5주차(2/3)

기계학습 작업 흐름 2

파이썬으로 배우는 기계학습

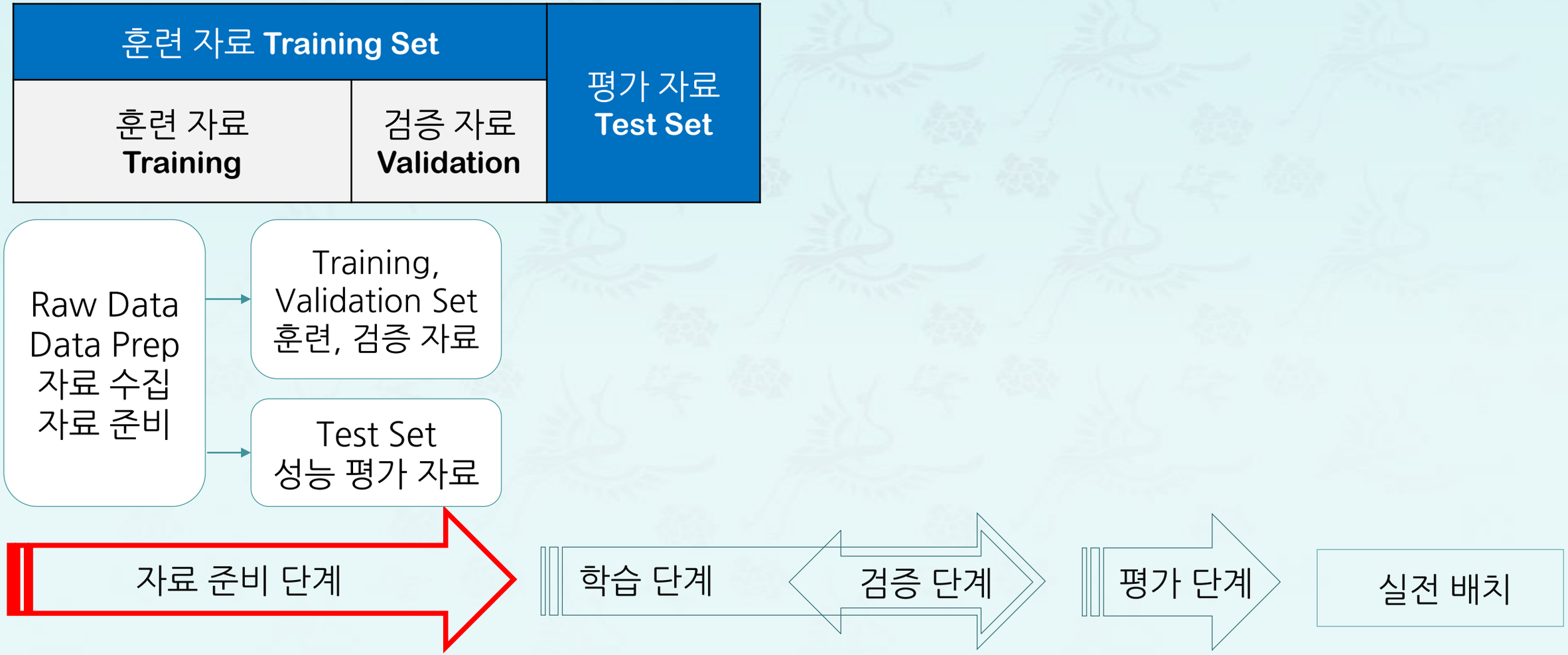
한동대학교
김영섭 교수

기계학습 작업 흐름 2

- 학습 목표
  - 기계학습의 전반적 작업의 흐름을 이해한다.

- 학습 내용
  - 학습 자료 준비
  - 학습 자료 전처리
  - 오차/정확도 측정

1. 학습 자료 준비 단계: 자료읽기



1. 학습 자료 준비 단계: 자료읽기

- joydata.txt

```
!cat data/joydata.txt
```

```
-1.72   -3.12   1
0.31    1.85    1
1.56    2.85    1
2.64    2.41    1
1.23    2.54    1
1.33    2.03    1
1.26    2.68    1
2.58    1.79    1
2.40    0.91    1
0.51    2.44    1
```

1. 학습 자료 준비 단계: 자료 특성 분석

```
import numpy as np
data = np.genfromtxt('data/joydata.txt')
x, y = data[:, :2], data[:, 2]
y = y.astype(np.int)
print(x[:5], y[:5])
print(x[-5:], y[-5:])
```

```
[[-1.72 -3.12]
 [ 0.31  1.85]
 [ 1.56  2.85]
 [ 2.64  2.41]
 [ 1.23  2.54]] [1 1 1 1 1]
[[-2.26  0.01]
 [-1.41 -0.23]
 [-1.2  -0.71]
 [-1.69  0.7 ]
 [-1.52 -1.14]] [0 0 0 0 0]
```

2. 학습 자료 전처리: 전처리 작업의 종류

- 셔플링**(shuffling)**
- 피쳐 스케일링**(feature scaling)**

2. 학습 자료 전처리: 셔플링(shuffling)

```
!cat data/joydata.txt
```

```
-1.72   -3.12   1
0.31    1.85    1
1.56    2.85    1
2.64    2.41    1
1.23    2.54    1
1.33    2.03    1
1.26    2.68    1
2.58    1.79    1
2.40    0.91    1
0.51    2.44    1
```

