

Elevate Your Lifestyle with Our Mobile Technology

LOL 승패 예측 데이터 분석

LEAGUE OF
LEGENDS
DIAMOND
RANKED GAMES
(10 MIN)

REALLYGREATSITE.COM

목차

3교시 머신러닝 이론

5교시 모델 마무리

4교시 데이터 전처리

5교시 모델 성능비교

5교시 머신러닝 모델링

#01 머신러닝 이론

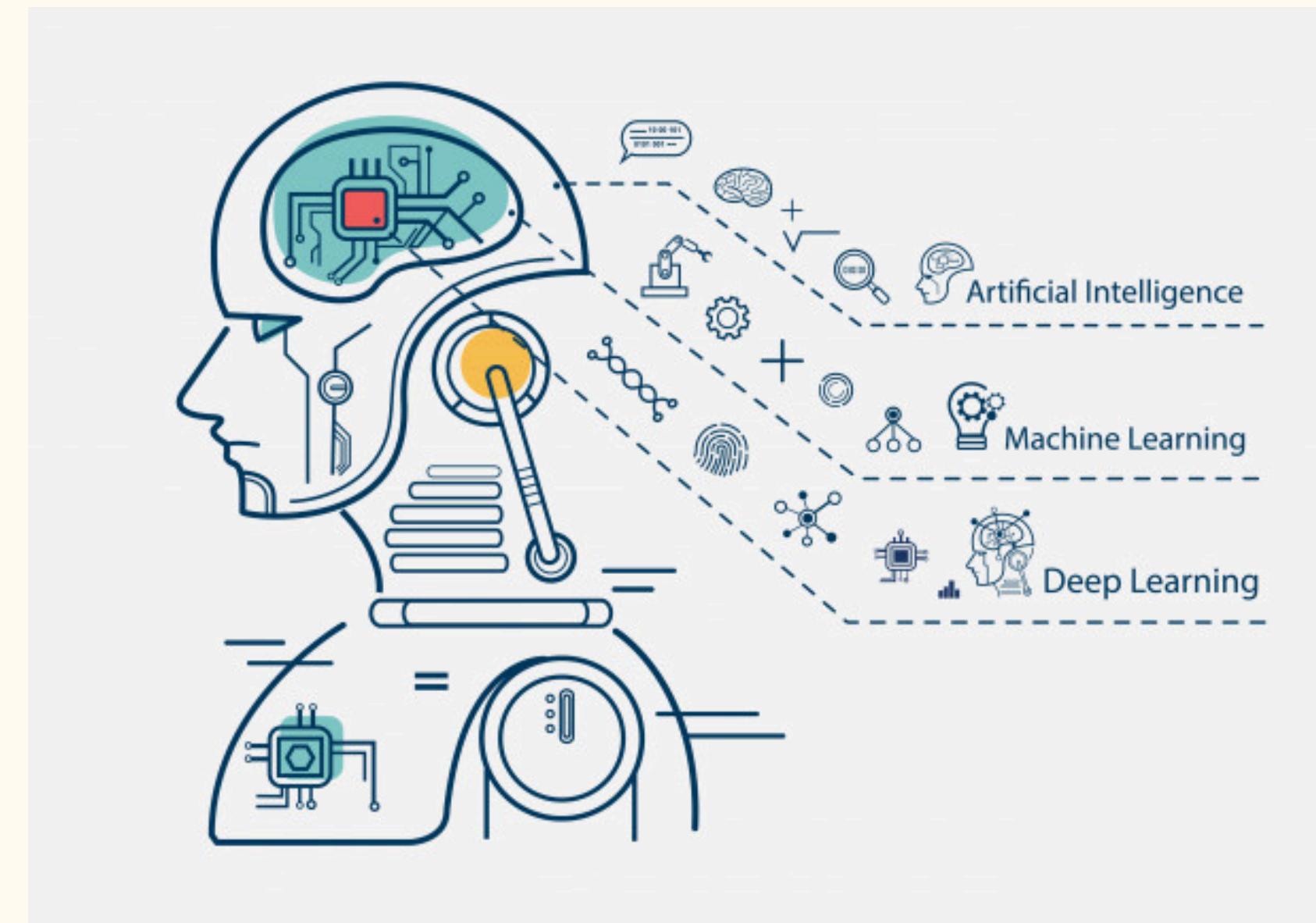
시작 하기 앞서



#01 머신러닝 이론

머신러닝이란?

말 그대로 기계학습. 컴퓨터가 데이터를 통해 학습하는 것



#01 머신러닝 이론

Quiz

머신러닝은 어디에 활용될까요?

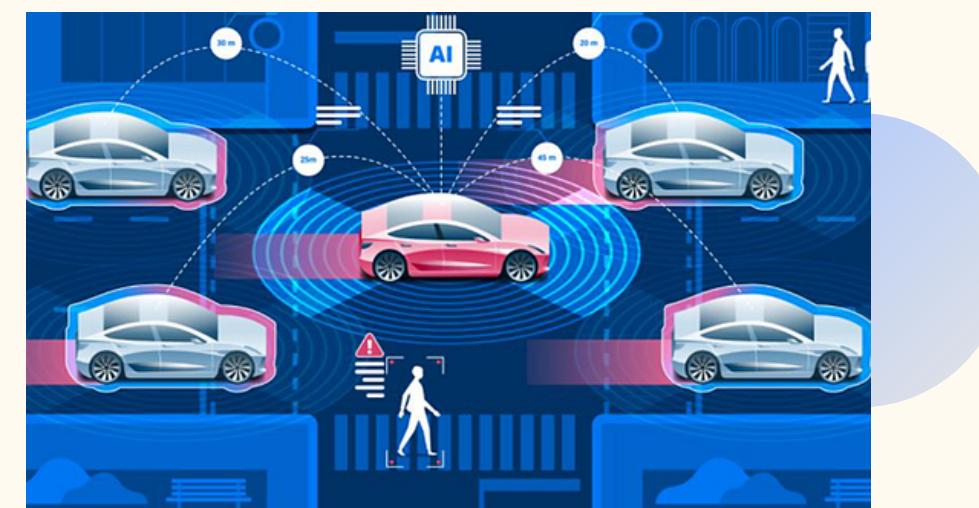
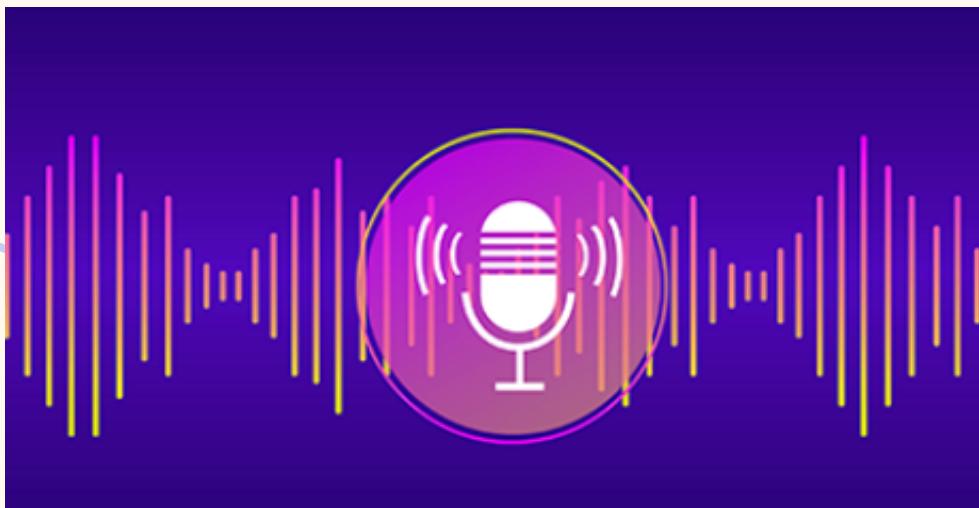


#01 머신러닝 이론

Quiz

머신러닝은 어디에 활용될까요?

ex) 스마트폰의 음성인식기능, 자율주행자동차,
데이터분석 등 모두 머신러닝의 결과물



#01 머신러닝 이론 데이터

데이터란?

