데이터 전처리

우도경
2025-07-09

1. 문제 상황

- Model Code

```
[ ]  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.]
```

- 테스트 데이터에 대한 분류 정확도 = 100%
- 모델이 완벽하게 테스트셋을 예측

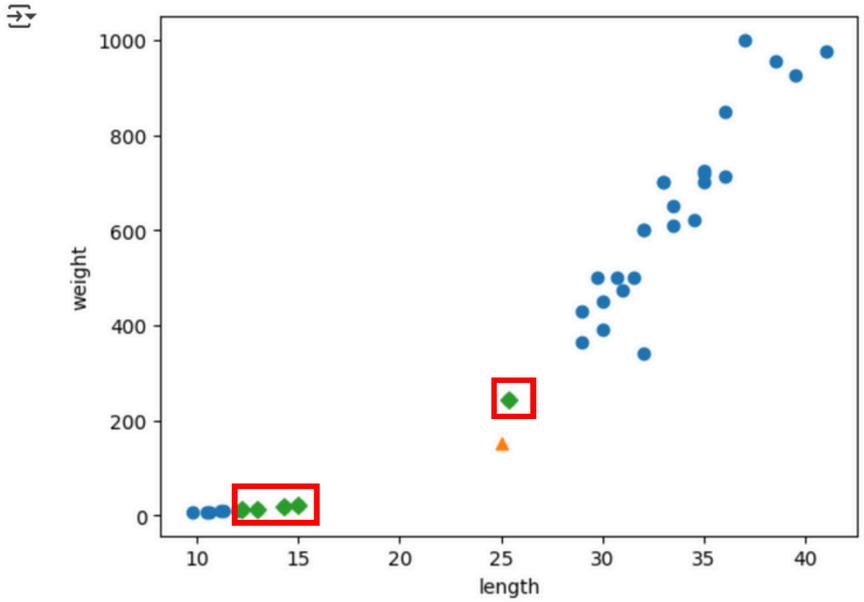
- 그러나, 실제 도미(1) 데이터를 넣었을 때, 모델은 빙어(0) 데이터로 예측

1. 문제 상황

- 산점도

```
[ ]  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()
```

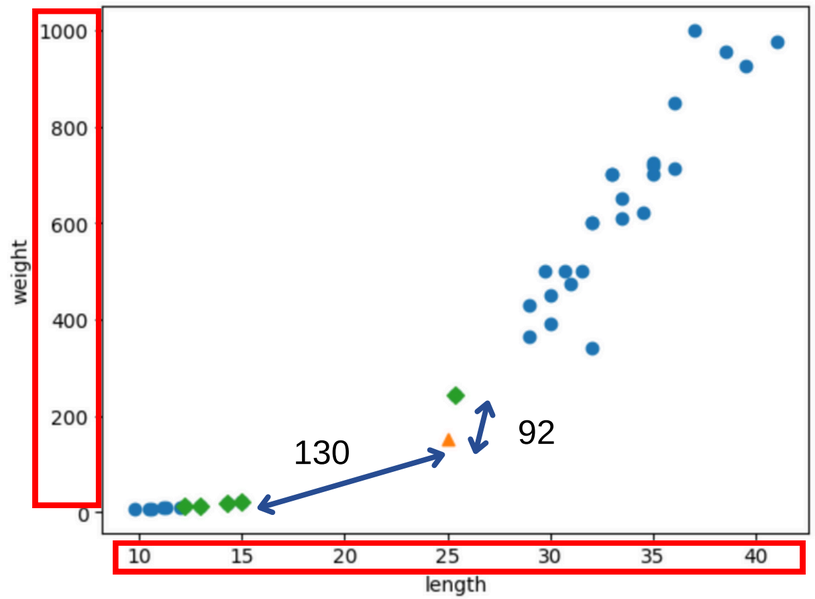


- 새 도미 데이터는 도미 군집에 가까워 보이지만,
- 모델은 빙어 데이터 4개를 2~5순위로 예측함.

2. 원인 분석

- 거리 계산

```
[25] print(train_input[indexes])

[[[ 25.4 242. ]
  [ 15.   19.9]
  [ 14.3  19.7]
  [ 13.   12.2]
  [ 12.2  12.2]]]

[26] print(train_target[indexes])

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

[27] print(distances)

[[ 92.00086956 130.48375378 130.73859415 138.32150953 138.39320793]]
```

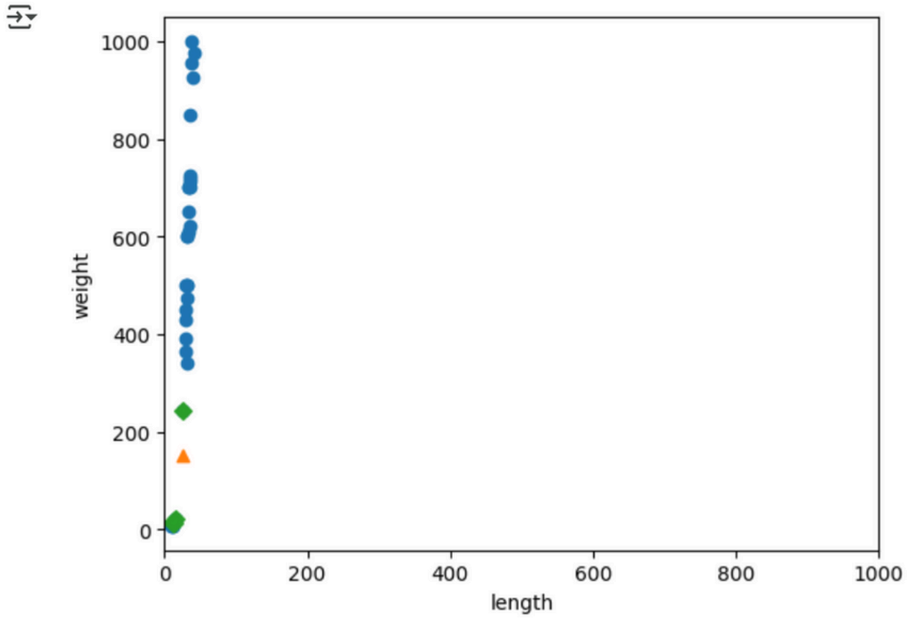


- x, y 축 스케일 차이로 인한 거리 왜곡

2. 원인 분석

- x,y 축의 범위 통일

```
[28] 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()
```



