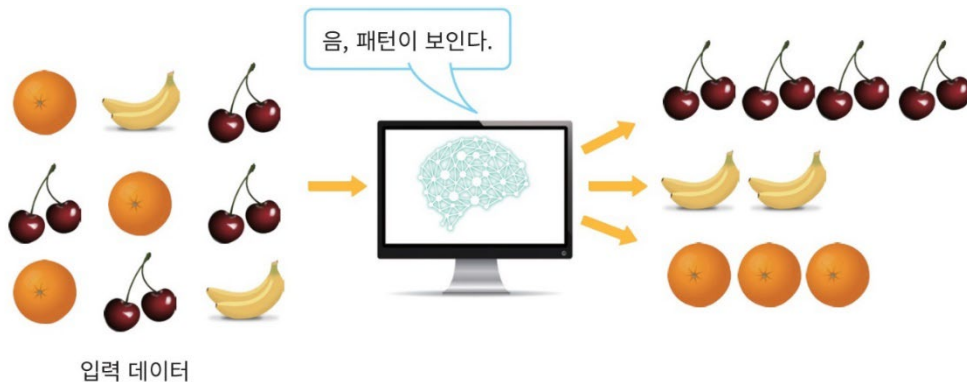


## ◆ 비지도 학습(Unsupervised Learning)

- 선생님없이 컴퓨터(기계)가 스스로 입력들을 분류하는 것을 의미.
- 레이블( $y$ )이 주어지지 않는 기계 학습.
- 방법: 기계가 입력 값들의 상관도를 분석하여 유사한 데이터들을 스스로 모으고 학습.

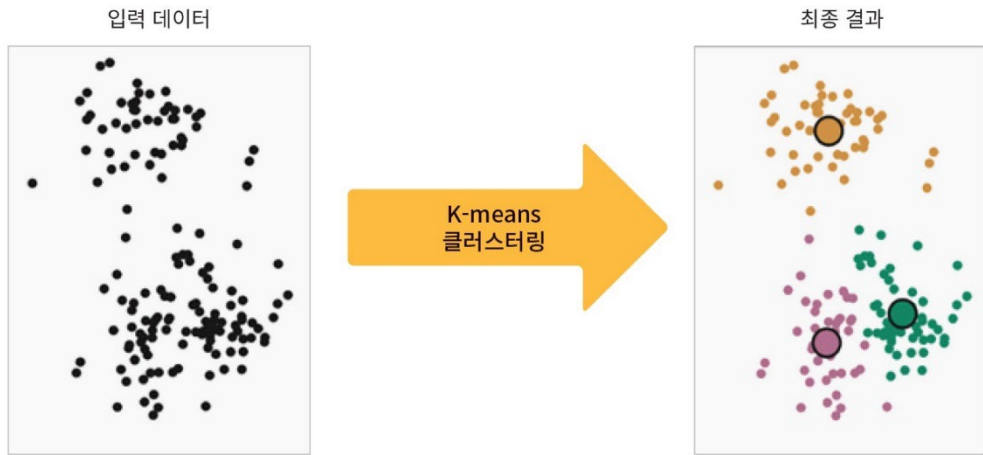


지도 학습: 체리, 바나나, 오렌지를 미리 알려줌. → 과일 이름을 알게 됨.

비지도 학습: 과일 이름을 알려주지 않음. → 과일 이름은 모르나 비슷한 것을 그냥 모으는 것.

## ◆ 비지도 학습(Unsupervised Learning)

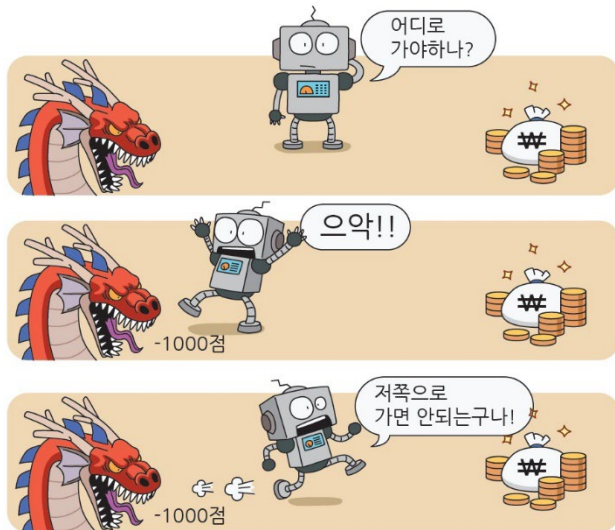
- 클러스터링(군집화, Clustering) 기법을 이용.
- 클러스터링이란 학습 데이터 간 거리를 계산하여 몇 개의 그룹으로 나누는 방식
- 대표적인 클러스터링 알고리즘: k-Means



입력 데이터를 3개의 카테고리로 분류

## ◆ 강화 학습(Reinforcement Learning)

- 컴퓨터(기계)가 어떤 행동을 할 때마다 외부에서 처벌이나 보상을 주는 것.
- 컴퓨터(기계)는 처벌을 피하고 보상을 많이 얻을 수 있는 전략을 스스로 터득.