머신러닝의 기본자료

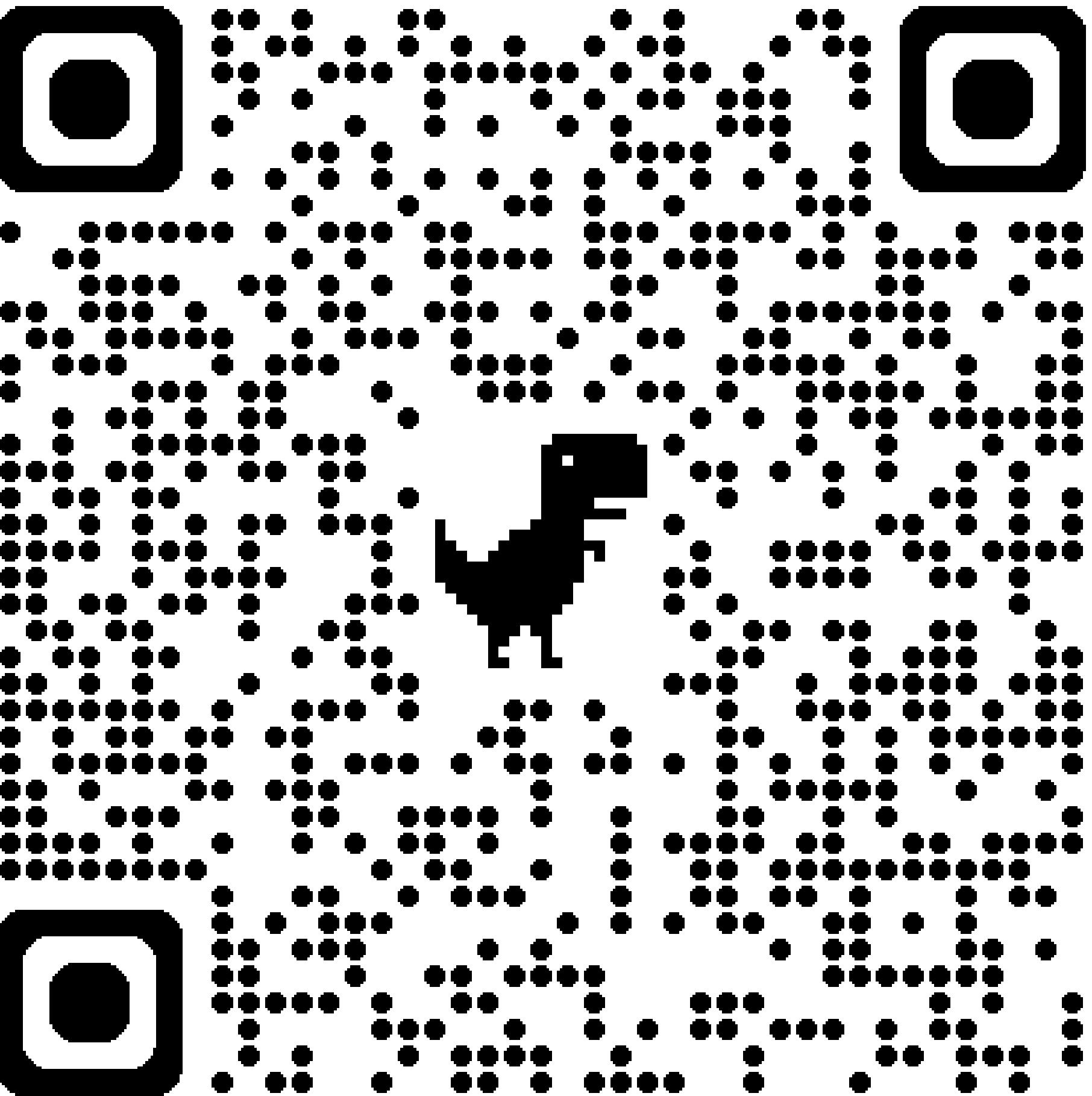
데이터는 우리가 알고싶은 정보나 예측하고 싶은 결과와 관련된 다양한 형태의 정보.
예를들어 키와 몸무게, 시험 점수, 쇼핑내역 등이 모두 학습에 쓰이는 데이터.

| id | hour | hour_bef_temperature | hour_bef_precipitation | hour_bef_windspeed | hour_bef_humidity | hour_bef_visibility | hour_bef_ozone | hour_bef_pm10 | hour_bef_pm2.5 | count |
|----|------|----------------------|------------------------|--------------------|-------------------|---------------------|----------------|---------------|----------------|-------|
| 3 | 20 | 16.3 | 1 | 1.5 | 89 | 576 | 0.027 | 76 | 33 | 4 |
| 6 | 13 | 20.1 | 0 | 1.4 | 48 | 916 | 0.042 | 73 | 40 | 15 |
| 7 | 6 | 13.9 | 0 | 0.7 | 79 | 1382 | 0.033 | 32 | 19 | 2 |
| 8 | 23 | 8.1 | 0 | 2.7 | 54 | 946 | 0.04 | 75 | 64 | 5 |
| 9 | 18 | 29.5 | 0 | 4.8 | 7 | 2000 | 0.057 | 27 | 11 | 43 |
| 13 | 2 | 13.6 | 0 | 1.7 | 80 | 1073 | 0.027 | 34 | 15 | 3 |
| 14 | 3 | 10.6 | 0 | 1.5 | 58 | 1548 | 0.038 | 62 | 33 | 2 |
| 16 | 21 | 16 | 0 | 6 | 21 | 1961 | 0.05 | 90 | 28 | 14 |
| 19 | 9 | 13.8 | 0 | 1.9 | 64 | 1344 | 0.039 | 93 | 19 | 3 |
| 20 | 14 | 17.2 | 0 | 2.1 | 32 | 1571 | 0.025 | 64 | 19 | 8 |
| 21 | 4 | 15.7 | 0 | 1.6 | 77 | 1060 | 0.028 | 11 | 18 | 4 |
| 22 | 10 | 15.4 | 0 | 2.7 | 62 | 1962 | 0.039 | 99 | 21 | 4 |
| 24 | 9 | 14.1 | 0 | 3.2 | 59 | 1809 | 0.028 | 52 | 18 | 5 |
| 27 | 10 | 9.2 | 0 | 1.5 | 46 | 202 | 0.023 | 9 | 84 | 6 |
| 28 | 1 | 20 | 0 | 1.8 | 58 | 2000 | | | | 7 |
| 29 | 13 | 14 | 1 | 2.8 | 42 | 1518 | 0.03 | 49 | 37 | |
| 30 | 21 | 18.8 | 0 | 2.2 | 34 | 2000 | 0.04 | 32 | 14 | 21 |
| 32 | 17 | 11.5 | 1 | 3 | 91 | 555 | 0.036 | 29 | 21 | 6 |
| 33 | 13 | 22.6 | 0 | 41 | 987 | 0.046 | 64 | 39 | 20 | |
| 34 | 18 | 18 | 1 | 1.9 | 82 | 685 | 0.044 | 99 | 42 | 1 |