2. 학습 자료 전처리: 셔플링(shuffling) 코드 (1)

```
import numpy as np
data = np.genfromtxt('data/joydata.txt')
x, y = data[:, :2], data[:, 2]
y = y.astype(np.int)
np.random.shuffle(x)
np.random.shuffle(y)
print(x[:5], y[:5])
print(x[-5:], y[-5:])
```

```
[[-5.27 -1.78]
 [ 1.33  2.03]
 [ 1.    0.46]
 [-1.48 -1.17]
 [ 1.14  3.01]] [1 0 1 1 0]
[[-3.45 -0.62]
 [-1.26 -2.9 ]
 [ 1.9   1.34]
 [-1.08 -1.23]
 [ 2.52  1.83]] [1 1 0 1 0]
```

2. 학습 자료 전처리: 셔플링(shuffling) 코드 (1) 버그

```
import numpy as np
data = np.genfromtxt('data/joydata.txt')
x, y = data[:, :2], data[:, 2]
y = y.astype(np.int)
np.random.shuffle(x)
np.random.shuffle(y)
print(x[:5], y[:5])
print(x[-5:], y[-5:])
```

```
[[-5.27 -1.78]
 [ 1.33  2.03]
 [ 1.    0.46]
 [-1.48 -1.17]
 [ 1.14  3.01]] [1 0 1 1 0]
[[-3.45 -0.62]
 [-1.26 -2.9 ]
 [ 1.9   1.34]
 [-1.08 -1.23]
 [ 2.52  1.83]] [1 1 0 1 0]
```

2. 학습 자료 전처리: 셔플링(shuffling) 코드 (2) – 버그 제거

```
import numpy as np
data = np.genfromtxt('data/joydata.txt')
np.random.shuffle(data)
x, y = data[:, :2], data[:, 2]
y = y.astype(np.int)
print(x[:5], y[:5])
print(x[-5:], y[-5:])
```

```
[[ 0.68  1.43]
 [-3.07 -2.09]
 [ 3.87  2.91]
 [-1.2  -0.71]
 [-3.08 -1.05]] [1 0 1 0 0]
[[-1.41 -0.23]
 [ 1.26  2.68]
 [ 1.9   1.34]
 [ 0.9   2.05]
 [ 1.26  1.17]] [0 1 1 1 1]
```

2. 학습 자료 전처리: 셔플링(shuffling) 코드 (2) – 버그 제거

```
import numpy as np
data = np.genfromtxt('data/joydata.txt')
np.random.shuffle(data)
x, y = data[:, :2], data[:, 2]
y = y.astype(np.int)
print(x[:5], y[:5])
print(x[-5:], y[-5:])
```

```
[[ 0.68  1.43]
 [-3.07 -2.09]
 [ 3.87  2.91]
 [-1.2  -0.71]
 [-3.08 -1.05]] [1 0 1 0 0]
[[-1.41 -0.23]
 [ 1.26  2.68]
 [ 1.9   1.34]
 [ 0.9   2.05]
 [ 1.26  1.17]] [0 1 1 1 1]
```

2. 학습 자료 전처리: 피쳐 스케일링(Feature Scaling)의 종류

- 정규화(**normalization**)
- 표준화(**standardization**)

3. 피쳐 스케일링(Feature Scaling): 정규화

- **min-max scaling**
- 자료 범위: **0** 부터 **1**사이
- 계산 방법:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

3. 피쳐 스케일링(Feature Scaling): 표준화

- 특이값에 영향을 덜 받음
- 자료 범위: 평균값 **0**, 표준편차 **1**

3. 피쳐 스케일링(Feature Scaling): 표준화

- 특이값에 영향을 덜 받음
- 자료 범위: 평균값 **0**, 표준편차 **1**
- 계산 방법:

$$\mathbf{x}_j := \frac{\mathbf{x}_j - \mu_j}{\sigma_j}$$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

- x_j: 입력**x**의 **j**번째 특성
- μ_j: 입력**x**의 **j**번째 특성의 평균값
- σ_j: 입력**x**의 **j**번째 특성의 표준편차

3. 피쳐 스케일링(Feature Scaling): 정규화 코드

```
xmax = np.max(x)
xmin = np.min(x)
x = (x - xmin)/(xmax - xmin)
```

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

3. 피쳐 스케일링(Feature Scaling): 표준화 코드

```
mu = x.mean(axis=0)
sigma = x.std(axis=0)
x = (x - mu) / sigma
```

$$\mathbf{x}_j := \frac{\mathbf{x}_j - \mu_j}{\sigma_j}$$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

