혼자 공부하는 머신러닝 + 딥러닝

CH02. 데이터 다루기

인공지능공학과 12223557 여예진

목차

1

첫번째 생선 분류 ML 모델

훈련 데이터

= 입력 + 타깃

1st) 입력 데이터(생선 데이터) 만들기

도미 데이터와 방어 데이터를 하나의 데이터로 합치기

- +(더하기) 사용

```
# 도미 : 35개   방어 : 14개
length = bream_length + smelt_length
weight = bream_weight + smelt_weight
```

- zip() 함수 사용
- 리스트 내포(list comprehension) 사용

```
# 생선 49개의 길이, 무게
fish_data = [[l, w] for l, w in zip(length, weight)]
print(fish_data)
```

```
[[25.4, 242.0], [26.3, 290.0], [26.5, 340.0], [29.0, 363.0], [29.0, 430.0], [29.7, 450.0], [29.7, 500.0], [30.0, 390.0], [30.0,
450.0], [30.7, 500.0], [31.0, 475.0], [31.0, 500.0], [31.5, 500.0], [32.0, 340.0], [32.0, 600.0], [32.0, 600.0], [33.0, 700.0],
[33.0, 700.0], [33.5, 610.0], [33.5, 650.0], [34.0, 575.0], [34.0, 685.0], [34.5, 620.0], [35.0, 680.0], [35.0, 700.0], [35.0, 7
25.0], [35.0, 720.0], [36.0, 714.0], [36.0, 850.0], [37.0, 1000.0], [38.5, 920.0], [38.5, 955.0], [39.5, 925.0], [41.0, 975.0],
[41.0, 950.0], [9.8, 6.7], [10.5, 7.5], [10.6, 7.0], [11.0, 9.7], [11.2, 9.8], [11.3, 8.7], [11.8, 10.0], [11.8, 9.9], [12.0, 9.
8], [12.2, 12.2], [12.4, 13.4], [13.0, 12.2], [14.3, 19.7], [15.0, 19.9]]
```

2nd) 타깃 데이터 만들기

도미 : 1, 빙어 : 0

- in ML) 2개를 구분하는 경우 -> 찾으려는 대상을 1로 놓고, 그 외에는 0으로 놓는다.

```
# 정답 데이터 리스트
fish_target = [1] * 35 + [0] * 14
print(fish_target)
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0]
```

3rd) K-NN 알고리즘 이용하여 훈련하기

KNeighborsClassifier 클래스 이용

```
from sklearn.neighbors import KNeighborsClassifier

kn = KNeighborsClassifier()
```

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

```
1.0
```

같은 데이터로 테스트한다면 모두 맞히는 것은 당연하지 않은가?

2

훈련 세트 & 테스트 세트 나누기

두번째 생선 분류 ML 모델

훈련 세트 & 테스트 세트

준비 방법

1) 평가를 위한 또 다른 데이터 준비하기
2) 이미 준비된 데이터 중 일부를 떼어내기 (일반적 경우)

훈련 세트

훈련에 사용되는 데이터

테스트 세트

평가에 사용되는 데이터

1st) 훈련 데이터 만들기

2nd) 훈련 세트 & 테스트 세트 나누기

슬라이싱(slicing) 연산 이용

```
# 훈련 세트 선택하기
train_input = fish_data[:35]  # 입력값
train_target = fish_target[:35] # 타깃값
# 테스트 세트 선택하기
test_input = fish_data[35:]  # 입력값
test_target = fish_target[35:] # 타깃값
```

3rd) K-NN 알고리즘 이용하여 훈련하기

KNeighborsClassifier 클래스 이용

```
# 사이킷런의 KNeighborsClassifier 클래스 임포트 하기
from sklearn.neighbors import KNeighborsClassifier
# 모델 객체 만들기
kn = KNeighborsClassifier()

# 데이터로 알고리즘 훈련시키기
kn = kn.fit(train_input, train_target)
# 모델 평가하기
kn.score(test_input, test_target)
```

