데이터 전처리

우도경
2025-07-09

INDEX

- 문제 상황
- 원인 분석
- 스케일링
- 모델링 및 예측/평가
- 결론

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.]
```

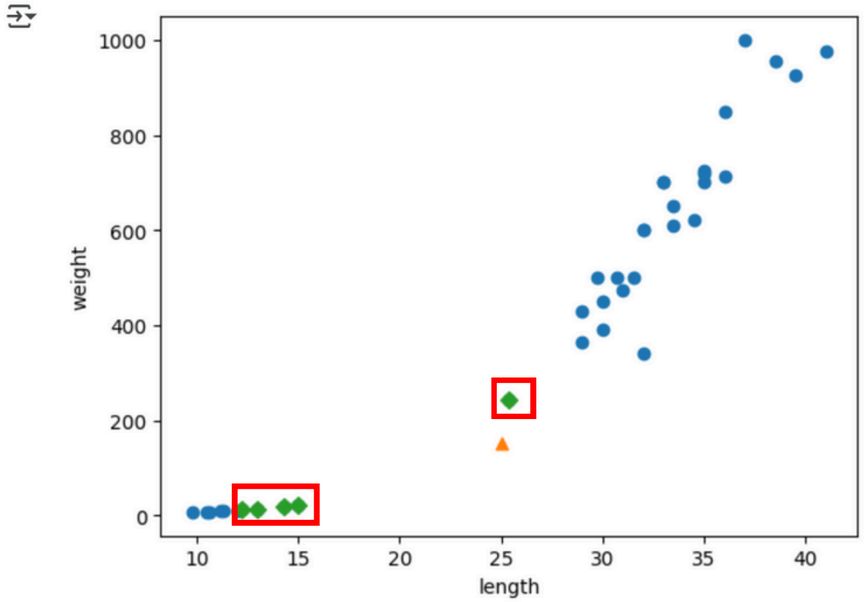
- 테스트 데이터에 대한 분류 정확도 = 100%
- 모델이 완벽하게 테스트셋을 예측
- 그러나, 실제 도미(1) 데이터를 넣었을 때, 모델은 빙어(0) 데이터로 예측

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 데이터(▲)는 도미 군집에 가까워 보임.
- 그러나, 모델은 빙어 데이터 4개를 2~5순위로 예측함.

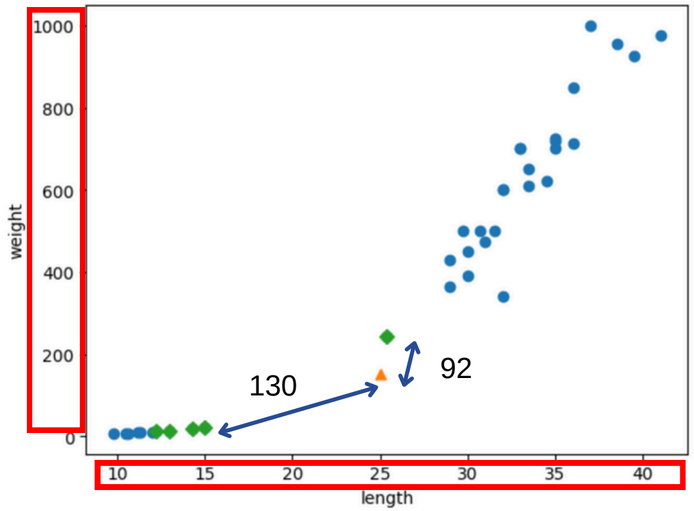
2. 원인 분석

- 거리 확인

```
[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.]]
```

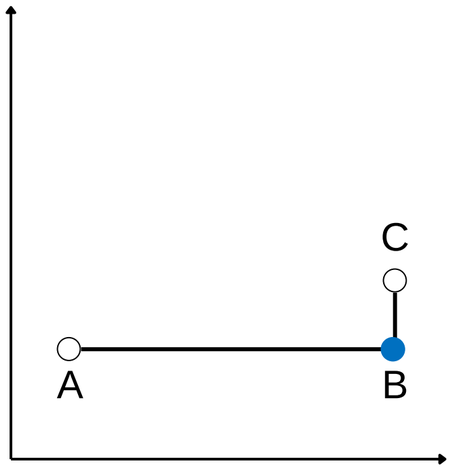
```
[27] print(distances)