#01 머신러닝 이론 데이터

여러분들의 신체데이터를 수집하겠습니다
(의명보장)

https://docs.google.com/spreadsheets/d/11qUPBlULLloBjAdDoFi4B3ElJSSw7Bb_eR1f4OQvT7UQ/edit?usp=sharing



#01 머신러닝 이론

기본용어

피쳐 & 레이블

피쳐(변수) - 입력 데이터의 속성 ex) 키, 몸무게

레이블(타겟변수) - 모델이 예측하려는 결과값 ex) 가족의 평균 키

머신러닝 모델은 피쳐를 통해 레이블을 예측

#01 머신러닝 이론

기본용어

훈련데이터 & 테스트데이터

훈련데이터 - 모델을 학습시키기 위해 사용하는 데이터

테스트데이터 - 학습된 모델의 성능을 평가하기 위해 사용하는 데이터

두개로 나누는 이유는 모델이 새로운데이터를 잘 예측하는지 확인하기 위함

#01 머신러닝 이론 기본용어

예시

신체데이터를 테스트 데이터와 훈련데이터로 분리하기

https://docs.google.com/spreadsheets/d/11qUPBlULLloBjAdDofFi4B3ElJSSw7Bb_eR1f40QvT7UQ/edit?usp=sharing

#01 머신러닝 이론

머신러닝 모델 생성 과정

1. 데이터 수집
2. 데이터 전처리
3. 모델 학습
4. 모델 평가
5. 모델 사용

#01 머신러닝 이론 데이터 수집

데이터를 모으는 단계

#01 머신러닝 이론 데이터 수집

데이터 수집하는 방법들

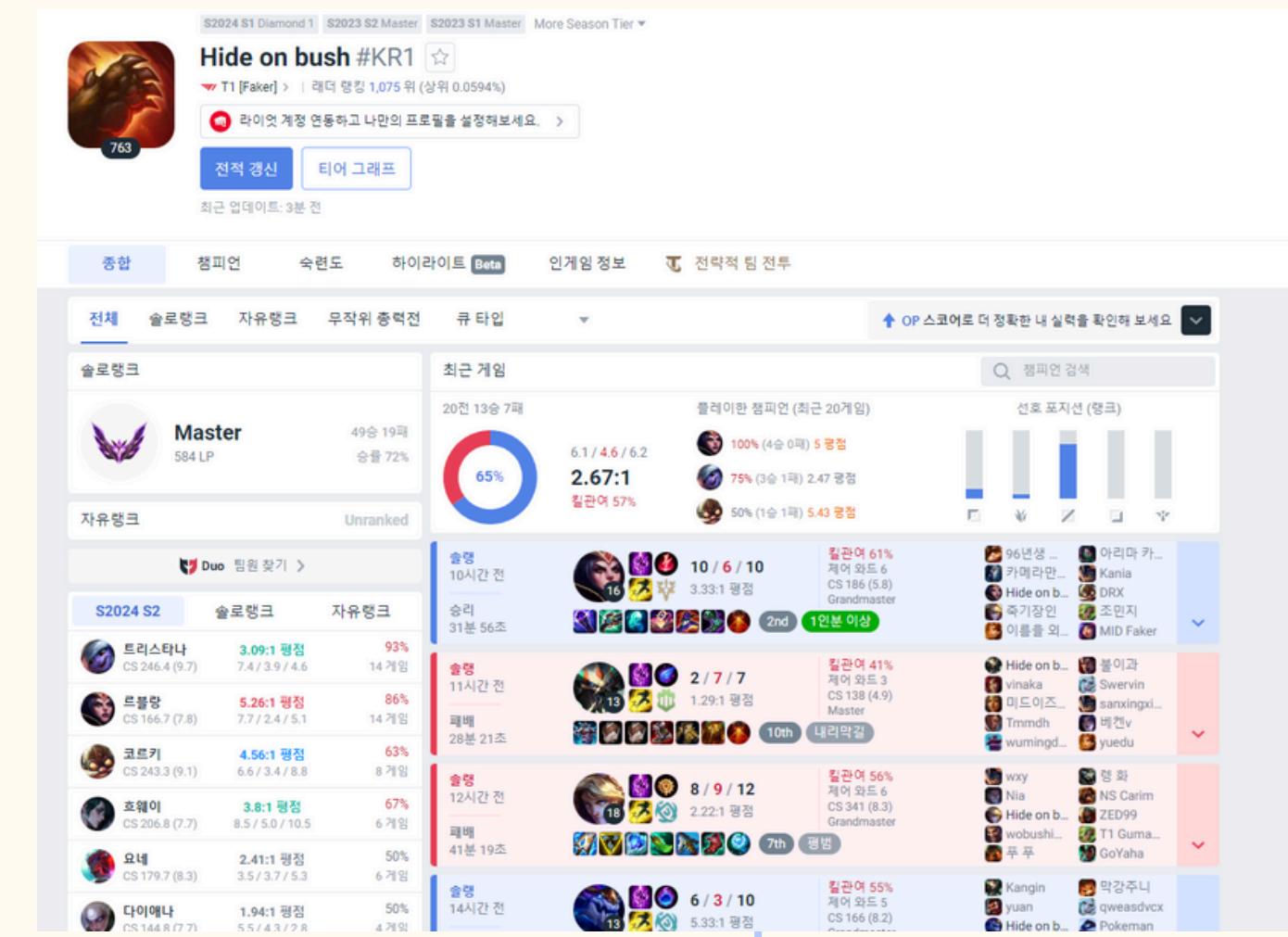
공개데이터 사용

웹 크롤링

API사용 (예 - op.gg)

센서 사용

The screenshot shows the homepage of the 'Library Bigdata' website. At the top, there is a navigation bar with links for '데이터 제공', '데이터 신청', '데이터 활용', '도서관 빅데이터 소개', and '알림소통'. Below the navigation bar, there is a search bar with the placeholder 'search' and a magnifying glass icon. The main content area has a header '장서/대출데이터' with the sub-instruction '도서관별로 공개된 장서/대출 데이터를 보실 수 있습니다.' Below this, there is a table titled '도서관명' listing various libraries and their data availability. The table includes columns for '도서관명', '데이터 유형', '공개범위', and '최근 데이터 제공일'. Each row has download links for 'Text', 'Excel', and 'API'. The table lists 11 entries, including '2.28도서관 장서/대출 데이터', 'U보라작은도서관 장서/대출 데이터', '가락물도서관 장서/대출 데이터', '가산도서관 장서/대출 데이터', '가수원도서관 장서/대출 데이터', '가슴따뜻한작은도서관 장서/대출 데이터', '가양도서관 장서/대출 데이터', '가오도서관 장서/대출 데이터', and '가온누리작은도서관 장서/대출 데이터'. The date provided for most entries is 2024-06-01.



