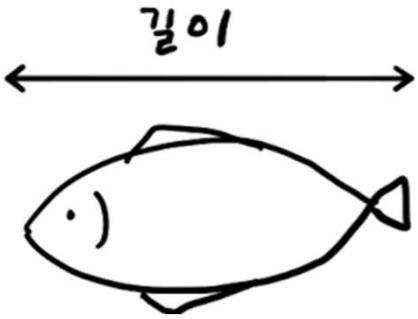
농어의 무게를 예측하라



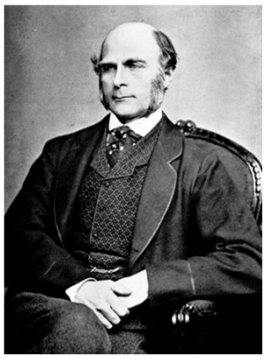
1

회귀 Regression

- 지도학습은 크게 분류와 회귀(Regression)로 나눔
- **분류** : 샘플을 몇 개의 클래스 중 하나로 분류하는 문제
- **회귀** : 클래스 중 하나로 분류하는 것이 아닌 임의의 어떤 숫자를 예측하는 문제임
  - 내년 경제 성장률 예측
  - 배달 도착시간 예측
  - 농어의 길이로 무게를 예측
- 회귀는 정해진 클래스가 없고 임의의 수치를 출력

2

회귀 Regression

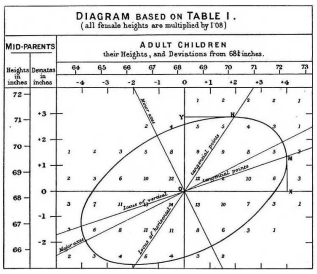


Francis Galton

여러 자료들 간의 관계성을 수학적으로 추정, 설명

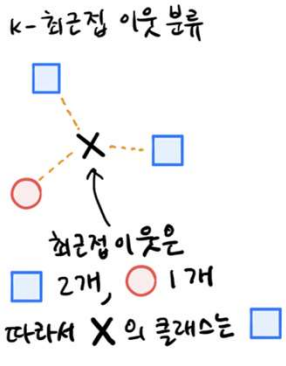
회귀분석은 인과관계를 증명하는 방법이 아니라,
인과관계가 상정된 모델을 구현할 수 있는 것에 불과

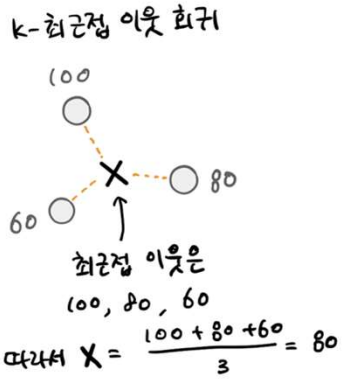
단순회귀분석(Simple Regression Analysis)
다중회귀분석(Multiple Regression Analysis)



3

k-최근접 이웃 회귀





K-최근접 이웃 회귀 모델은 분류와 동일하게 가장 가까운 k개의 이웃을 찾음
그 다음 이웃 샘플의 타깃 값을 평균하여 이 샘플 예측 값으로 사용함