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

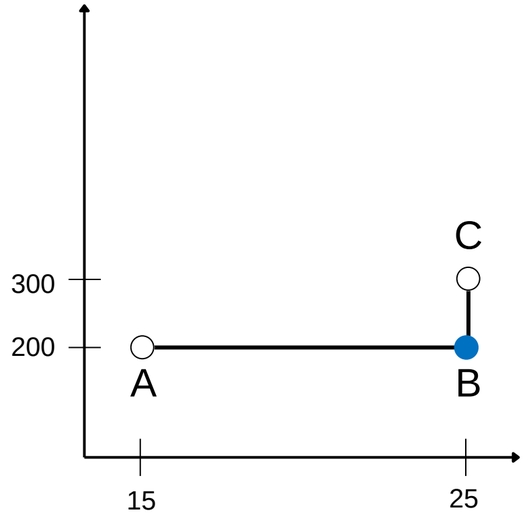


- x, y 축 스케일 차이로 인한 거리 왜곡

2. 원인 분석

- 변수의 스케일



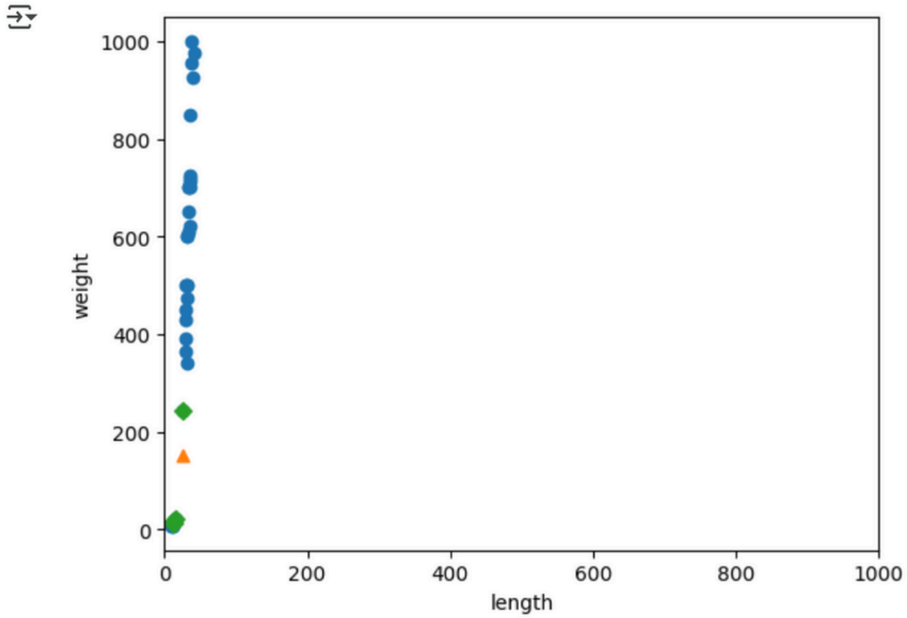


- 왼쪽 그래프의 점 B 기준으로 B에서 거리가 가까운 점은 C
- 오른쪽 그래프의 점 B 기준으로 B에서 거리가 가까운 점은 A
- 거리를 계산해서 모델을 작성하는 알고리즘 사용 시, 변수의 스케일을 맞추어 놓는 것이 필수

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. 스케일링

- 스케일링은 데이터의 스케일(scale)을 맞추는 작업
- lenght에서의 1과 weight에서의 1은 완전히 다른 영향을 미침
- K-최근접 이웃은 거리 기반의 알고리즘이므로, 왜곡된 예측을 할 수 있음
- 스케일링은 이러한 문제를 해결할 목적으로 인위적으로 각 변수가 비슷한 범위를 가지도록 만듦