#01 머신러닝 이론 데이터 수집

공개데이터 사용

<https://www.data.go.kr/index.do>

The screenshot shows the search results for the keyword "인구수". The search bar at the top contains "인구수" and the search button is labeled "AND". Below the search bar are filters: 연관 (Related), 합계 (Total), 인원수 (Population), 개체수 (Individual count), 총인구 (Total population), and 갯수 (Count). To the right are buttons for "제공기관별 검색" (Search by provider) and "상세검색" (Advanced search). A message below the search bar states: "'인구수'에 대해 총 439건이 검색되었습니다." (439 items found for 'population').

조건검색 (Conditions Search)

미리보기 (Preview) and **다운로드** (Download) buttons are present for each result item.

- 행정안전부_지역별(행정동) 성별 연령별 주민등록 인구수**
행정동(읍면동)별 성별 연령별 주민등록 인구에 대한 데이터입니다. 행정동은 주민들이 거주하는 지역을 행정능률과 주민편의를 위하여 구분한 행정구역 단위입니다.
제공기관: 행정안전부 | 수정일: 2024-06-05 | 조회수: 22532 | 다운로드: 12465 | 주기성 데이터: 28 | 키워드: 인구수, 성별인구, 주민등록인구
- 행정안전부_지역별(법정동) 성별 연령별 주민등록 인구수**
법정동(읍면동리) 성별 연령별 주민등록 인구에 대한 데이터입니다. 법정동은 시 또는 구의 하위 행정구역으로 법률로 지정한 구역을 말합니다.
제공기관: 행정안전부 | 수정일: 2024-06-05 | 조회수: 15174 | 다운로드: 8030 | 주기성 데이터: 29 | 키워드: 주민등록인구, 인구, 인구수
- XLSX 서울특별시 종량구 연령별 인구수 현황**
서울특별시 종량구의 연령별 인구 분포 정보를 제공합니다. 연령별 인구현황, 주민등록자, 거주자, 거주불명자, 재외국민 등의 정보를 제공합니다.
제공기관: 서울특별시 종량구 | 수정일: 2024-05-23 | 조회수: 6126 | 다운로드: 2574 | 주기성 데이터: 3 | 키워드: 등별, 거주자, 분포
- CSV 경상남도_거창군_행정마을별세대인구수**
경상남도 거창군 행정마을별 데이터로 마을별 세대수, 인구(전체), 인구(남), 인구(여) 등의 항목을 제공합니다.

#01 머신러닝 이론 데이터 전처리

데이터 전처리

데이터를 정리하고 분석하기 쉽게 분리하는 단계

#01 머신러닝 이론 데이터 전처리

결측값(비어있는 값) 처리

이상치(정상 범위를 벗어난 값) 처리

그룹화

정규화

표준화

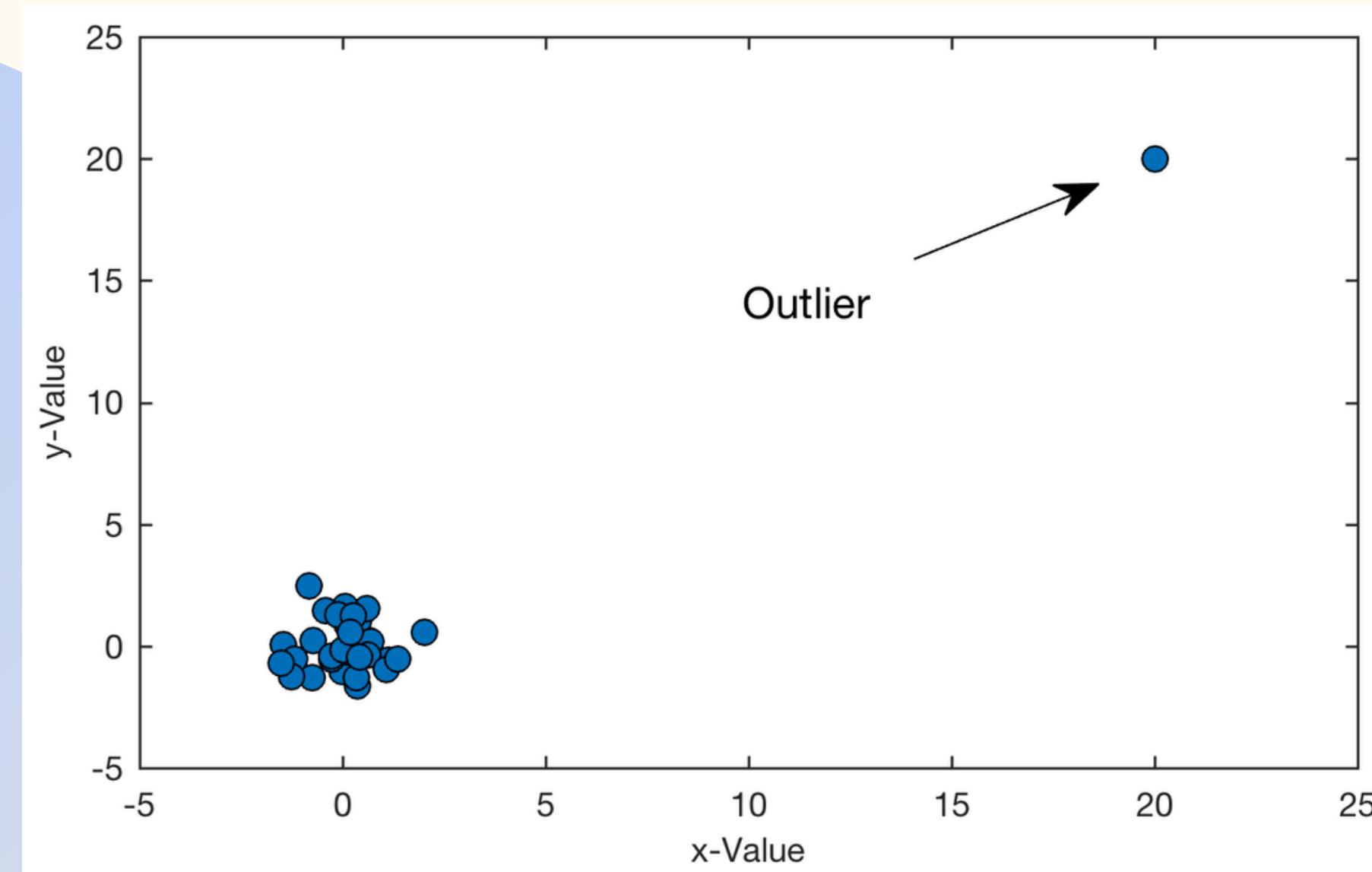
등

#01 머신러닝 이론 데이터 전처리

| Respondent | Variables | | | Missing values replaced by means | | |
|------------|-----------|----|----|-------------------------------------|----|----|
| | A | B | C | A | B | C |
| 1 | 2 | 6 | | 2 | 6 | 8 |
| 2 | | 6 | 2 | 8 | 6 | 2 |
| 3 | | 6 | | 8 | 6 | 8 |
| 4 | 10 | 10 | 10 | 10 | 10 | 10 |
| 5 | 10 | 10 | 10 | 10 | 10 | 10 |
| 6 | 10 | 10 | 10 | 10 | 10 | 10 |
| Average | | 8 | 8 | 8 | 8 | 8 |

