의 첫 머신러닝 마켓과 머신러닝

K-최근접 이웃(K Nearest Neighbor) 알고리즘이란 새로운 개체의 목표값을 예측하기 위하여 과거의 유사한 데이터를 이용하여 그 값을 예측하는 것이다.

준비된 도미와 빙어의 데이터를 k-최근접 이웃 알고리즘을 이용해 구분하도록 한다.

01 나의 첫 머신러닝 마켓과 머신러닝

두 리스트를 병합

```
[9]   length = bream_length + smelt_length
      weight = bream_weight + smelt_weight
```

zip() 함수를 이용해 두 리스트를 합친 2차원 리스트 생성

```
[12]  fish_data = [[l, w] for l, w in zip(length, weight)] #zip()은 나열된 리스트에서 원소를 하나씩 꺼내서 반환
```

01 나의 첫 머신러닝 마켓과 머신러닝

도미가 찾는 대상이기에 도미를 1, 빙어를 0에 대응하는 리스트 생성

```
[13]  fish_target = [1] * 35 + [0] * 14 #도미가 앞에서부터 35마리, 빙어가 그 뒤로 14마리
```

사이킷런 패키지에서 클래스 임포트

```
[14]  from sklearn.neighbors import KNeighborsClassifier #KNN 알고리즘을 구현한 클래스 import

      kn = KNeighborsClassifier() #KNeighborsClassifier 객체 생성
```

01 나의 첫 머신러닝 마켓과 머신러닝

훈련(training)을 위해 fit( ) 메서드 사용

```
kn.fit(fish_data, fish_target)
```

fit( ) 메서드는 주어진 데이터로 알고리즘 훈련

01 나의 첫 머신러닝 마켓과 머신러닝

훈련된 모델의 정확도 평가

```
kn.score(fish_data, fish_target)
```

```
1.0
```

score( ) 메서드는 모델을 평가하고
0에서 1 사이의 값 반환

01 나의 첫 머신러닝 마켓과 머신러닝

예시가 어느 집단에 가까운지 판단

```
kn.predict([[40, 500]])
```

```
array([1])
```

몸 길이 40cm, 무게 500g인 예시를
1, 즉 도미라고 판단

```
kn.predict([[10, 30]])
```

```
array([0])
```