- y 값의 변화량에 비해 x 값의 변화량은 거의 변화가 없음
  → 가장 가까운 이웃을 찾을 때, 길이보다 무게의 크기에 따라 예측 결과가 왜곡됨

3. 데이터 전처리

- 앞선 경우와 같이 두 특성(길이와 무게)의 값이 놓인 범위가 매우 다를 때, 이를 두 특성의 스케일 불일치라고 표현함
- 특성의 단위 또는 범위가 다르면 알고리즘이 올바르게 예측할 수 없음.
- 특히 k-최근접 이웃은 샘플 간의 거리에 영향을 많이 받으므로 특성값을 일정한 기준으로 맞춰줘야 함.
- 전처리 방법 중 하나로, 표준점수(Standard Score)가 자주 사용됨

3. 데이터 전처리

- 표준점수
- 어떤 값이 전체 평균에서 얼마나 떨어져 있는지를 “표준편차 단위“로 나타낸 값
- 공식

$$z = \frac{x - \mu}{\sigma}$$

- x: 데이터 값
- μ: 평균
- σ: 표준편차
- z: 표준점수

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

```
[ ]  print(mean, std)
```

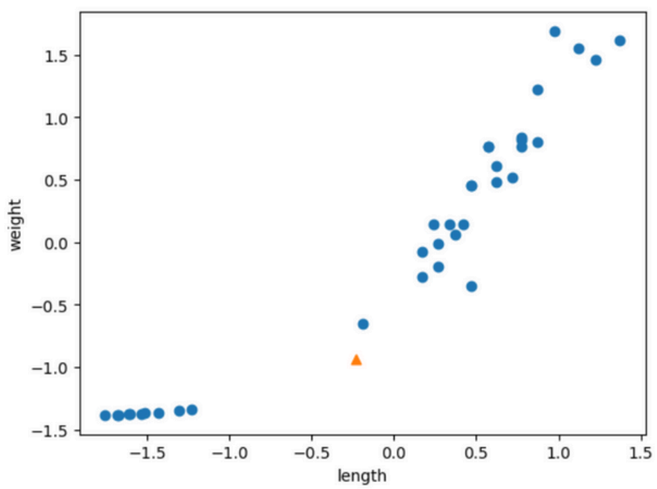
```
[ 27.29722222 454.09722222] [  9.98244253 323.29893931]
```

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

4. 결과

- 전처리된 데이터를 기반으로 모델 학습 수행

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




```
[ ]  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.]
```

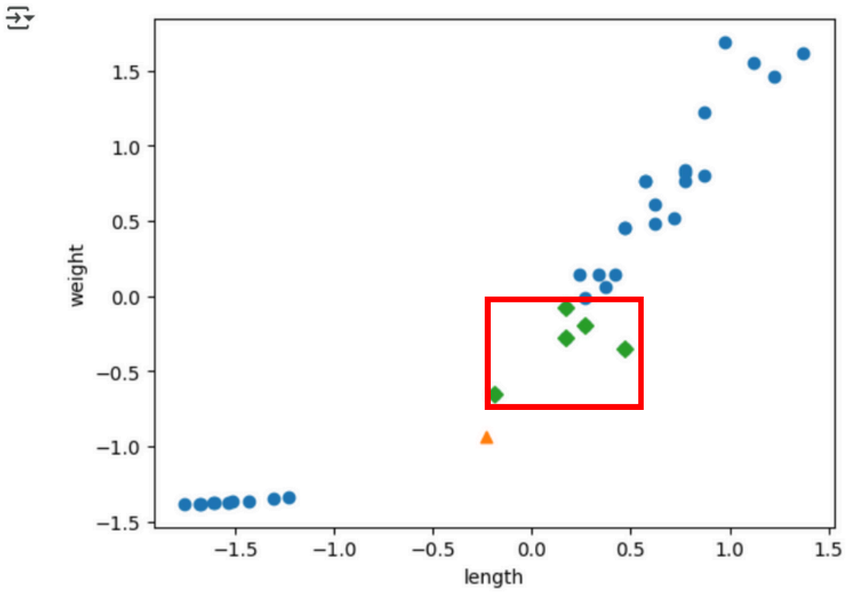
- 훈련 세트와 동일한 방식으로 테스트 세트도 변환해 일관성 유지

4. 결과

- 이웃 확인

```
[ ]  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()
```



- 전처리 이후, 도미 데이터의 이웃들이 모두 도미로 예측됨
- 거리 기준이 정상화되면서 정확도 향상 확인

5. 결론

- 단위 차이가 큰 특성은 전처리 없이 사용 시 예측 오류 발생
- 표준점수 기반 스케일 조정으로 정확도 향상
- 전처리는 모델 성능을 좌우하는 핵심 요소

Q&A

Thank You