4

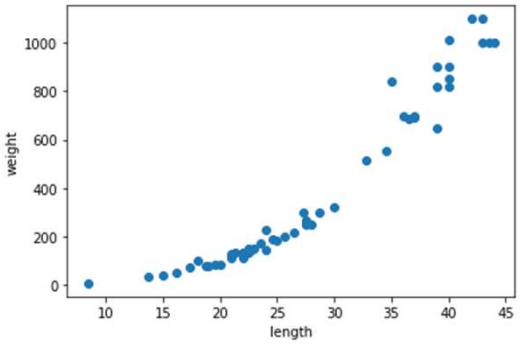
데이터 준비

```
perch_length = np.array(
       [8.4, 13.7, 15.0, 16.2, 17.4, 18.0, 18.7, 19.0, 19.6, 20.0, 21.0,
       21.0, 21.0, 21.3, 22.0, 22.0, 22.0, 22.0, 22.0, 22.5, 22.5, 22.7,
       23.0, 23.5, 24.0, 24.0, 24.6, 25.0, 25.6, 26.5, 27.3, 27.5, 27.5,
       27.5, 28.0, 28.7, 30.0, 32.8, 34.5, 35.0, 36.5, 36.0, 37.0, 37.0,
       39.0, 39.0, 39.0, 40.0, 40.0, 40.0, 40.0, 42.0, 43.0, 43.0, 43.5,
       44.0])

perch_weight = np.array(
      [5.9, 32.0, 40.0, 51.5, 70.0, 100.0, 78.0, 80.0, 85.0, 85.0, 110.0,
       115.0, 125.0, 130.0, 120.0, 120.0, 130.0, 135.0, 110.0, 130.0,
       150.0, 145.0, 150.0, 170.0, 225.0, 145.0, 188.0, 180.0, 197.0,
       218.0, 300.0, 260.0, 265.0, 250.0, 250.0, 300.0, 320.0, 514.0,
       556.0, 840.0, 685.0, 700.0, 700.0, 690.0, 900.0, 650.0, 820.0,
       850.0, 900.0, 1015.0, 820.0, 1100.0, 1000.0, 1100.0, 1000.0,
       1000.0])
```

5

농어의 길이만 사용

```
import matplotlib.pyplot as plt

plt.scatter(perch_length, perch_weight)
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
```



6

훈련 세트 준비

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(
    perch_length, perch_weight, random_state=42)

train_input = train_input.reshape(-1, 1) #2차원 배열 변환
test_input = test_input.reshape(-1, 1)
```

[1, 2, 3] → [[1], [2], [3]]

크기: (3,) → 크기: (3, 1)

7

회귀 모델 훈련

```
from sklearn.neighbors import KNeighborsRegressor

knr = KNeighborsRegressor()
knr.fit(train_input, train_target)

knr.score(test_input, test_target)
0.9928094061010639
```

위의 결과가 좋은 값이지만, 정확도처럼 R^2가 직감적으로 얼마나 좋은지 이해하기는 어려움

8

결정계수 R^2

- 회귀에서는 결정계수 coefficient of determination 로 평가

$$R^2 = 1 - \frac{(\text{타깃} - \text{예측})^2\text{의 합}}{(\text{타깃} - \text{평균})^2\text{의 합}}$$

  - 각 샘플의 **타깃**과 **예측**한 값의 **차이**를 **제곱**해 더함
  - 타깃과 타깃 평균의 차이를 제공하여 더한 값으로 나눔
- 만약 **타깃의 평균 정도가 예측하는 수준**(즉 분자와 분모가 비슷함)이라면 R^2는 0에 가까워짐
- **예측이 타깃에 아주 가까워 지면** 1에 가까운 값이 됨

9

회귀 모델 훈련

사이킷런에서 제공하는 측정도구 중 mean_absolute_error 이용해
타깃과 예측 값의 오차를 평균으로 반환 해봄

```
from sklearn.metrics import mean_absolute_error

test_prediction = knr.predict(test_input)
mae = mean_absolute_error(test_target, test_prediction)
print(mae)
19.157142857142862
```

결과에서 예측이 평균적으로 19g 정도 타깃 값과 다르다는 것을 알 수 있음

10

과대적합과 과소적합

```
knr.score(train_input, train_target)
0.9698823289099255

knr.score(test_input, test_target)
0.9928094061010639
```

11

과대적합과 과소적합

- **과대적합(overfitting)**
  - 결과는 훈련 세트에서 점수가 좋지만,
    테스트 점수에서 점수가 좋지 못함

- 훈련세트에만 잘 맞는 모델이라
  테스트 세트와 나중에 실전에 투입해서 새로운 샘플에 대한
  예측을 만들 때 동작하지 않을 것

12

과대적합과 과소적합

- **과소 적합(underfitting)**
  - 훈련세트 보다 테스트 세트 점수가 높거나
  - 두 점수가 모두 낮은 경우
- 모델이 너무 단순해서 데이터를 대표한다고 가정하기 때문에 훈련 세트를 잘 학습하는 것이 중요

13

과대적합과 과소적합