0.0

샘플링 편향 문제 발생 – 훈련 세트 : 도미만 있음, 테스트 세트 : 빙어만 있음

3

train_test_split()

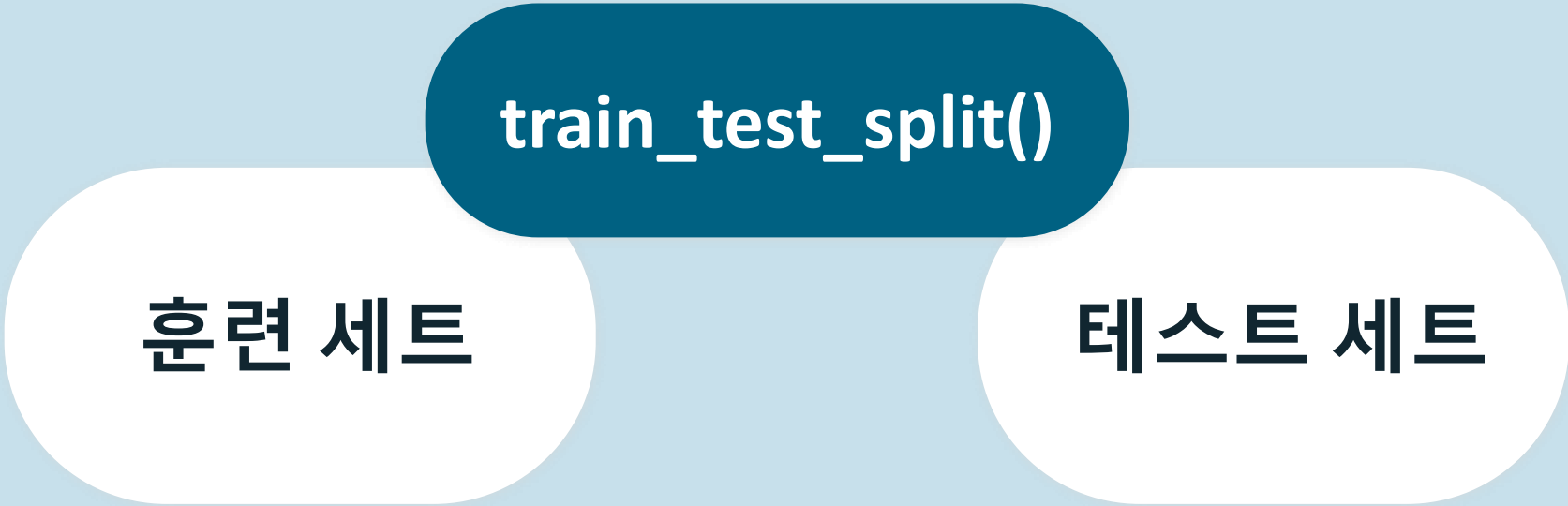
세번째 생선 분류 ML 모델

train_test_split()

전달되는 리스트나 배열(훈련 데이터)을 비율에 맞게 훈련 세트와 테스트 세트로 나누어준다.

매개변수

- random_state : 랜덤 시드 지정
- shuffle : 훈련, 테스트 세트로 나누기 전에 무작위로 섞을지 여부 결정 (기본값 = True)
- test_size : 테스트 세트로 나눌 비율 지정 (기본값 = 0.25)



1st) 훈련 데이터 만들기

numpy 이용

- 입력 데이터 만들기

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

- 타깃 데이터 만들기

```
fish_target = np.concatenate((np.ones(35), np.zeros(14)))
```

2nd) 훈련 세트 & 테스트 세트 나누기

train_test_split() 이용

```
# train_test_split() 함수 임포트하기
from sklearn.model_selection import train_test_split
# 훈련 세트와 테스트 세트 나누기
train_input, test_input, train_target, test_target = train_test_split(fish_data, fish_target, random_state=42)
```

3rd) 각 클래스(도미와 빙어)가 잘 섞여있는지 확인하기

```
# 도미와 빙어 잘 섞였는지 확인하기
print(test_target)
```

```
[1. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

[의문] *Q. 3rd에서 왜 테스트 세트(test_target)에서의 비율만 확인했지?*

- (위 글을 통해) 훈련 세트와 테스트 세트에 샘플의 클래스 비율이 일정해야 하니까
- 훈련 세트(train_target)에서의 비율도 확인해보자.