결측값 처리

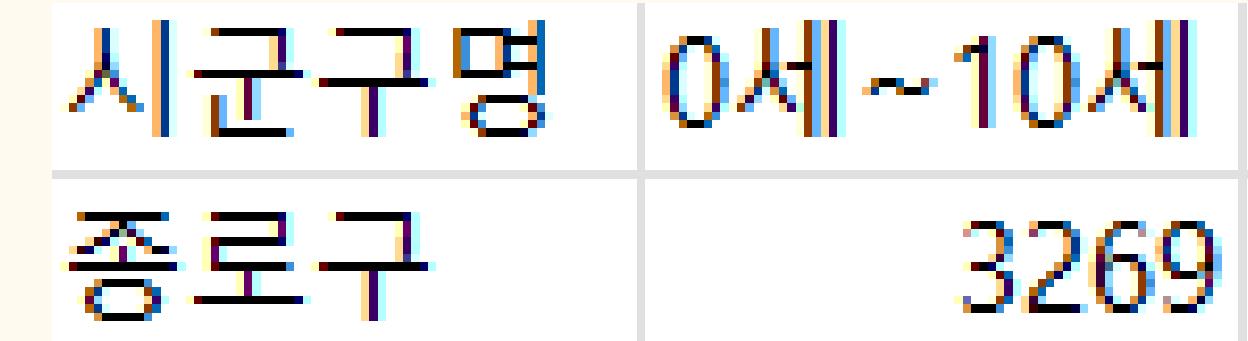
#01 머신러닝 이론 데이터 전처리



이상치 처리

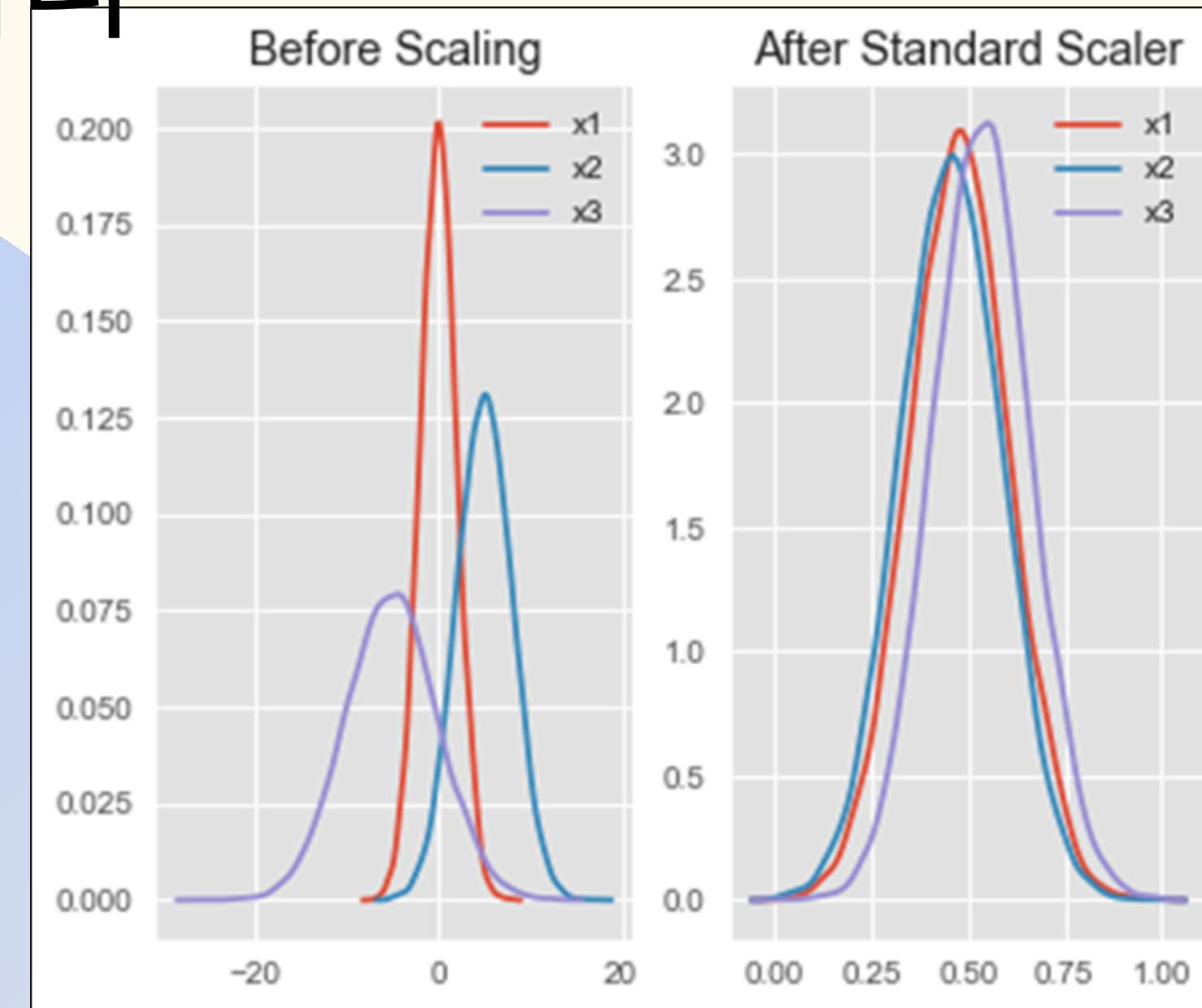
#01 머신러닝 이론 데이터 전처리

| 시군구명 | 읍면동명 | 0세남자 | 1세남자 | 2세남자 | 3세남자 | 4세남자 | 5세남자 | 6세남자 | 7세남자 | 8세남자 | 9세남자 | 10세남자 |
|------|-----------|------|------|------|------|------|------|------|------|------|------|-------|
| 종로구 | 청운효자동 | 23 | 18 | 17 | 24 | 32 | 23 | 31 | 44 | 43 | 40 | 34 |
| 종로구 | 사직동 | 8 | 17 | 21 | 13 | 16 | 25 | 25 | 25 | 35 | 38 | 21 |
| 종로구 | 삼청동 | 1 | 2 | 6 | 3 | 2 | 3 | 4 | 4 | 5 | 6 | 7 |
| 종로구 | 부암동 | 10 | 10 | 20 | 9 | 20 | 8 | 24 | 25 | 35 | 30 | 25 |
| 종로구 | 평창동 | 31 | 35 | 43 | 37 | 51 | 58 | 54 | 57 | 75 | 81 | 78 |
| 종로구 | 무악동 | 11 | 21 | 15 | 26 | 14 | 21 | 30 | 36 | 28 | 31 | 32 |
| 종로구 | 교남동 | 36 | 19 | 35 | 32 | 40 | 36 | 40 | 38 | 46 | 44 | 35 |
| 종로구 | 가화동 | 5 | 7 | 8 | 9 | 8 | 9 | 8 | 14 | 10 | 13 | 7 |
| 종로구 | 종로1.2.3.4 | 8 | 8 | 9 | 5 | 10 | 6 | 4 | 9 | 17 | 4 | 9 |
| 종로구 | 종로5.6가동 | 6 | 1 | 6 | 3 | 9 | 3 | 13 | 12 | 9 | 6 | 4 |
| 종로구 | 이화동 | 5 | 9 | 6 | 5 | 11 | 6 | 7 | 9 | 13 | 12 | 13 |
| 종로구 | 혜화동 | 20 | 26 | 23 | 9 | 25 | 20 | 24 | 23 | 35 | 37 | 45 |
| 종로구 | 창신제1동 | 2 | 9 | 8 | 6 | 7 | 3 | 3 | 2 | 4 | 8 | 6 |
| 종로구 | 창신제2동 | 3 | 5 | 7 | 8 | 7 | 8 | 12 | 10 | 6 | 13 | 10 |
| 종로구 | 창신제3동 | 9 | 12 | 18 | 12 | 18 | 13 | 11 | 22 | 16 | 13 | 20 |
| 종로구 | 승인제1동 | 10 | 5 | 9 | 9 | 7 | 11 | 5 | 11 | 10 | 18 | 15 |
| 종로구 | 승인제2동 | 10 | 11 | 12 | 12 | 7 | 7 | 7 | 12 | 8 | 14 | 8 |