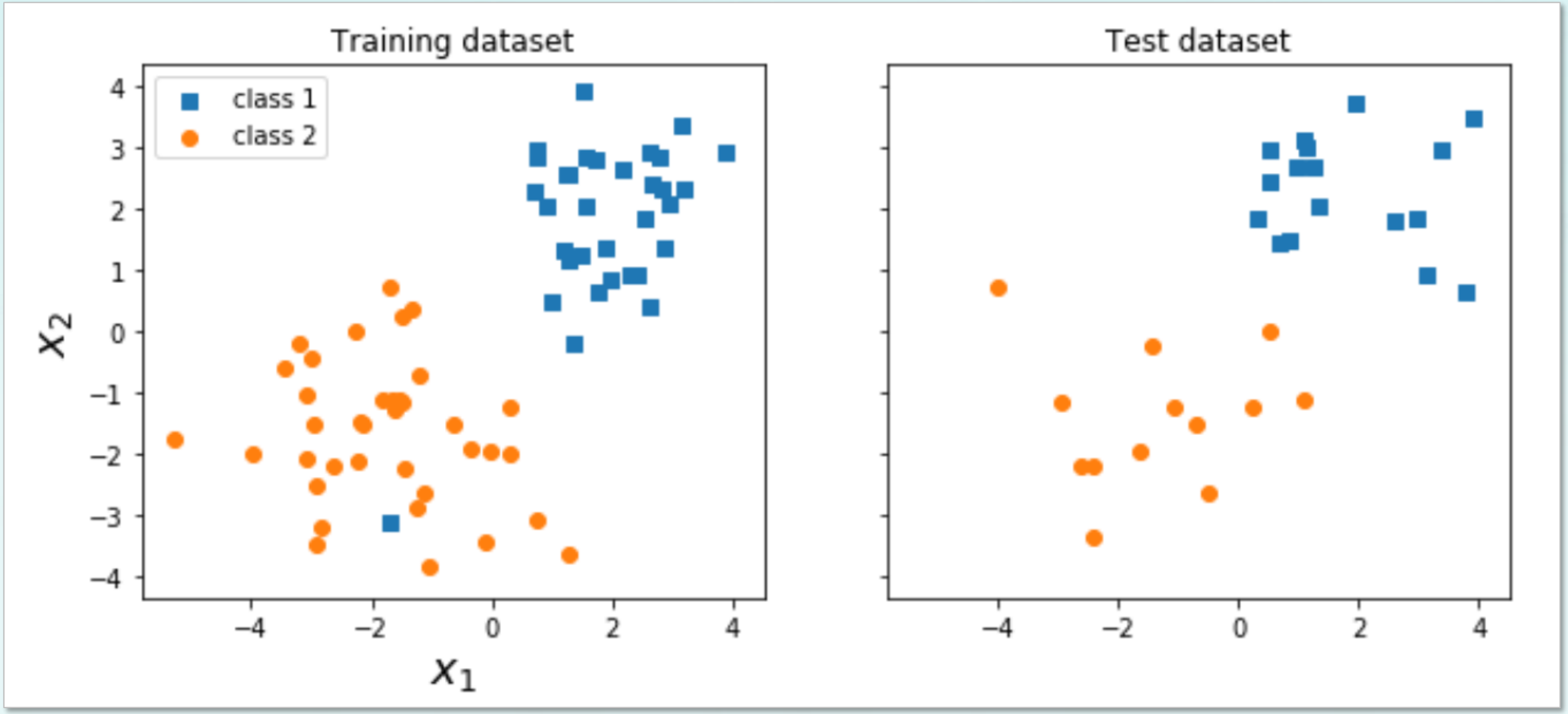
4. 학습 자료 준비 단계: 자료분리

- 훈련 자료 **vs** 테스트 자료
- **8:2** 혹은 **7:3**

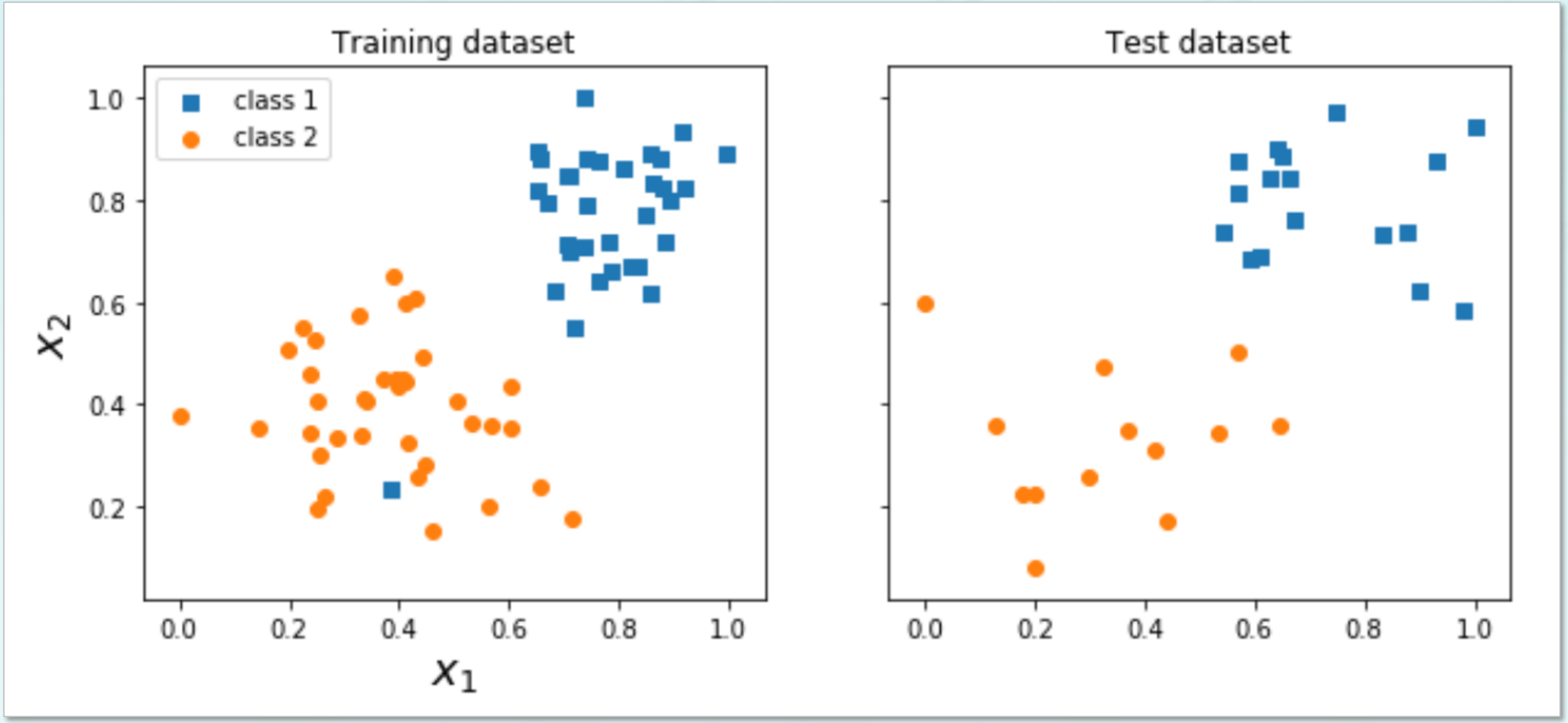
4. 학습 자료 준비 단계: 자료분리

```
1  import numpy as np
2  data = np.genfromtxt('data/joydata.txt')
3  np.random.seed(1)
4  np.random.shuffle(data)
5  x, y = data[:, :2], data[:, 2]
6  y = y.astype(np.int)
7
8  num = int(x.shape[0] * 0.8) ## percentage
9  x_train, x_test = x[:num], x[num:]
10 y_train, y_test = y[:num], y[num:]
```