속담(격언)

매가 약이다.

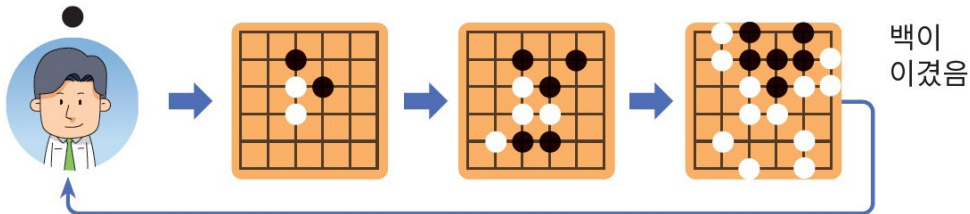
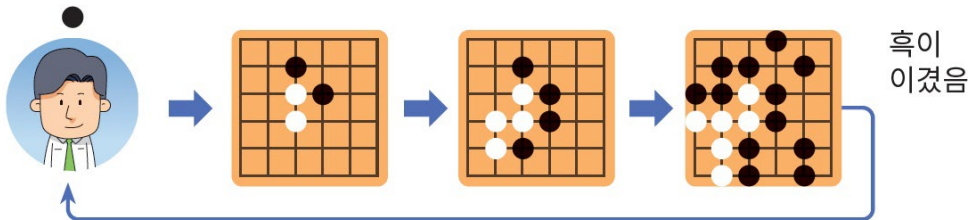
칭찬은 고래도 춤추게 한다.

즉, 당근과 채찍



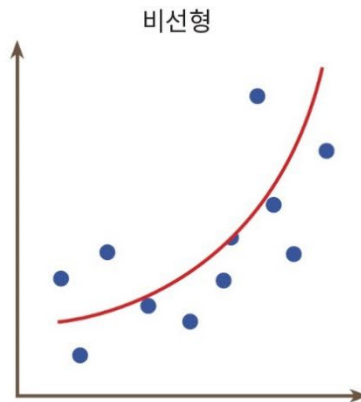
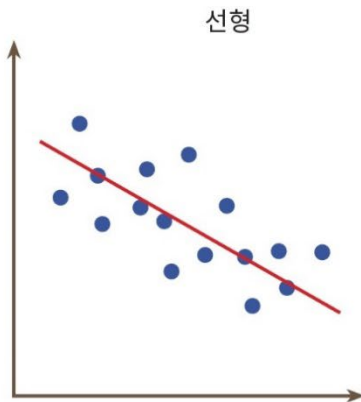
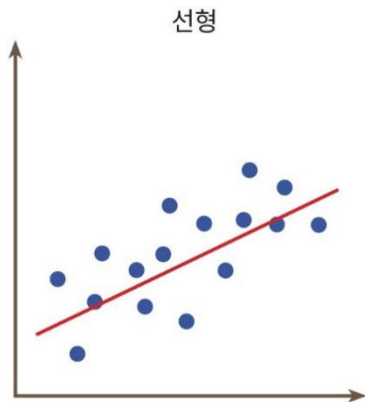
## ◆ 강화 학습(Reinforcement Learning)

- 알파고 최종 버전도 강화 학습 사용



## ◆ 회귀(回歸, Regression)

- 회귀라는 단어 유래: 중심(평균)으로 돌아간다는 통계학 용어에서 유래함.
- 수 많은 데이터들을 가장 잘 대표하는 함수를 찾는 것을 의미.
- 이 중에서 선형 함수(직선)를 찾는 회귀를 선형 회귀라고 함.