그룹화

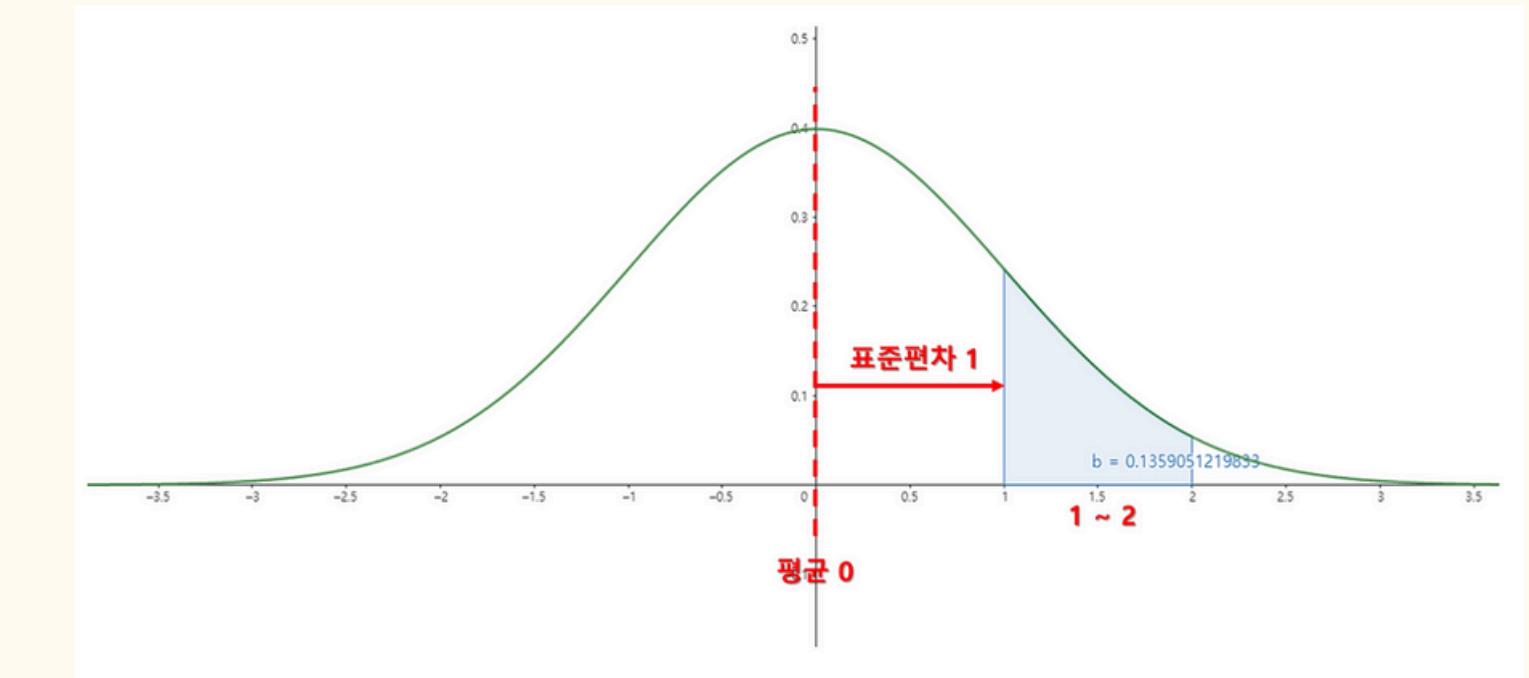
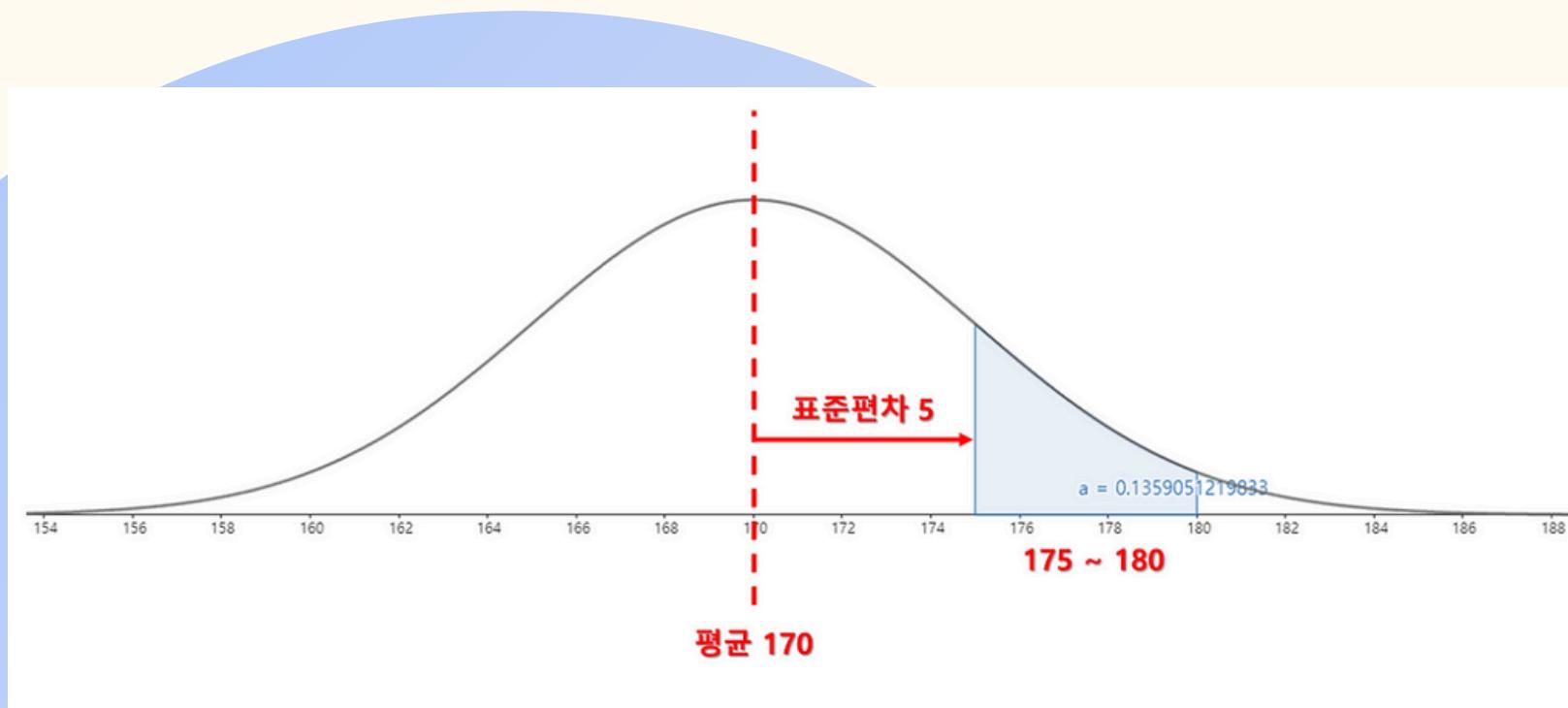
#01 머신러닝 이론 데이터 전처리