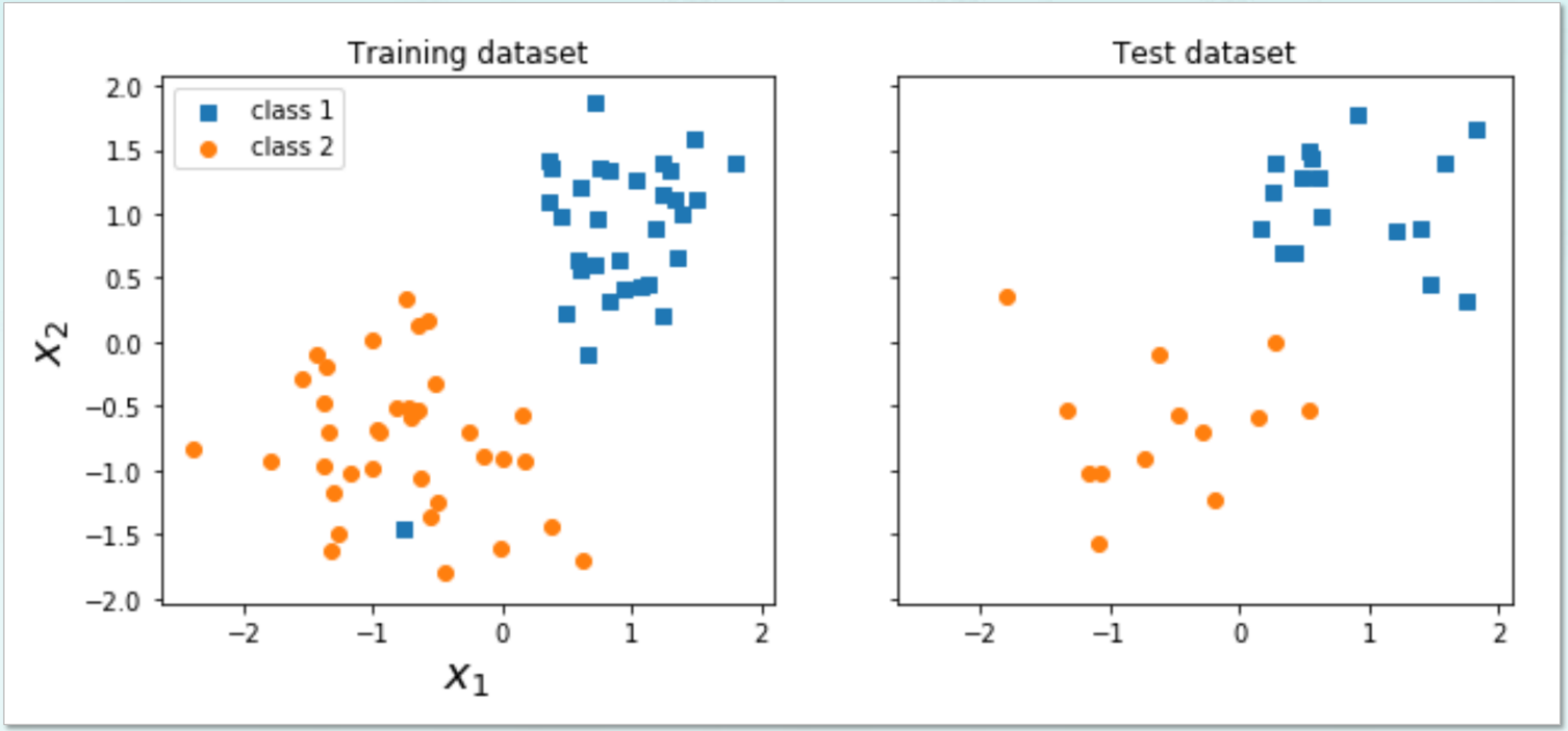
5. 훈련자료 시각화: 셔플링(Shuffling)