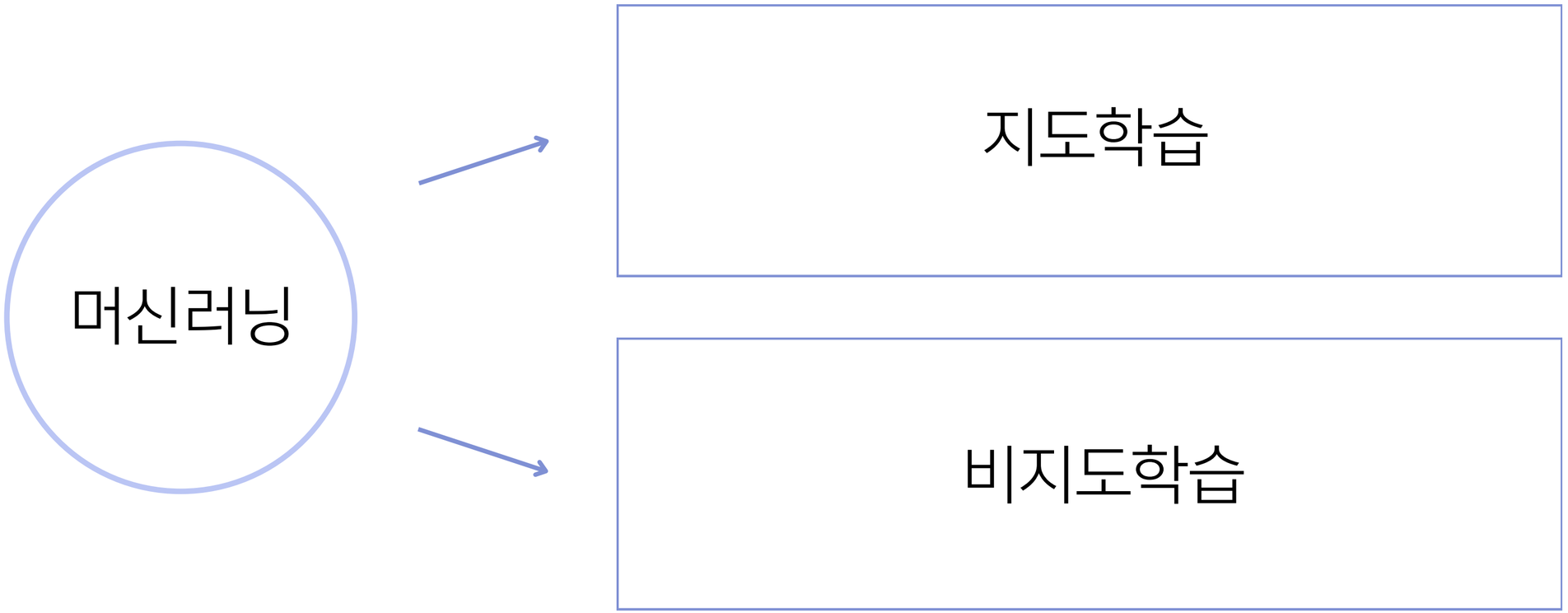
몸 길이 10cm, 무게 30g인 예시를
0, 즉 빙어라고 판단

01 나의 첫 머신러닝 마켓과 머신러닝

KNN 알고리즘은 전달받은 데이터를 모두 저장하고, 새로운 데이터를 받으면
가장 가까운 데이터를 참고하여 판단

판단할 때 참고하는 데이터의 기본 개수는 변경 가능

02 데이터 다루기 훈련 세트와 테스트 세트

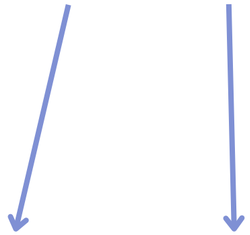


02 데이터 다루기 훈련 세트와 테스트 세트

지도학습	입력 데이터와 그에 대응하는 답을 사용하여 학습
비지도학습	입력 데이터만 제공하여 패턴이나 특징을 학습

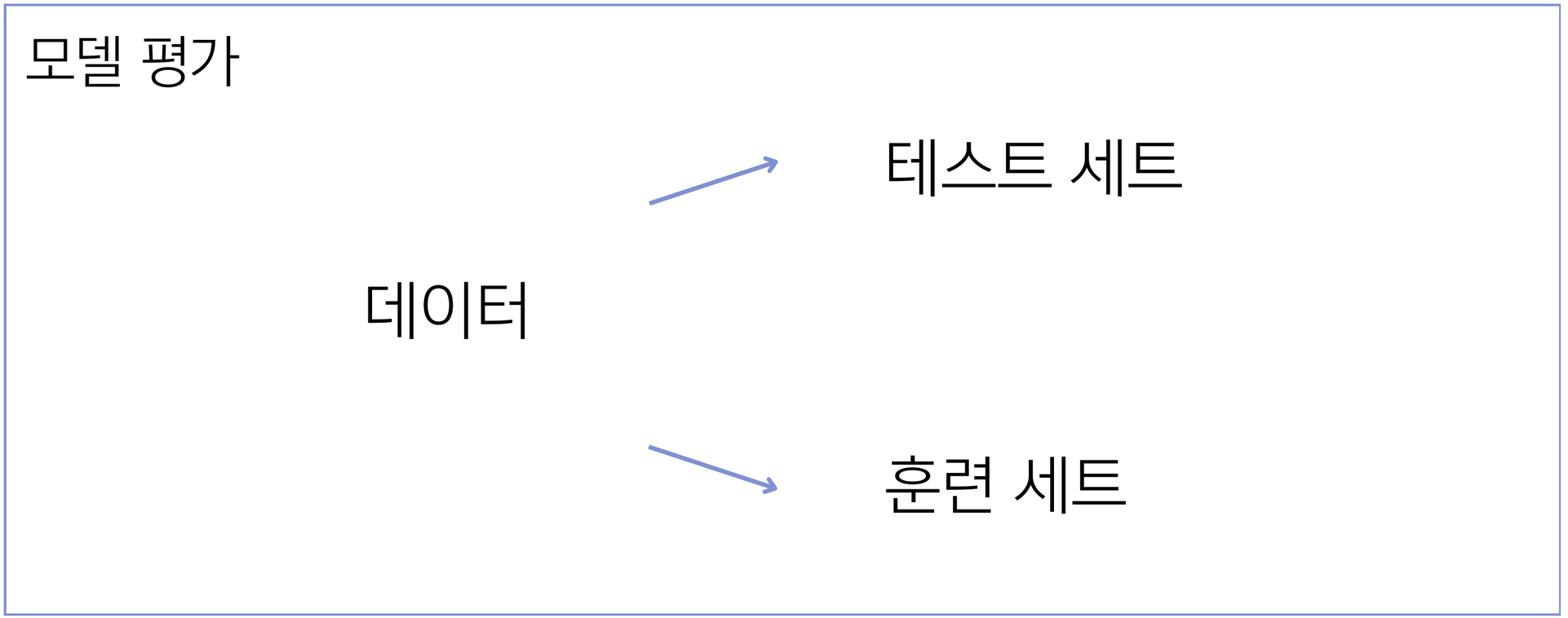
02 데이터 다루기 훈련 세트와 테스트 세트

지도학습 | 훈련하기 위한 데이터, 정답 필요



훈련 데이터 = 입력 + 타깃

02 데이터 다루기 훈련 세트와 테스트 세트



02 데이터 다루기 훈련 세트와 테스트 세트

```
[ ]  # 생선 데이터
     fish_length = [25.4, 26.3, 26.5, 29.0, 29.0, 29.7, 29.7, 30.0, 30.0, 30.7, 31.0, 31.0,
                    31.5, 32.0, 32.0, 32.0, 33.0, 33.0, 33.5, 33.5, 34.0, 34.0, 34.5, 35.0,
                    35.0, 35.0, 35.0, 36.0, 36.0, 37.0, 38.5, 38.5, 39.5, 41.0, 41.0, 9.8,
                    10.5, 10.6, 11.0, 11.2, 11.3, 11.8, 11.8, 12.0, 12.2, 12.4, 13.0, 14.3, 15.0]
     fish_weight = [242.0, 290.0, 340.0, 363.0, 430.0, 450.0, 500.0, 390.0, 450.0, 500.0, 475.0, 500.0,
                    500.0, 340.0, 600.0, 600.0, 700.0, 700.0, 610.0, 650.0, 575.0, 685.0, 620.0, 680.0,
                    700.0, 725.0, 720.0, 714.0, 850.0, 1000.0, 920.0, 955.0, 925.0, 975.0, 950.0, 6.7,
                    7.5, 7.0, 9.7, 9.8, 8.7, 10.0, 9.9, 9.8, 12.2, 13.4, 12.2, 19.7, 19.9]
```

```
[ ]  fish_data = [[l,w] for l,w in zip(fish_length, fish_weight)]
     fish_target = [1]*35 + [0]*14
```

02 데이터 다루기 훈련 세트와 테스트 세트

```
[ ]  from sklearn.neighbors import KNeighborsClassifier
     kn = KNeighborsClassifier()
```