```
# 도미와 빙어 잘 섞였는지 확인하기 (train_target 이용)
print(train_target)
```

```
[1. 1. 1. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0.
 1. 0. 1. 1. 1. 0. 1. 1. 0. 1. 1. 0.]
```

- 총 36개의 훈련 세트 = 도미(1) 25개 + 빙어(0) 11개
- 잘 섞인 것 같지만 빙어의 비율이 모자란다.

- 원래) 도미:빙어 = 35:14 = **2.5:1**
- 훈련) 도미:빙어 = 25:11 = **2.3:1**

- 총 13개의 테스트 세트 = 도미(1) 10개 + 빙어(0) 3개
- 잘 섞인 것 같지만 빙어의 비율이 모자란다.

- 원래) 도미:빙어 = 35:14 = **2.5:1**
- 테스트) 도미:빙어 = 10:3 = **3.3:1**

샘플링 편향이 조금 나타났다.

훈련 세트와 테스트 세트에 샘플의 클래스 비율이 일정하지 않다.

- 샘플링 편향이 발생하진 않았지만, 테스트 세트에서의 샘플의 클래스 비율과 차이가 난다. (테스트 세트 3.3:1) -> 모델이 일부 샘플을 올바르게 학습 x

4

stratify 매개변수

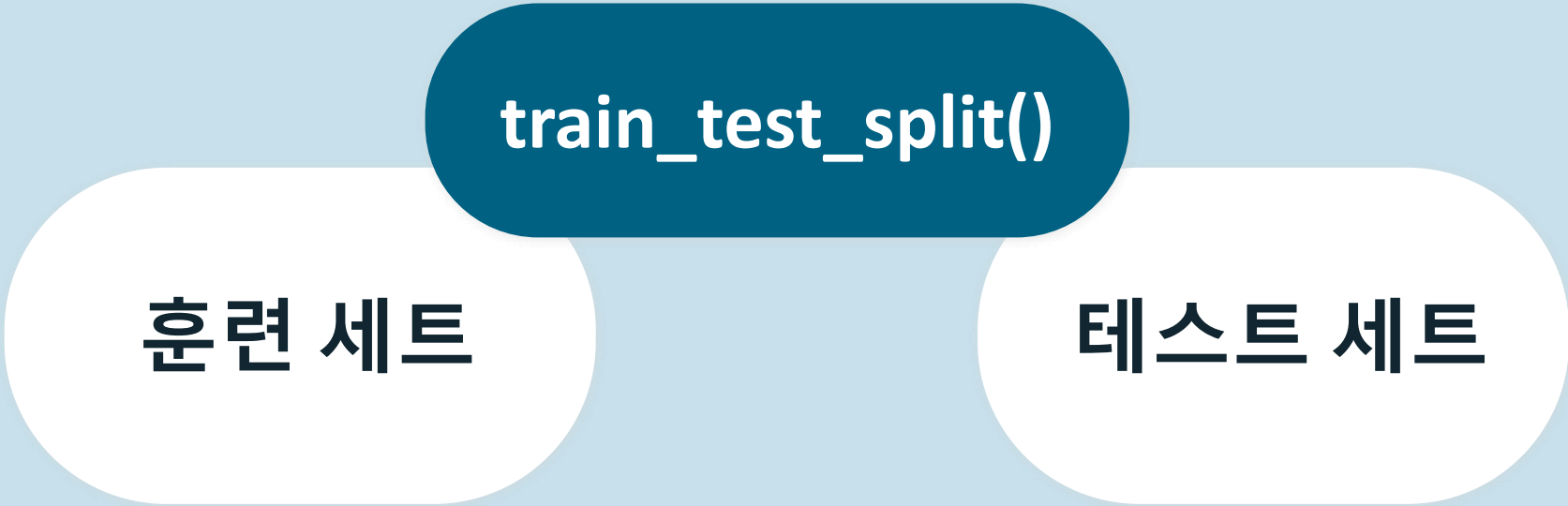
네번째 생선 분류 ML 모델

train_test_split()

전달되는 리스트나 배열(훈련 데이터)을 비율에 맞게 훈련 세트와 테스트 세트로 나누어준다.

stratify 매개변수

- 클래스 레이블이 담긴 배열(일반적으로 타깃 데이터)을 전달하면 클래스 비율에 맞게 훈련 세트와 테스트 세트를 나눈다.
- 훈련 데이터가 작을 때 or 특정 클래스의 샘플 개수가 적을 때 유용하다.



1st) 훈련 데이터 만들기

numpy 이용

2nd) 훈련 세트 & 테스트 세트 나누기

stratify 매개변수로 클래스 비율 맞추기