5. 훈련자료 시각화: 피쳐 스케일링 (Feature Scaling) - 정규화



5. 훈련자료 시각화: 피쳐 스케일링 (Feature Scaling) - 표준화



6. 편향 처리 방식

- 편향을 포함한 자료 구조

$$
\begin{aligned}
z &= \mathbf{w}^{\mathbf{T}}\mathbf{x} \\
&= w_0x_0 + w_1x_1 + \ldots + w_nx_n \\
&= \sum_{j=0}^{n} x_jw_j
\end{aligned}
$$

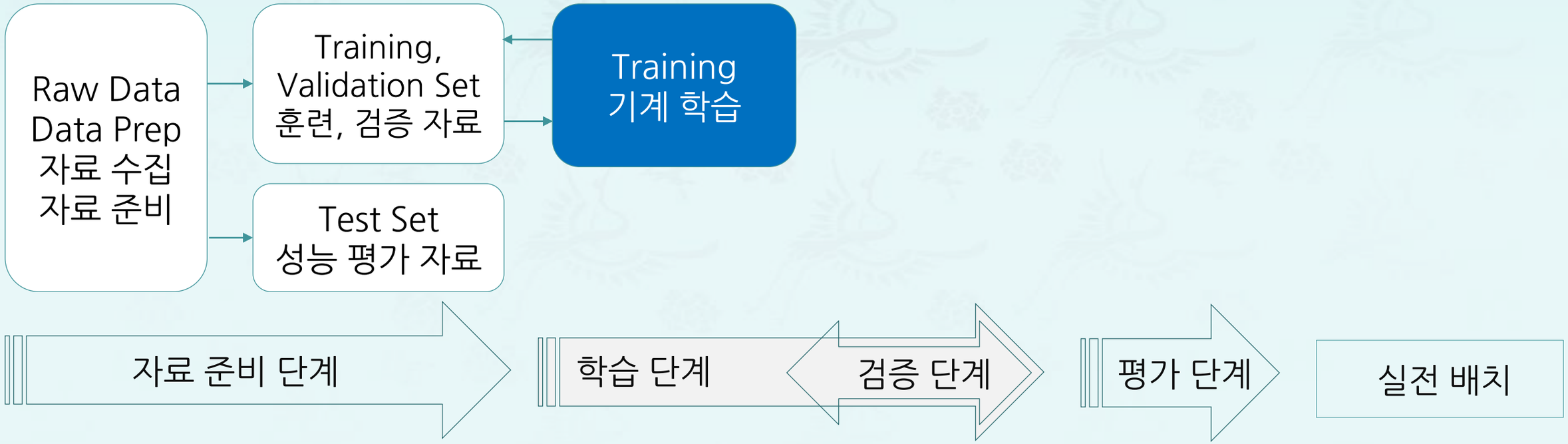
방법 1

- 편향을 포함하지 않은 자료 구조

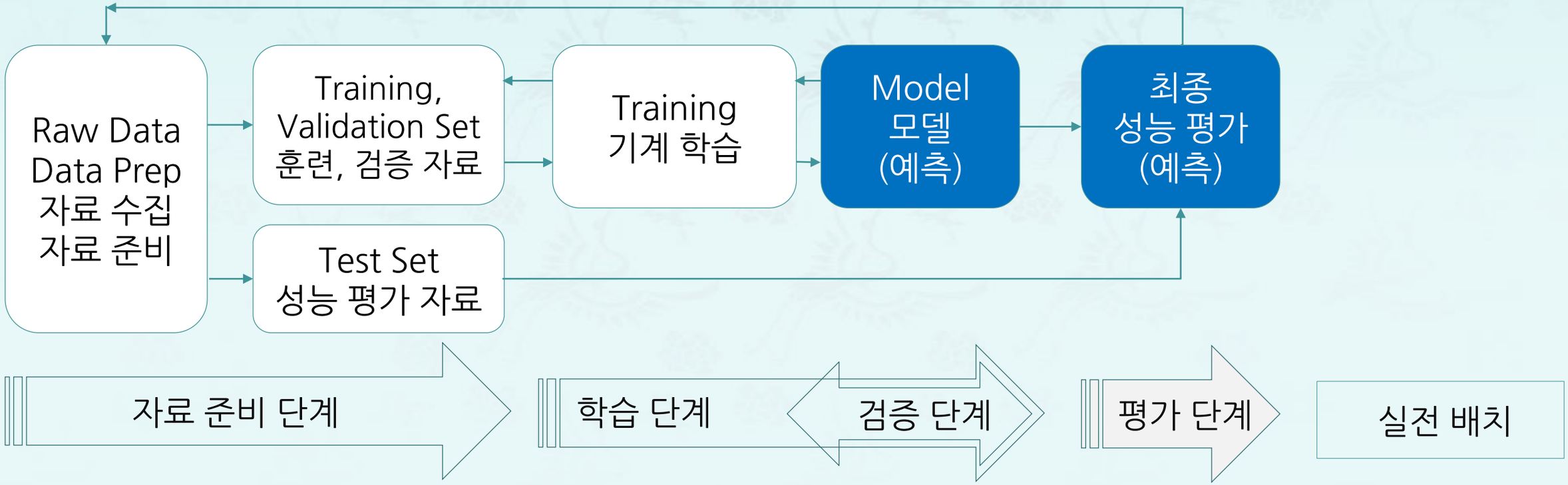
$$
\begin{aligned}
z &= \mathbf{w}^{\mathbf{T}}\mathbf{x} + b \\
&= w_1x_1 + w_2x_2 + \ldots + w_nx_n \\
&= \sum_{j=1}^{n} x_jw_j + b
\end{aligned}
$$