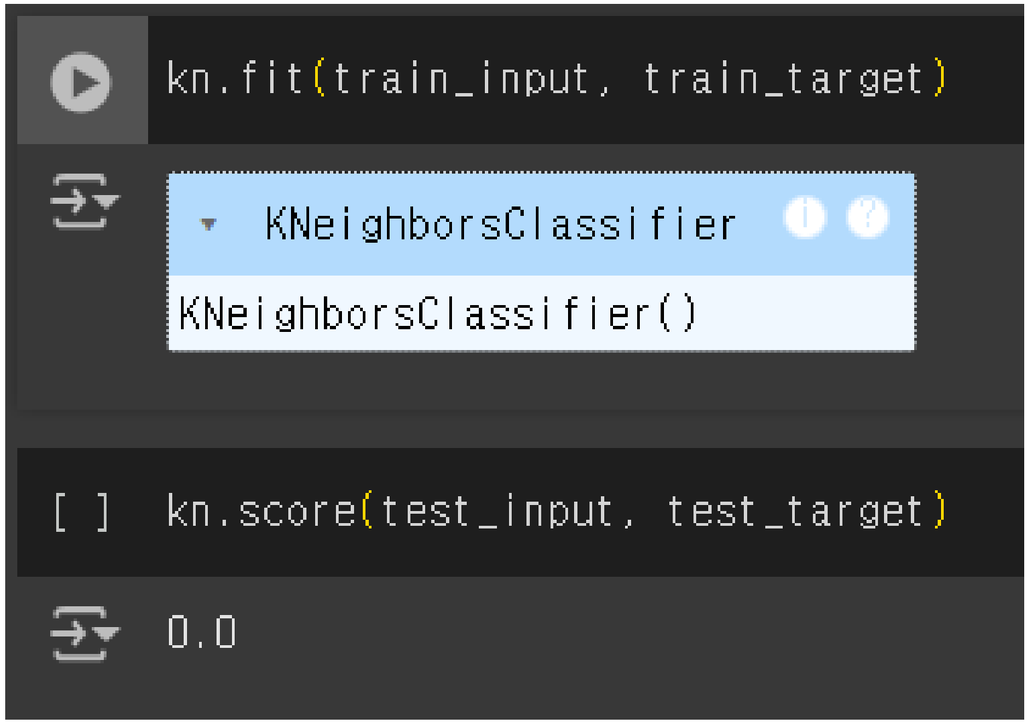
```
[ ]  train_input = fish_data[:35]
     train_target = fish_target[:35]

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

모델 지정

훈련 / 테스트 세트

02 데이터 다루기 훈련 세트와 테스트 세트



모델 학습

모델 정확도 평가

02 데이터 다루기 훈련 세트와 테스트 세트

```
[ ]  kn.score(test_input, test_target)
```

```
0.0
```

모델 정확도 평가

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

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

→ 훈련 세트에는 도미 데이터만 있고,

테스트 세트에는 빙어 데이터만 있음

02 데이터 다루기 훈련 세트와 테스트 세트

샘플링 편향

- 훈련 세트와 테스트 세트에 샘플이 골고루 섞여 있지 않은 상태

02 데이터 다루기 훈련 세트와 테스트 세트

```
[ ]  import numpy as np

     input_arr = np.array(fish_data)  # 입력 배열
     target_arr = np.array(fish_target)  # 타겟 배열