# 선형 회귀

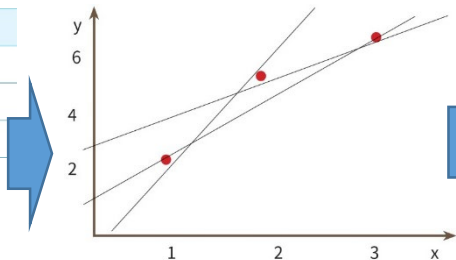
## ◆ 선형 회귀 기본식

- $f(x) = Wx + b$
- $x$  : 데이터(샘플)
- $W$  : 기울기(가중치)
- $b$  : 절편(바이어스)

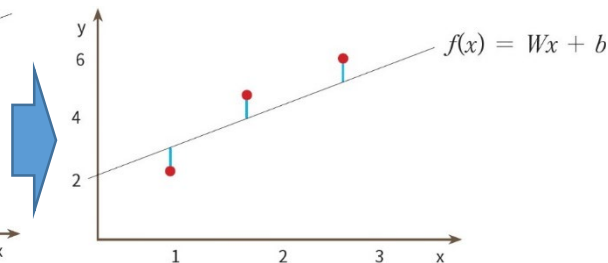
◆ 선형 회귀는 입력 데이터( $x$ )를 가장 설명하는 기울기( $W$ )와 절편( $b$ )을 찾는 것.

| x | y |
|---|---|
| 1 | 2 |
| 2 | 5 |
| 3 | 6 |

데이터



데이터와 후보 함수



선형 회귀 결과

## ◆ 손실 함수(Loss Function)

- 직선과 데이터 사이의 간격을 제공하여 합한 값.
- 비용 함수(Cost Function)라고도 함.

$$Loss = \frac{1}{3} ((f(x_1) - y_1)^2 + (f(x_2) - y_2)^2 + (f(x_3) - y_3)^2)$$



일반화

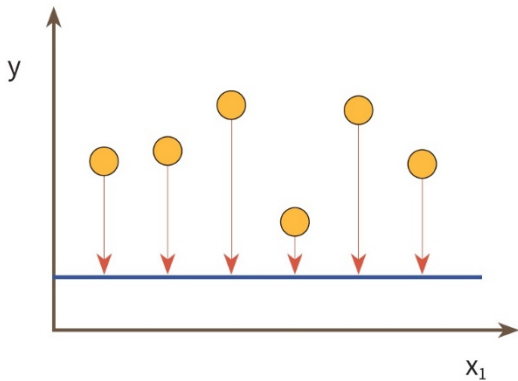
$$Loss(W, b) = \frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2 = \frac{1}{n} \sum_{i=1}^n ((Wx_i + b) - y_i)^2$$



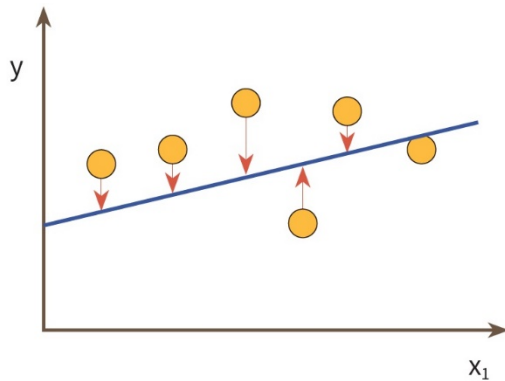
손실 함수 값이 최소가 되는  $W, b$

$$\underset{W, b}{\operatorname{argmin}} Loss(W, b)$$

## ◆ 손실 함수(Loss Function)



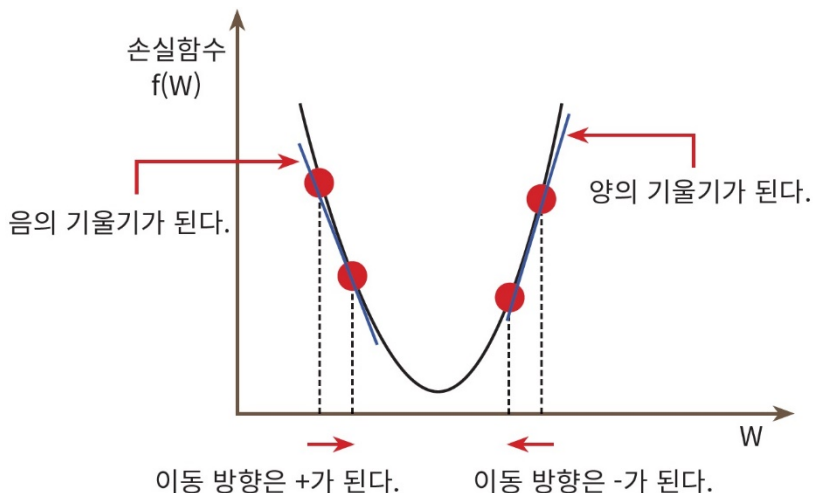
손실 함수 값이 크다.