정규화

0부터 1 사이의 값으로 치환

#01 머신러닝 이론 데이터 전처리



표준화

평균을 0, 표준편차를 1로 만드는 것

#01 머신러닝 이론 모델 학습

전처리된 데이터를 사용해 머신러닝 모델을 학습시키는 단계

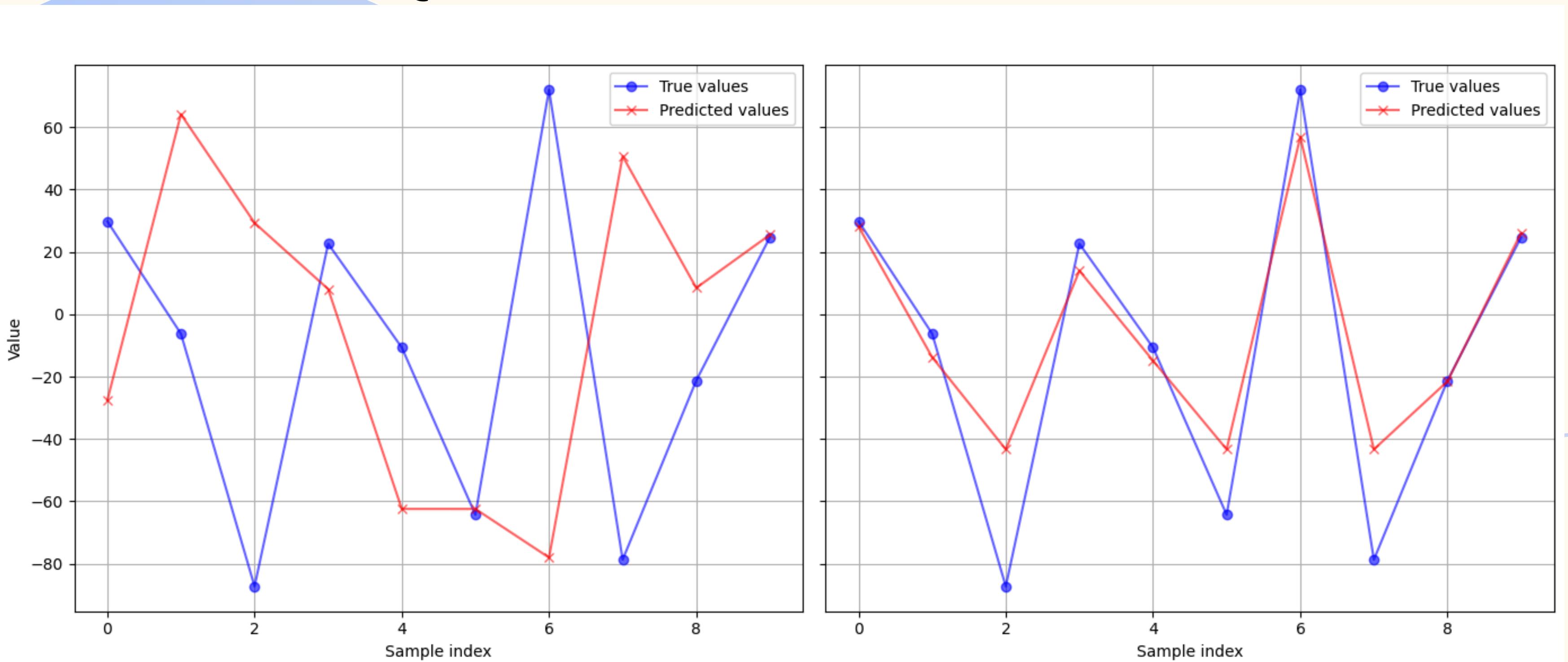
model.fit(X_train, y_train)