     np.random.seed(42)  # 시드 설정
     index = np.arange(49)  # 0~48로 구성된 배열 생성
     np.random.shuffle(index)  # 배열 섞기
```

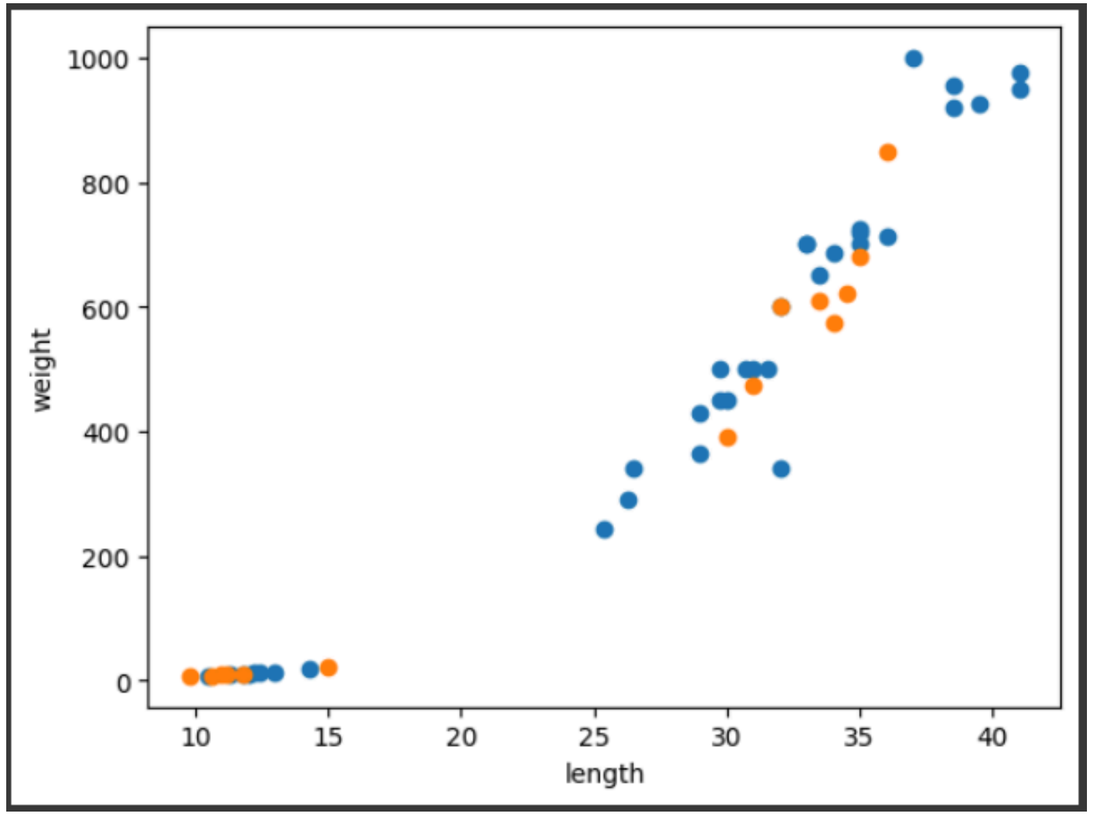
02 데이터 다루기 훈련 세트와 테스트 세트

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

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()
```



02 데이터 다루기 훈련 세트와 테스트 세트

```
[ ]  kn.fit(train_input, train_target)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
[ ]  kn.score(test_input, test_target)
```

```
1.0
```

```
[ ]  kn.predict(test_input)
```

```
array([0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0])
```

```
[ ]  test_target
```

```
array([0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0])
```

02 데이터 다루기 데이터 전처리

```
[2]  import numpy as np

     fish_data = np.column_stack((fish_length, fish_weight))
     fish_target = np.concatenate((np.ones(35), np.zeros(14)))
```

02 데이터 다루기 데이터 전처리

```
[5]  from sklearn.model_selection import train_test_split

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

```
array([1., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

✅random_state=n # 랜덤 시드 설정하는 파라미터

02 데이터 다루기 데이터 전처리

```
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)
test_target
```

```
array([0., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 1.])
```

✅stratify=(target data) # 비율에 맞게 데이터를 분배해주는 파라미터

02 데이터 다루기 데이터 전처리

```
[7]  from sklearn.neighbors import KNeighborsClassifier

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