```
# stratify 매개변수를 도입하여 훈련 세트와 테스트 세트 나누기
train_input, test_input, train_target, test_target = train_test_split(fish_data, fish_target, stratify=fish_target, random_sta
```

3rd) 각 클래스(도미와 빙어)가 잘 섞여있는지 확인하기

```
# 도미와 빙어 잘 섞였는지 확인하기
print(test_target)
print(train_target)
```

```
[0. 0. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1.]
[1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1.
 0. 1. 0. 1. 1. 1. 0. 1. 1. 0. 1. 1.]
```

- 원래) 도미:빙어 = 35:14 = **2.5:1**
- 테스트) 도미:빙어 = 9:4 = **2.25:1**
- 훈련) 도미:빙어 = 26:10 = **2.6:1**

데이터가 작아 전체 훈련 데이터의 비율과 테스트의 비율을 동일하게 맞출 수 없지만 꽤 비슷한 비율이다.

4th) 훈련 세트와 테스트 세트로 K-NN 모델 훈련하기

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

```
1.0
```

정확도 : 1.0 -> 테스트 세트의 도미와 빙어를 모두 올바르게 분류했다.

5th) 훈련할 때 들어 있지 않은 샘플로 테스트하기

도미 데이터 [25,150] 넣고 결과 확인해보기 -> 1이 나와야 한다.

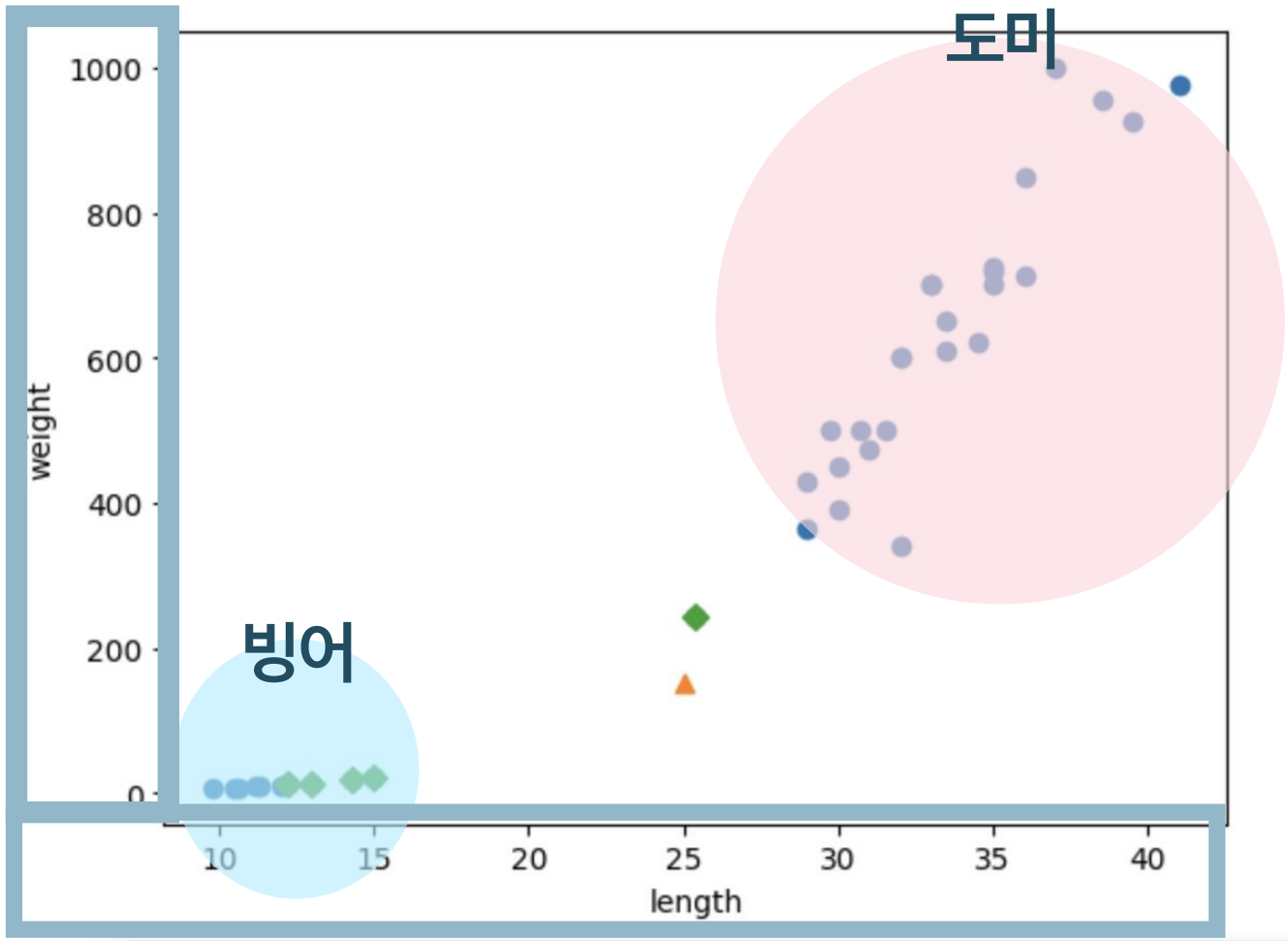
```
kn.predict([[25, 150]])
```

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