3. 스케일링

- 표준화 스케일링 : 데이터를 표준화된 정규분포로 만들어주는 방법
- 각 변수를 기준으로 하여, 각 데이터에 해당 변수의 평균을 빼주고 이를 표준편차로 나눔
- 데이터를 표준화하면 평균은 0이고 분산은 1인 형태로 데이터가 변경됨

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

- 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)
```

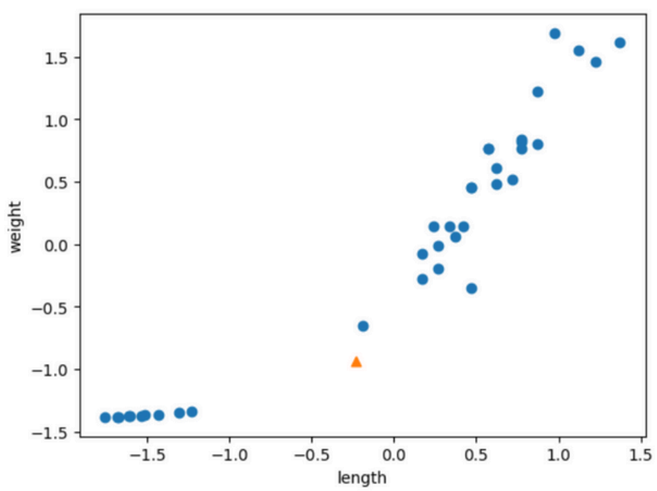
```
[ 27.29722222 454.09722222] [  9.98244253 323.29893931]
```

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

4. 모델링 및 예측/평가

- 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.]
```

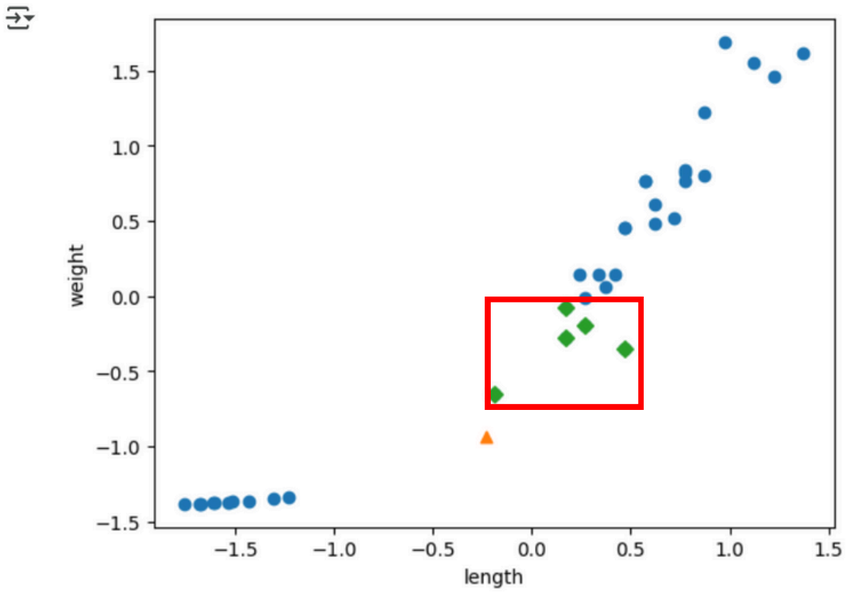
- 훈련 세트와 동일한 방식으로 테스트 세트도 변환해 일관성 유지

4. 모델링 및 예측/평가

- 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

5. 결론

- KNN과 같은 거리 기반 모델에서는 변수의 스케일이 중요하게 작용함.
- 스케일링을 통해 변수들의 스케일을 맞춰주는 과정이 필수적

Q&A

Thank You

참고문헌

- 아다치 하루카, 김태현(역), 조휘용(감수). 백견불여일타 머신러닝 데이터 전처리 입문. 로드북, 2020.
- 권시현(데싸노트). Must Have 데싸노트의 실전에서 통하는 머신러닝. 골든래빗, 2022.