```
1.0
```

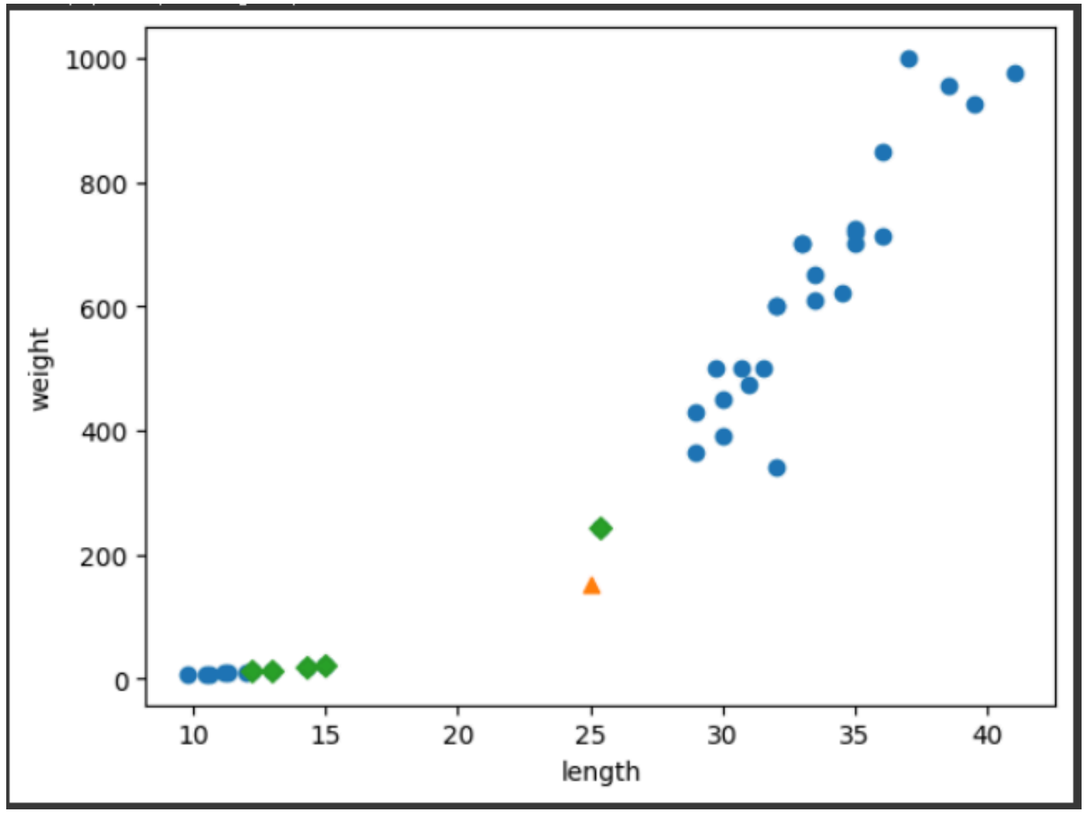
```
[8]  kn.predict([[25,150]])
```