방법 2

7. 학습 단계: 훈련 (기계학습)



7. 예측 및 평가 단계: 예측 및 성능 평가



8. 이진분류 퍼셉트론 예측: 예측함수 코드

- 함수 이름
  - **perceptron_predict(X, w)**
- 입력 인자
  - **X:** 예측할 입력 자료
    (검증자료, 테스트 자료**)**
  - **w:** 가중치
- 함수 반환 값
  - **yhat**
    - 형상: **(m samples, 1)**
    - 형식: **0, 1**

```python
def perceptron_predict(X, w):
    z = np.dot(X, w)
    yhat = np.where(z > 0., 1, 0)
    return yhat
```

9. 이진분류 퍼셉트론 평가: 평가 방법

- **y**: 클래스 레이블(주어진 값, 정답)
- **yhat**: 예측값
- **y**와 **yhat**을 비교

```python
def perceptron_predict(X, w):
    z = np.dot(X, w)
    yhat = np.where(z > 0., 1, 0)
    return yhat
```

9. 이진분류 퍼셉트론 평가: 평가 코드 버전 1

- **y**: 클래스 레이블(주어진 값, 정답)
- **yhat**: 예측값
- **y**와 **yhat**을 비교

```python
def perceptron_predict(X, w):
    z = np.dot(X, w)
    yhat = np.where(z > 0., 1, 0)
    return yhat
```

```python
#version 0.1
yhat = perceptron_predict(X_train, w)
missed = 0  # misclassified count
m_samples = len(y_train)
for m in range(m_samples):
    if yhat[m] != y_train[m]:
        missed  += 1
print('Misclassified:{}/{}'.
      format(missed, m_samples))
```

```
Misclassified:1/80
```

9. 이진분류 퍼셉트론 평가: 평가 코드 버전 2

- 자료 변경!

```python
def perceptron_predict(X, w):
    z = np.dot(X, w)
    yhat = np.where(z > 0., 1, 0)
    return yhat
```

```python
#version 0.1
yhat = perceptron_predict(X_test, w)
missed = 0
m_samples = len(y_test)
for m in range(m_samples):
    if yhat[m] != y_test[m]:
        missed  += 1
print('Misclassified:{}/{}'.
      format(missed, m_samples))
```

```
Misclassified:1/20
```

```python
#version 0.1
yhat = perceptron_predict(X_train, w)
missed = 0  # misclassified count
m_samples = len(y_train)
for m in range(m_samples):
    if yhat[m] != y_train[m]:
        missed  += 1
print('Misclassified:{}/{}'.
      format(missed, m_samples))
```

```
Misclassified:1/80
```

9. 이진분류 퍼셉트론 평가: 평가 코드 버전 2

- 자료 변경!

```python
def perceptron_predict(X, w):
    z = np.dot(X, w)
    yhat = np.where(z > 0., 1, 0)
    return yhat
```

```python
#version 0.1
yhat = perceptron_predict(X_test, w)
missed = 0
m_samples = len(y_test)
for m in range(m_samples):
    if yhat[m] != y_test[m]:
        missed  += 1
print('Misclassified:{}/{}'.
      format(missed, m_samples))
```

```
Misclassified:1/20
```

```python
#version 0.1
yhat = perceptron_predict(X_train, w)
missed = 0  # misclassified count
m_samples = len(y_train)
for m in range(m_samples):
    if yhat[m] != y_train[m]:
        missed  += 1
print('Misclassified:{}/{}'.
      format(missed, m_samples))
```

```
Misclassified:1/80
```

9. 이진분류 퍼셉트론 평가: 평가 코드 버전 2

```python
def perceptron_predict(X, w):
    z = np.dot(X, w)
    yhat = np.where(z > 0., 1, 0)
    return yhat
```

```python
#version 0.1
yhat = perceptron_predict(X_test, w)
missed = 0
m_samples = len(y_test)
for m in range(m_samples):
    if yhat[m] != y_test[m]:
        missed  += 1
print('Misclassified:{}/{}'.
      format(missed, m_samples))
```

```
Misclassified:1/20
```

```python
#version 0.1
yhat = perceptron_predict(X_train, w)
missed = 0  # misclassified count
m_samples = len(y_train)
for m in range(m_samples):
    if yhat[m] != y_train[m]:
        missed  += 1
print('Misclassified:{}/{}'.
      format(missed, m_samples))
```

```
Misclassified:1/80
```

9. 이진분류 퍼셉트론 평가: 평가 코드 버전 3

```
#version 0.1
yhat = perceptron_predict(X_test, w)
missed = 0
m_samples = len(y_test)
for m in range(m_samples):
    if yhat[m] != y_test[m]:
        missed  += 1
print('Misclassified:{}/{}'.
      format(missed, m_samples))
```