손실 함수 값이 작다.

## ◆ 손실 함수 값을 최소화하는 방법

- 경사 하강법(Gradient Descent Method)
- 함수를 변수에 대해 미분했을 때 기울기가 음수이면 변수 값을 증가 → 손실 함수 값 감소
- 함수를 변수에 대해 미분했을 때 기울기가 양수이면 변수 값을 감소 → 손실 함수 값 감소



## ◆ 경사 하강법

$$Loss(W, b) = \frac{1}{n} \sum_{i=1}^n ((Wx_i + b) - y_i)^2$$

$W$ 에 대해 미분:  $\frac{\partial Loss(W, b)}{\partial W} = \frac{1}{n} \sum_{i=1}^n 2((Wx_i + b) - y_i)(x_i) = \frac{2}{n} \sum_{i=1}^n x_i((Wx_i + b) - y_i)$

$b$ 에 대해 미분:  $\frac{\partial Loss(W, b)}{\partial b} = \frac{2}{n} \sum_{i=1}^n ((Wx_i + b) - y_i)$

나중 값    현재 값    학습률    미분 값

$$W = W - 0.01 * \frac{\partial Loss}{\partial W}$$

$$b = b - 0.01 * \frac{\partial Loss}{\partial b}$$

미분 값이 양수이면 감소.  
미분 값이 음수이면 증가.

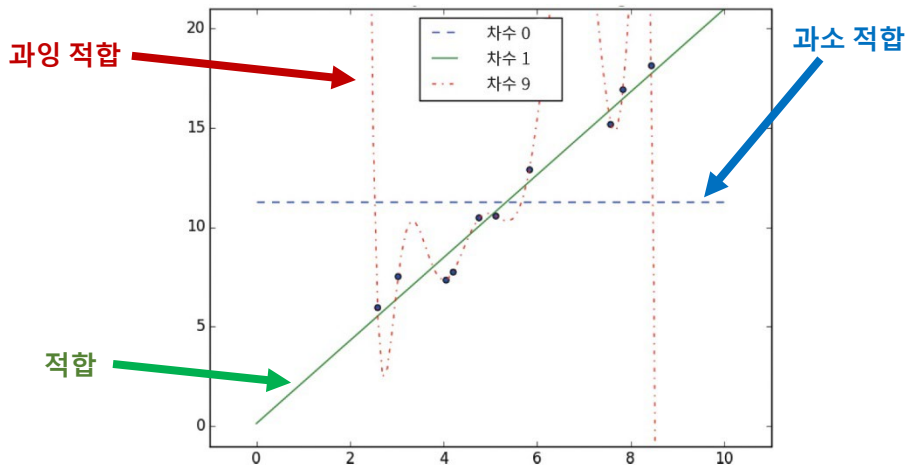
# 과잉 적합과 과소 적합

## ◆ 과잉 적합(Overfitting)

- 학습하는 데이터에서는 성능이 뛰어나지만 새로운 데이터(일반화)에 대해서는 성능이 잘 나오지 않는 모델.

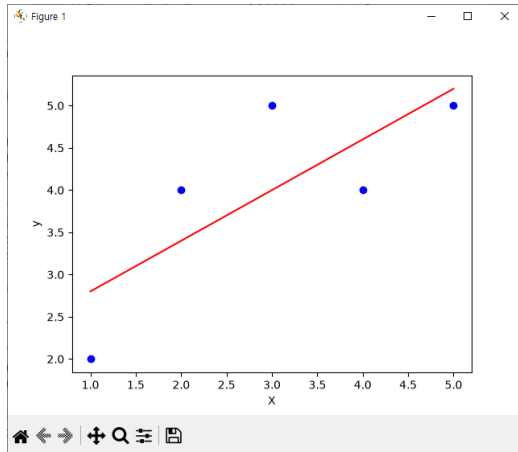
## ◆ 과소 적합(Underfitting)

- 모델이 적합하지 않아 학습 데이터에서도 성능이 좋지 않은 모델.