샘플의 예측값이 틀렸다. **WHY?**

1이 나와야 하는데 0이 나왔다. -> *문제 발생!!*

WHY? 문제 발생 원인



Q.
산점도를 보면
직관적으로 도미와 가깝게 보이는데
왜 가까운 이웃에 빙어가 많은가?

A.
길이와 무게의 값이 놓인 범위가
매우 다르다.

두 특성의 스케일이 다르다.

모델이 빙어 데이터에 가깝다고 판단한 이유 :
가장 가까운 이웃에 도미는 하나 뿐이고, 나머지 4개의 샘플이 모두 빙어이다.

5

데이터 전처리 – 표준점수

다섯번째 생선 분류 ML 모델

데이터 전처리 (data preprocessing)

거리 기반 알고리즘은 샘플 간의 거리의 영향을 많이 받기에 특성 값을 일정한 기준으로 맞춰줘야 한다.

표준점수

- 훈련 세트의 스케일을 바꾸는 대표적인 방법
- 계산 방법) 원본 데이터에서 특성의 평균을 빼고 표준편차로 나눈다.

  -> 마치 표준 정규 분포의 z값을 구하는 느낌이다!

1st) 각 특성의 평균(mean), 표준편차(std) 구하기

numpy 이용

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

```
# 각 특성의 평균, 표준편차 구하기
print(mean, std)
```

```
[ 27.29722222 454.09722222] [  9.98244253 323.29893931]
```

2nd) 표준점수 구하기

브로드캐스팅(broadcasting = 크기가 다른 넘파이 배열에서 자동으로 사칙 연산을 모든 행이나 열로 확장하여 수행하는 기능

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

전처리 데이터로 모델을 훈련할 때
테스트 세트와 샘플도 훈련 세트와 동일한 비율로 변환해야 한다.

3rd) 전처리 데이터(train_scaled)로 K-NN 모델 훈련하기