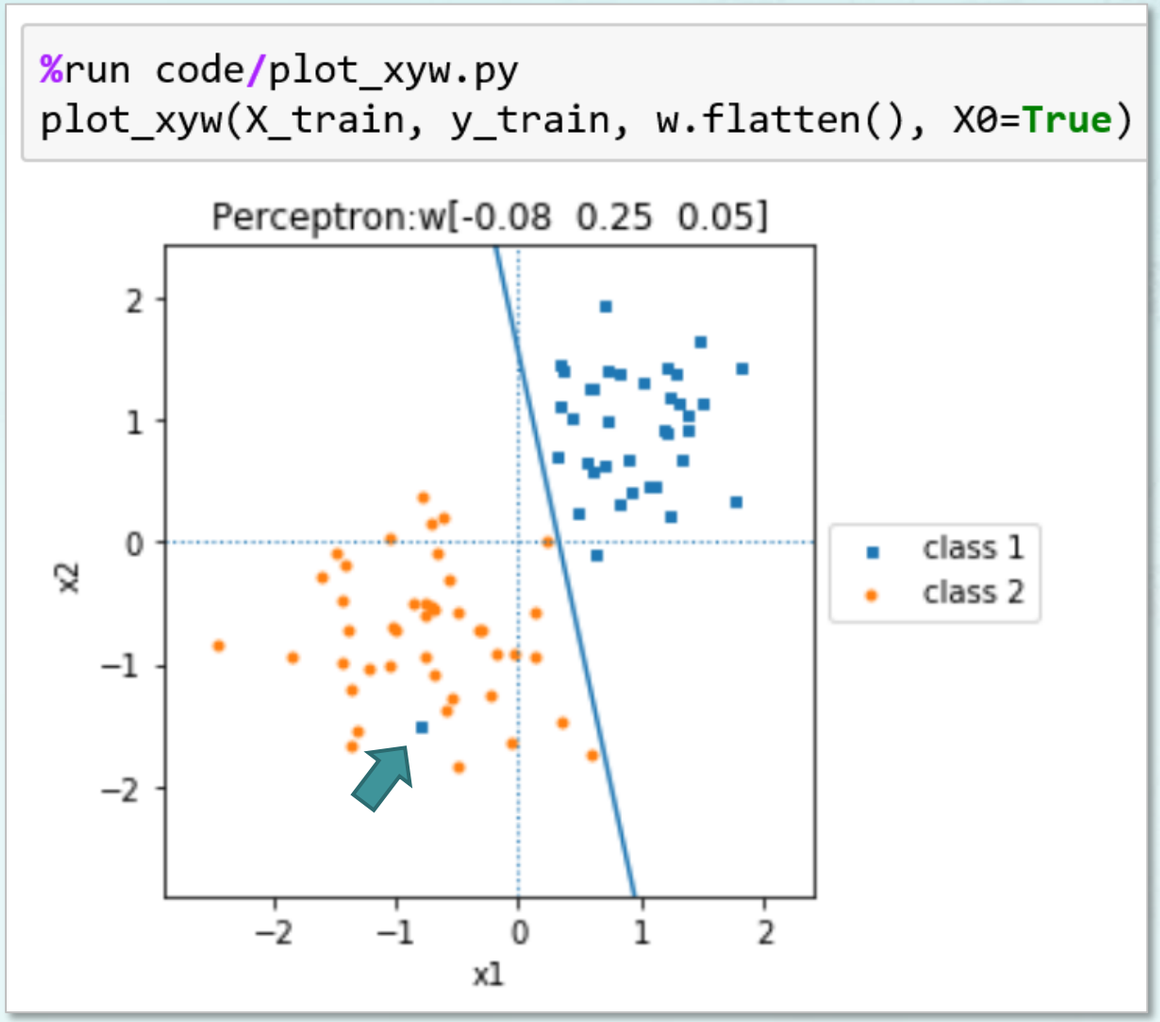
```
Misclassified:1/20
```

```
#version 0.2
yhat = perceptron_predict(X_test, w)
missed = np.sum(yhat.flatten() != y_test)
print('Misclassified:{}/{}'.
      format(missed, m_samples))
```

```
Misclassified:1/20
```