# 인공지능 코딩

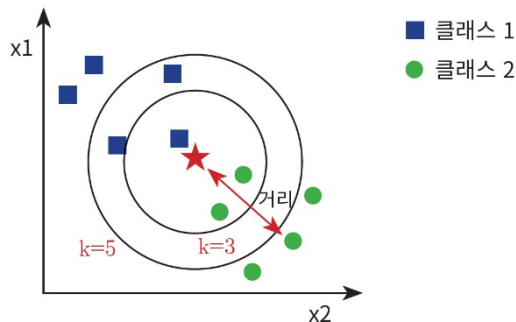
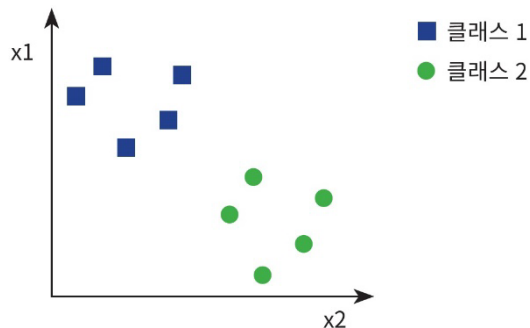
## ◆ 선형회귀 예제



```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from sklearn.linear_model import LinearRegression
4
5 # 1. 학습용 데이터 생성 (X: 독립 변수, y: 종속 변수)
6 # 2차원 배열 형태로 변환 (N 샘플 수, 1 특성 수)
7 X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
8 y = np.array([2, 4, 5, 4, 5])
9
10 # 2. 선형 회귀 모델 생성 및 학습
11 model = LinearRegression()
12 model.fit(X, y)
13
14 # 3. 모델의 기울기(coef_)와 절편(intercept_) 확인
15 print(f'기울기 (Slope/Coefficient): {model.coef_[0]:.4f}')
16 print(f'절편 (Intercept): {model.intercept_:.4f}')
17
18 # 4. 예측 수행
19 y_pred = model.predict(X)
20
21 # 5. 결과 시각화
22 plt.scatter(X, y, color='blue')
23 plt.plot(X, y_pred, color='red')
24 plt.xlabel('X')
25 plt.ylabel('y')
26 plt.show()
```

# kNN 알고리즘

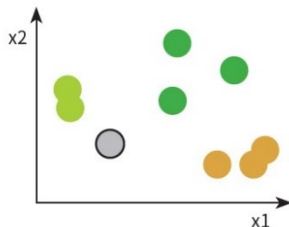
- ◆ k-Nearest Neighbor(kNN) 알고리즘은 모든 기계 학습 알고리즘 중에서도 가장 간단하고 이해하기 쉬운 분류 알고리즘
- ◆ 새로운 데이터가 입력되면 기존 데이터와의 거리를 비교하여 어느 데이터 군에 속하는지 판단 → 분류(Classification) 알고리즘 원리



# kNN 알고리즘

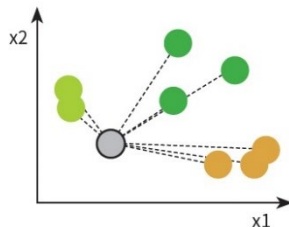
## ◆ kNN 알고리즘 과정

1. 데이터를 관찰한다.



회색 원은 어디에 속해야 할까?

2. 거리를 계산한다.



회색 원과 다른 원들간의 거리를 계산한다.

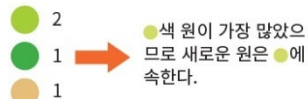
3. 이웃을 찾는다.

| 점 |  | 거리  |      |
|---|--|-----|------|
|   |  | 2.1 | → 1등 |
|   |  | 2.4 | → 2등 |
|   |  | 3.1 | → 3등 |
|   |  | 4.5 | → 4등 |

거리에 따라서 이웃 원들을 정렬한다.

4. 새로운 데이터에 대하여 투표한다.

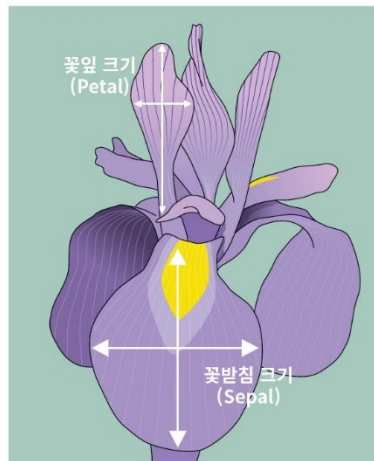
클래스 투표수



가장 가까운 k개의 이웃 중에서 가장 많은 표를 얻은 클래스로 분류한다.