```
kn.fit(train_scaled, train_target)
```

4th) 변환된 테스트 세트로 모델 평가하기

```
# 테스트 세트 변환하기
test_scaled = (test_input - mean) / std
# 모델 평가하기
kn.score(test_scaled, test_target)
```

```
1.0
```

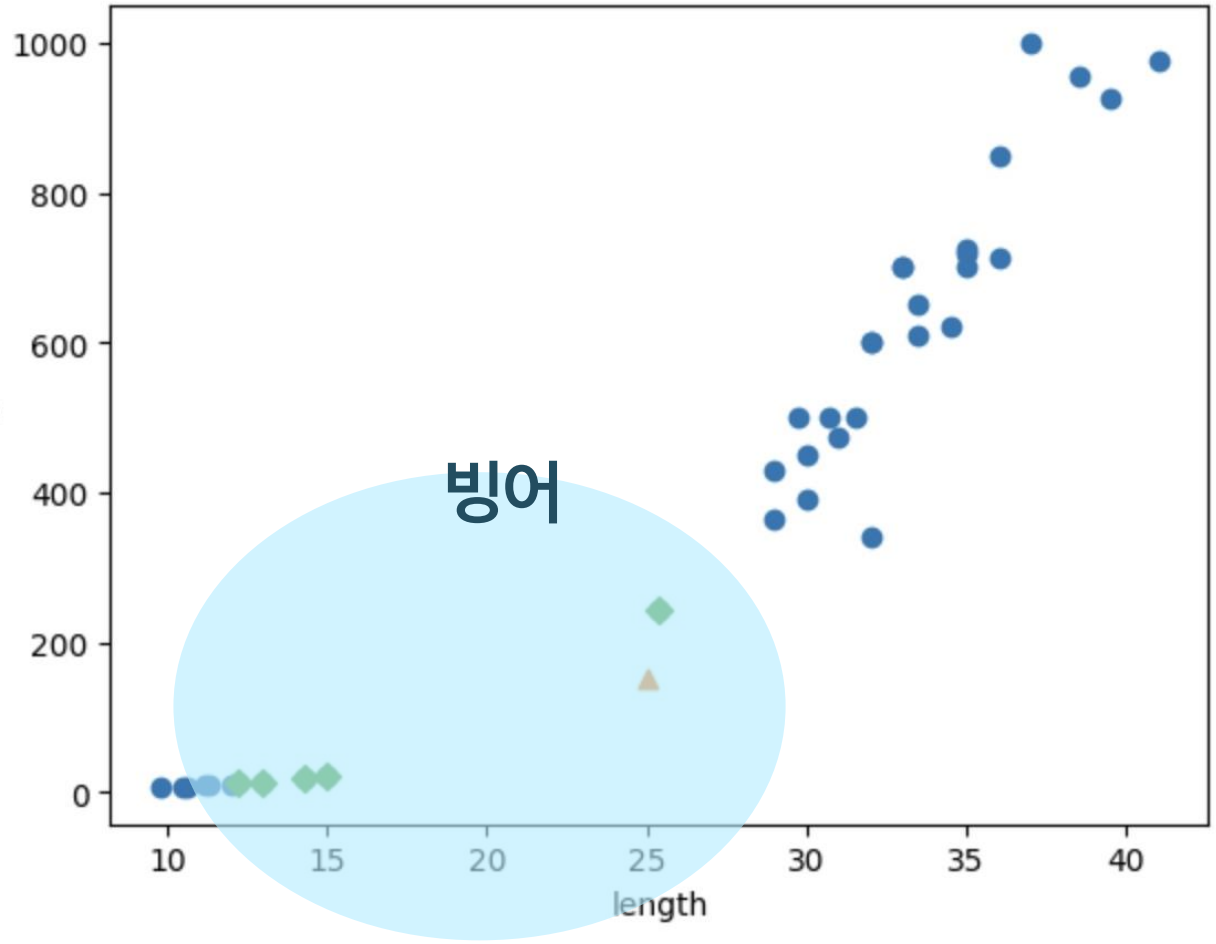