```
array([0.])
```

02 데이터 다루기 데이터 전처리

```
import matplotlib.pyplot as plt

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



스케일이 다름

02 데이터 다루기 데이터 전처리

데이터 전처리

- 특성 값을 일정한 기준으로 맞춤

02 데이터 다루기 데이터 전처리

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

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

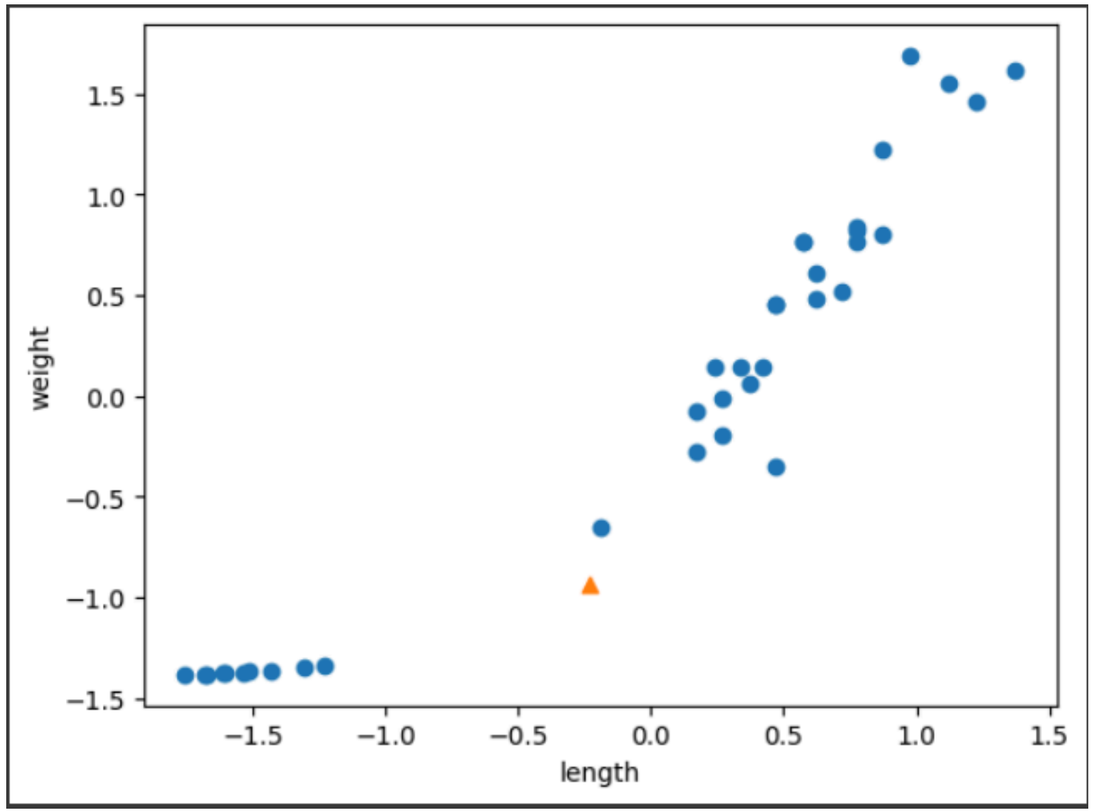
✅표준점수 : (값 – 평균) / 표준편차

02 데이터 다루기 데이터 전처리

```
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()
```

⚠️샘플도 동일한 작업을 해주어야 함



02 데이터 다루기 데이터 전처리

```
[14] kn.fit(train_scaled, train_target)  # 모델 학습
     test_scaled = (test_input - mean) / std  # 테스트 세트 표준점수로 변환
     kn.score(test_scaled, test_target)  # 모델 평가
```

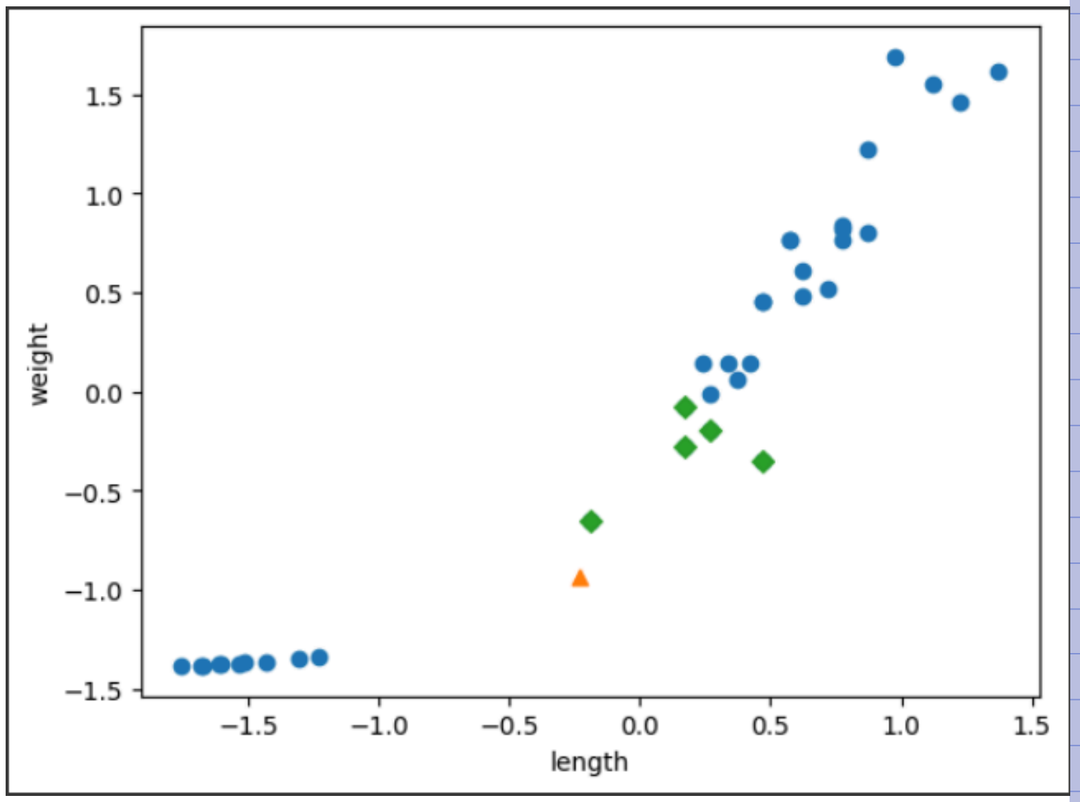
```
1.0
```

```
[15] kn.predict([new])
```

```
array([1.])
```