## ◆ kNN 예제

- 아이리스 (붓꽃) 분류
- sklearn 라이브러리에는 아이리스 데이터셋(50개의 꽃 샘플)이 있음



|                                 | 1. 꽃받침 길이 (Sepal Length) | 2. 꽃받침 너비 (Sepal Width) | 3. 꽃잎 길이 (Petal Length) | 4. 꽃잎 너비 (Petal Width) |
|---------------------------------|--------------------------|-------------------------|-------------------------|------------------------|
| 5.1,3.8,1.9,0.4,Iris-setosa     | 5.1                      | 3.8                     | 1.9                     | 0.4                    |
| 4.8,3.0,1.4,0.3,Iris-setosa     | 4.8                      | 3.0                     | 1.4                     | 0.3                    |
| 5.1,3.8,1.6,0.2,Iris-setosa     | 5.1                      | 3.8                     | 1.6                     | 0.2                    |
| 4.6,3.2,1.4,0.2,Iris-setosa     | 4.6                      | 3.2                     | 1.4                     | 0.2                    |
| 5.3,3.7,1.5,0.2,Iris-setosa     | 5.3                      | 3.7                     | 1.5                     | 0.2                    |
| 5.0,3.3,1.4,0.2,Iris-setosa     | 5.0                      | 3.3                     | 1.4                     | 0.2                    |
| 7.0,3.2,4.7,1.4,Iris-versicolor | 7.0                      | 3.2                     | 4.7                     | 1.4                    |
| 6.4,3.2,4.5,1.5,Iris-versicolor | 6.4                      | 3.2                     | 4.5                     | 1.5                    |
| 6.9,3.1,4.9,1.5,Iris-versicolor | 6.9                      | 3.1                     | 4.9                     | 1.5                    |
| 5.5,2.3,4.0,1.3,Iris-versicolor | 5.5                      | 2.3                     | 4.0                     | 1.3                    |
| 6.5,2.8,4.6,1.5,Iris-versicolor | 6.5                      | 2.8                     | 4.6                     | 1.5                    |
| 5.7,2.8,4.5,1.3,Iris-versicolor | 5.7                      | 2.8                     | 4.5                     | 1.3                    |

(아이리스 꽃: 꽃받침 길이와 너비, 꽃잎 길이와 너비)

## ◆ kNN 예제

**iris setosa**



petal

sepal

**iris versicolor**



petal

sepal

**iris virginica**



petal

sepal

[ Petal: 꽃잎, Sepal: 꽃받침 ]

## ◆ kNN 예제

- 파이썬 코드: sklearn 라이브러리 사용

```
1 # 1. 필요한 라이브러리 및 데이터 불러오기
2 from sklearn.datasets import load_iris
3 from sklearn.model_selection import train_test_split
4 from sklearn.neighbors import KNeighborsClassifier
5 from sklearn.metrics import accuracy_score
6
7 # 붓꽃 데이터셋 로드
8 iris = load_iris()
9 X = iris.data # 특성 데이터 (꽃받침/꽃잎의 길이와 너비)
10 y = iris.target # 정답 데이터 (0: Setosa, 1: Versicolor, 2: Virginica)
11
12 # 2. 학습용 데이터와 테스트용 데이터 분리 (8:2 비율)
13 X_train, X_test, y_train, y_test = train_test_split(
14     X, y, test_size=0.2, random_state=42
15 )
16
17 # 3. KNN 분류 모델 생성 및 학습 (이웃 수 n_neighbors = 3 설정)
18 knn_clf = KNeighborsClassifier(n_neighbors=3)
19 knn_clf.fit(X_train, y_train)
20
```

```
21 # 4. 테스트 데이터를 이용해 예측 수행
22 pred = knn_clf.predict(X_test)
23
24 # 5. 모델 정확도 평가
25 accuracy = accuracy_score(y_test, pred)
26 print(f"KNN 모델 정확도(Accuracy): {accuracy:.4f}")
27
28 # 6. 새로운 가상의 데이터 예측해보기
29 # 예: [꽃받침 길이 6.1, 꽃받침 너비 2.8, 꽃잎 길이 4.7, 꽃잎 너비 1.2]
30 new_data = [[6.1, 2.8, 4.7, 1.2]]
31 new_pred = knn_clf.predict(new_data)
32 print(f"새로운 데이터의 예측 품종 이름: {iris.target_names[new_pred][0]}")
33
```

<

Shell ×

```
>>> %Run -c $EDITOR_CONTENT
KNN 모델 정확도(Accuracy): 1.0000
새로운 데이터의 예측 품종 이름: versicolor
```