5th) 훈련될 때 들어있지 않은 샘플로 테스트하기

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

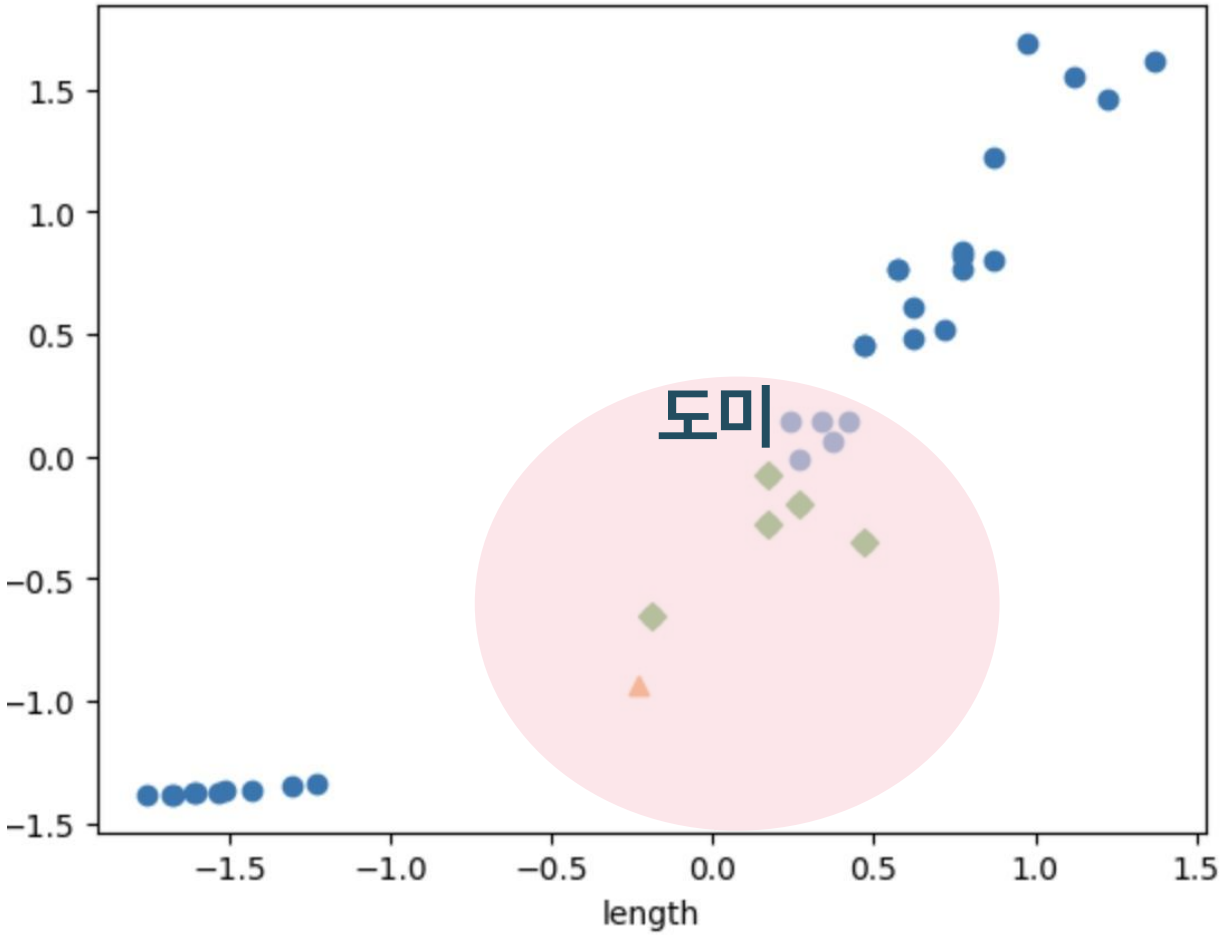
```
[1.]
```

샘플(new)의 K-NN 산점도로 나타내기

이전 모델



표준점수로 변환한 모델



Thank You