10. 예측 및 평가 시각화: 훈련 자료 분류 결과

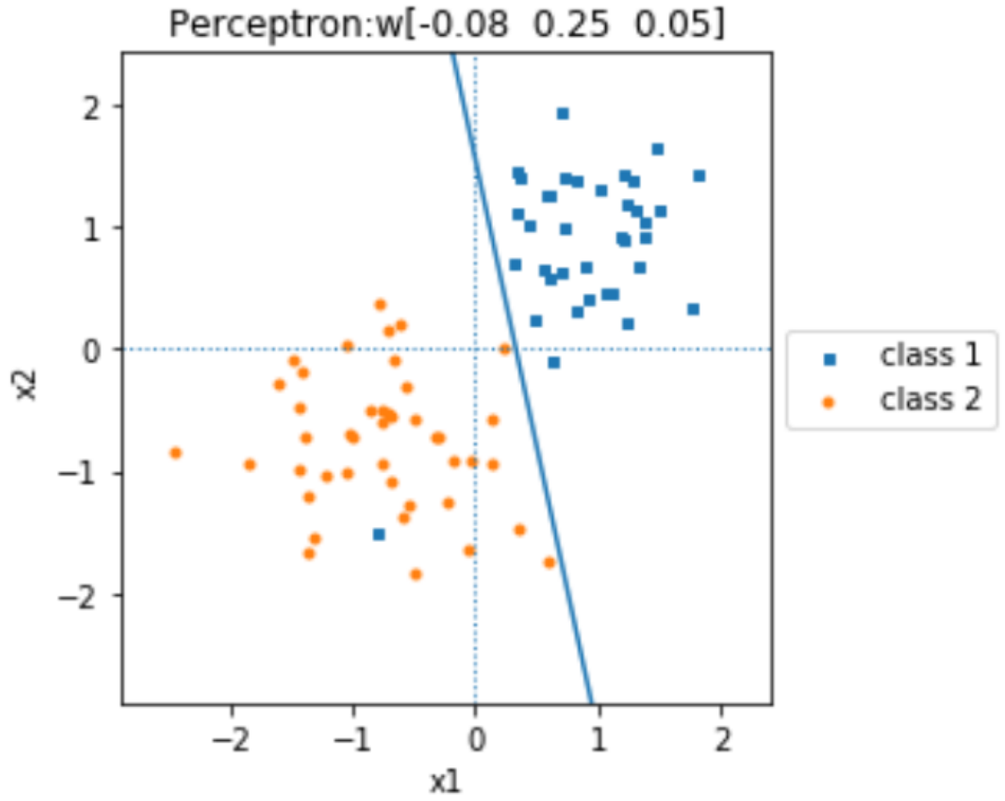
```
%run code/plot_xyw.py
plot_xyw(X_train, y_train, w.flatten(), X0=True)
```



10. 예측 및 평가 시각화

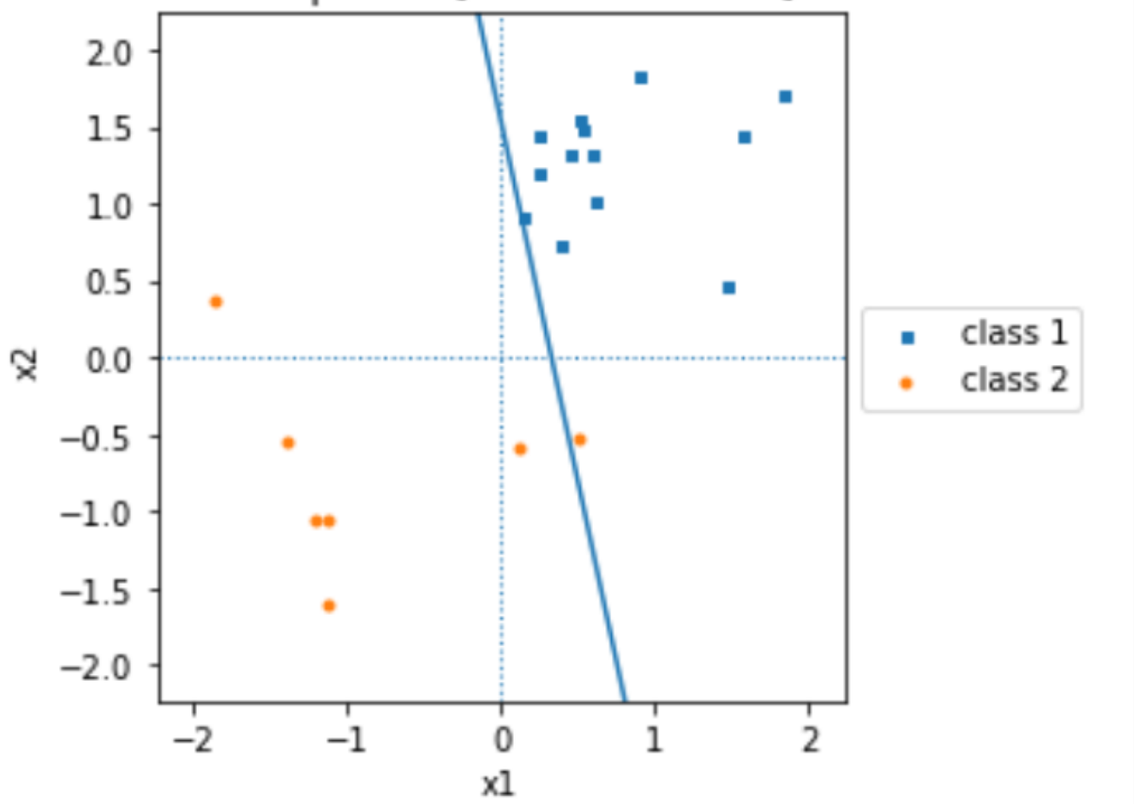
- 훈련 자료 분류 결과

```
%run code/plot_xyw.py
plot_xyw(X_train, y_train, w.flatten(), X0=True)
```



- 테스트 자료 분류 결과

```
plot_xyw(X_test, y_test, w.flatten(), X0=True)
```



기계학습 작업 흐름 2

- 학습 정리
  - 기계학습을 적용하는데 필요한 작업 흐름도를 이해하기
  - 자료 준비와 전처리 과정 필요성과 역할
  - 모델의 정확도 평가