```
knr.score(train_input, train_target)
0.9698823289099255

knr.score(test_input, test_target)
0.9928094061010639
```

과소 적합 상태(데이터가 너무 작음)

그럼 문제를 어떻게 해결 할 것인가?

14

여기서 과소 적합 해결 방법!

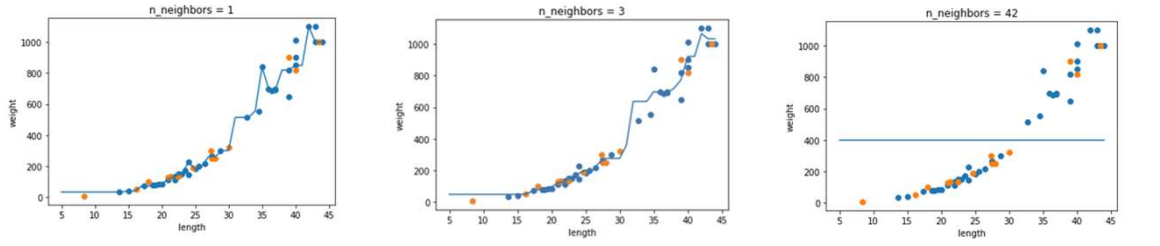
- 모델을 더 복잡하게 하면 됨
- 여기선(K-최근접 이웃 회귀 알고리즘) 검색하는 이웃의 개수를 줄임
- 이웃을 개수를 줄여 훈련세트에 있는 국지적인 패턴에 민감해짐
- 이웃의 개수를 늘리면 데이터 전반에 일반적인 패턴을 따름
- 기본은 5개, 여기선 3개로 줄임

15

이웃 개수 줄이기

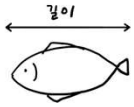
```
knr.n_neighbors = 3
knr.fit(train_input, train_target)

print(knr.score(train_input, train_target))
0.9804899950518966

print(knr.score(test_input, test_target))
0.974645996398761
```

과대적합 ↓ 이웃의 개수 ↑ 과소적합



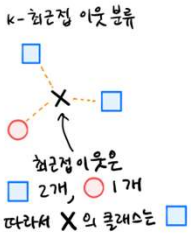
16

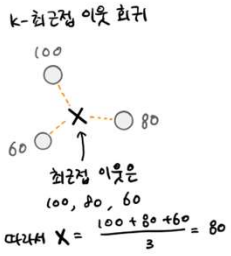
정리



무게

$$R^2 = 1 - \frac{(\text{타깃} - \text{예측})^2\text{의 합}}{(\text{타깃} - \text{평균})^2\text{의 합}}$$






```
knr.score(train_input, train_target)
0.9698823289099255

knr.score(test_input, test_target)
0.9928094061010639
```

과대적합 ↓ 이웃의 개수 ↑ 과소적합

17

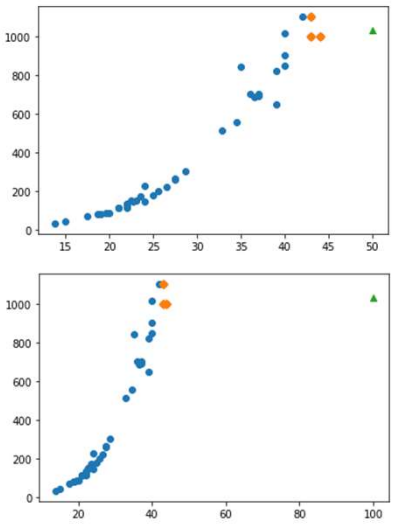
아주 큰 농어


```
print(knr.predict([[50]]))
[1033.33333333]
```

18

50cm 농어의 이웃

```
# 50cm 농어의 이웃을 구합니다
distances, indexes = knr.kneighbors([[50]])

# 훈련 세트의 산점도를 그립니다
plt.scatter(train_input, train_target)
# 훈련 세트 중에서 이웃 샘플만 다시 그립니다
plt.scatter(train_input[indexes], train_target[indexes],
            marker='D')
# 50cm 농어 데이터
plt.scatter(50, 1033, marker='^')
plt.show()
```



19

감사합니다

내용 출처 정보 : https://www.hanbit.co.kr/store/books/look.php?p_code=B2002963743

20