02 데이터 다루기 데이터 전처리

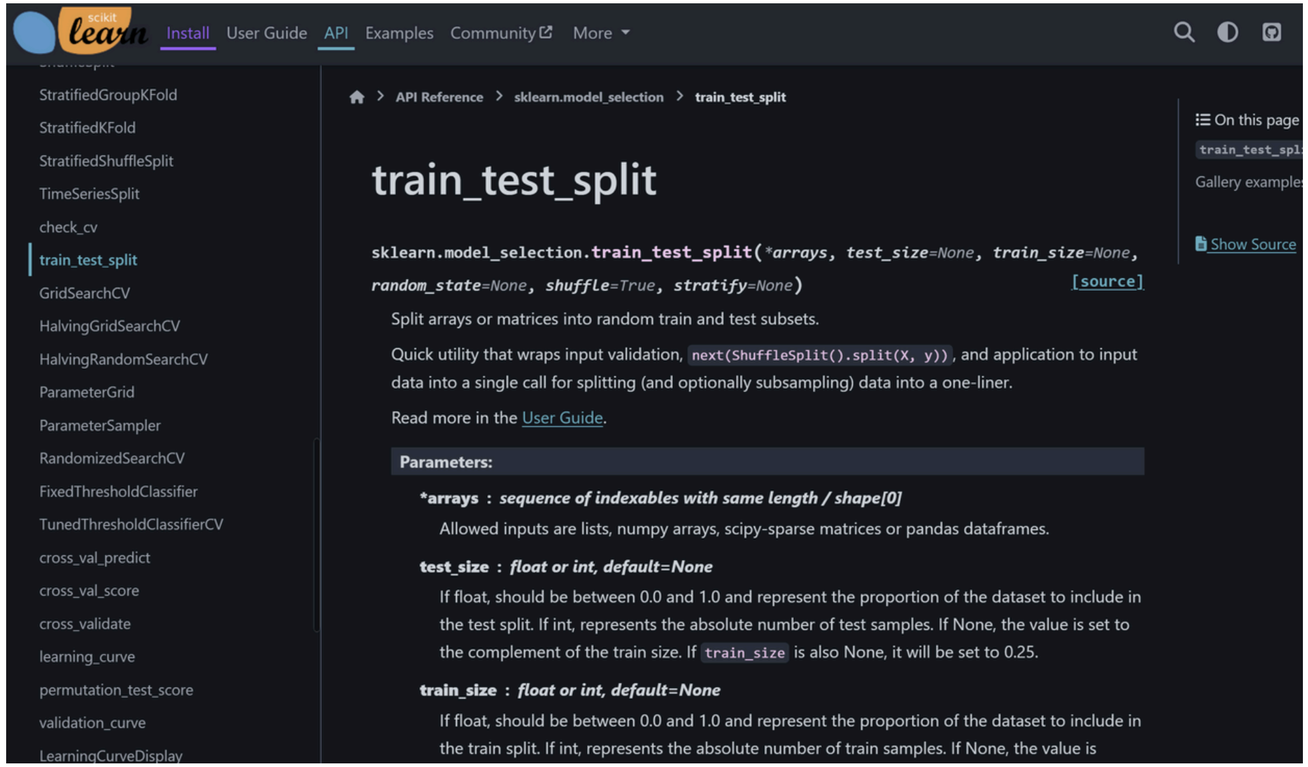
```
distances, indexes = kn.kneighbors([new])
plt.scatter(train_scaled[:,0], train_scaled[:,1])
plt.scatter(new[0], new[1], marker='^')
plt.scatter(train_scaled[indexes,0], train_scaled[indexes,1], marker='D')
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
```



03 질의응답 Q & A

궁금한 점이 있다면 자유롭게 질문해 주세요.

1회차 과제



제공된 사이킷런 페이지의 정리 및 분석

제출기한: 1/9(목) 23시 59분까지

THANK YOU

감사합니다