#01 머신러닝 이론 모델 평가

학습된 모델의 성능을 테스트 데이터를 사용해 평가하는 단계

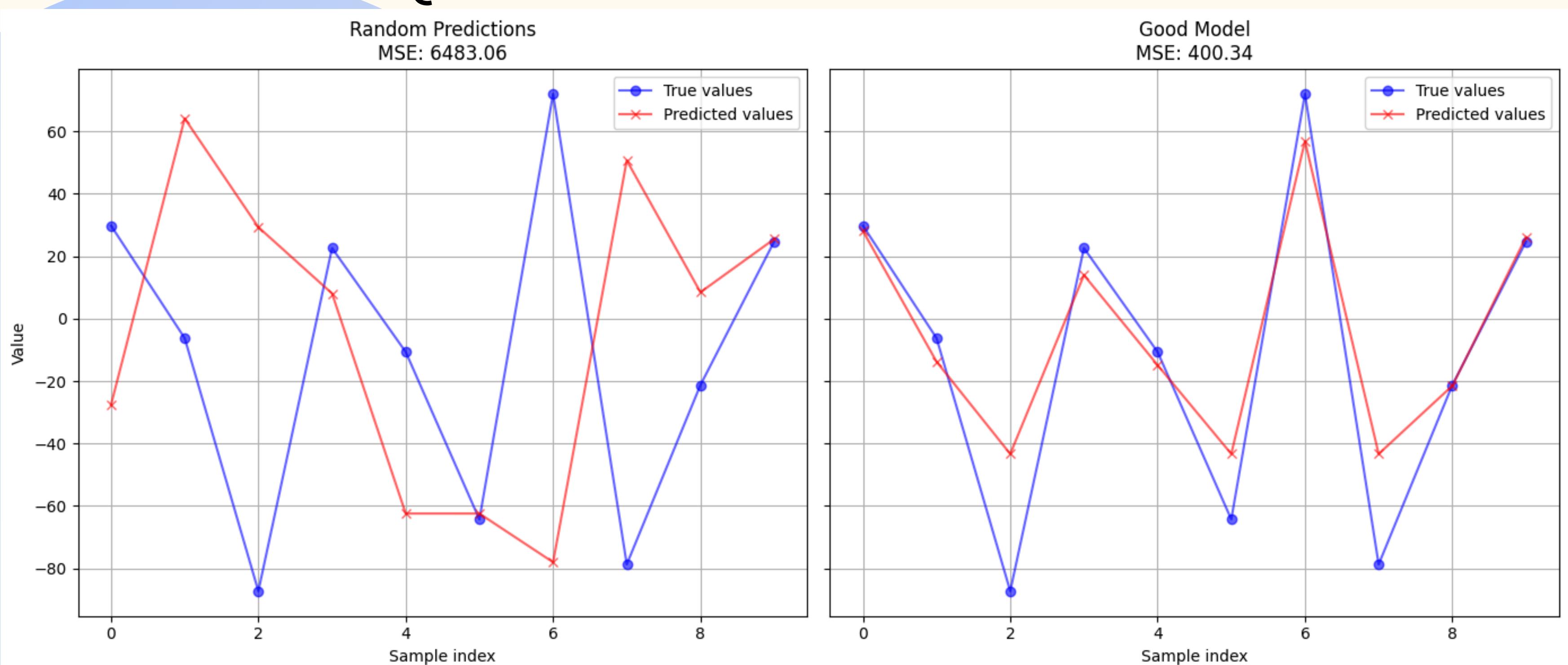
#01 머신러닝 이론 모델 평가

Q. 어떤게 더 잘 학습된 모델일까?



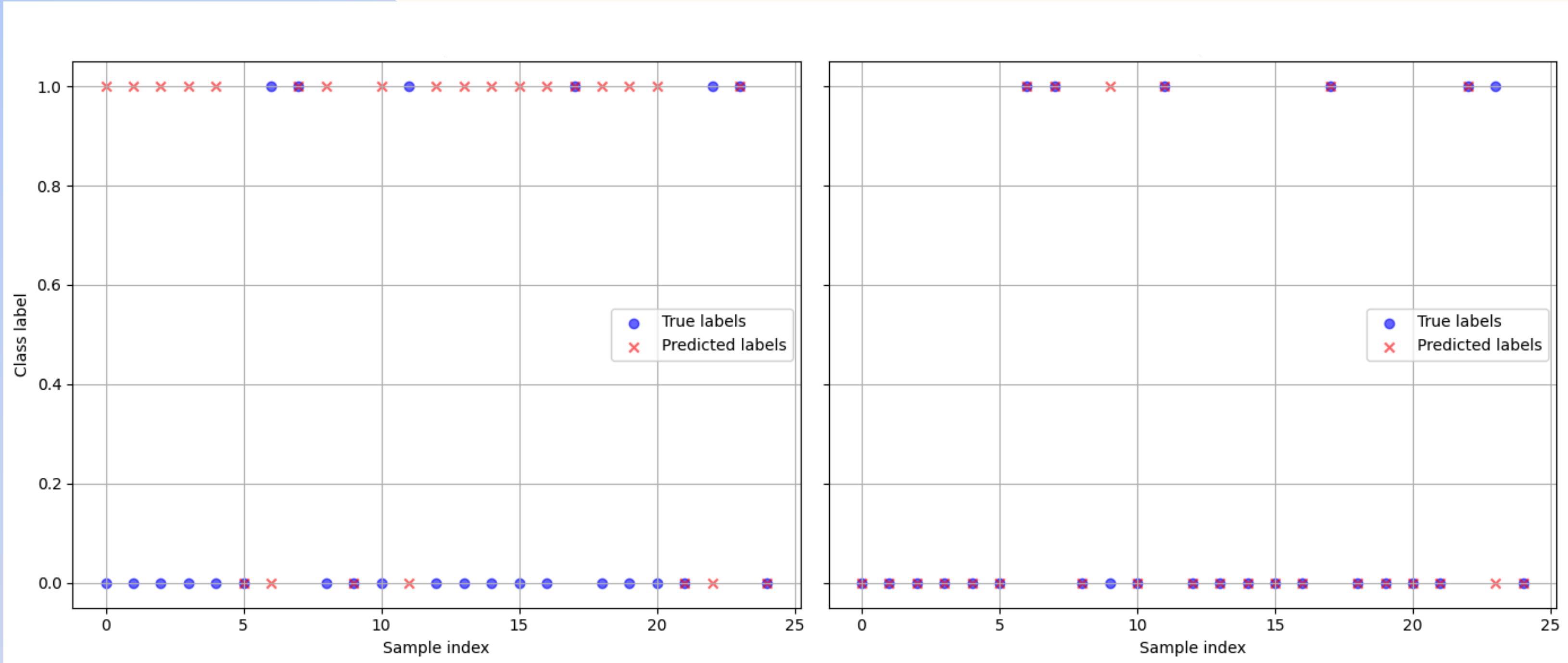
#01 머신러닝 이론 모델 평가

Q. 어떤게 더 잘 학습된 모델일까?



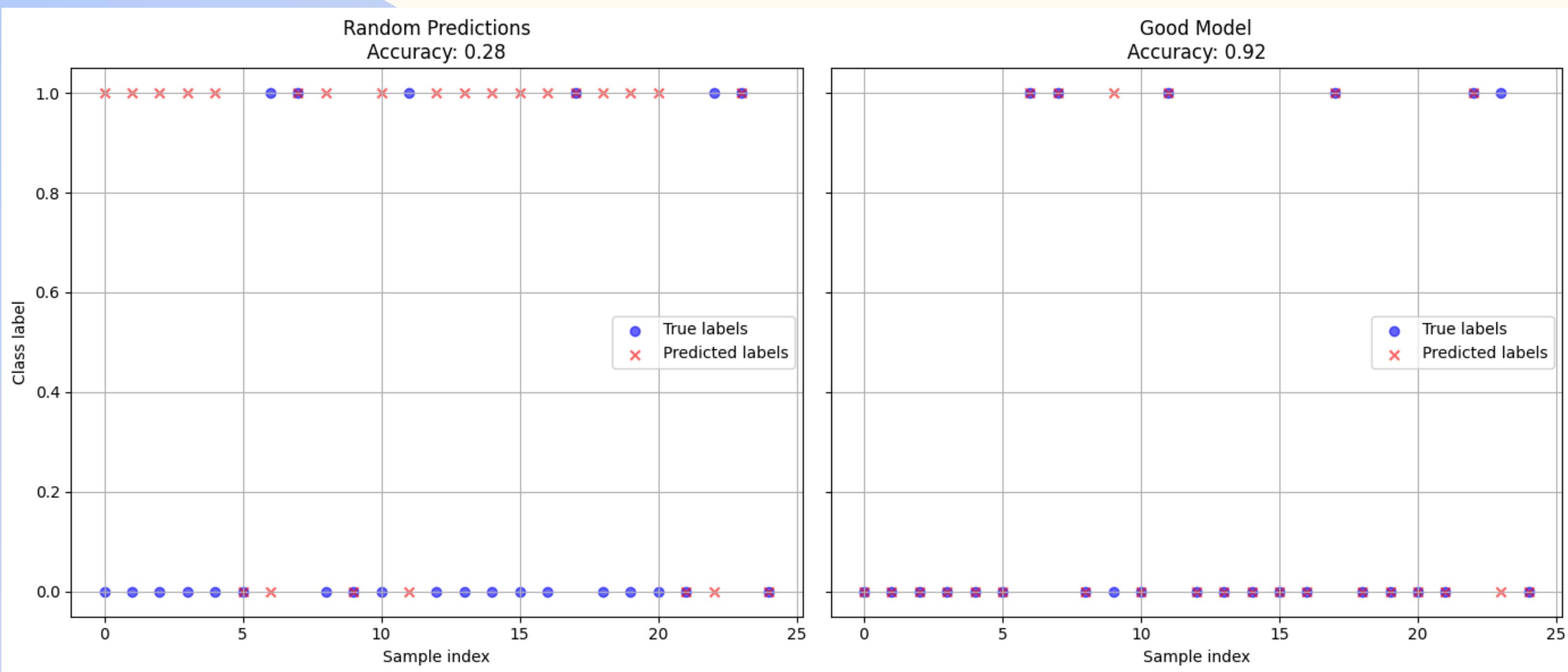
#01 머신러닝 이론 모델 평가

Q. 어떤게 더 잘 학습된 모델일까?



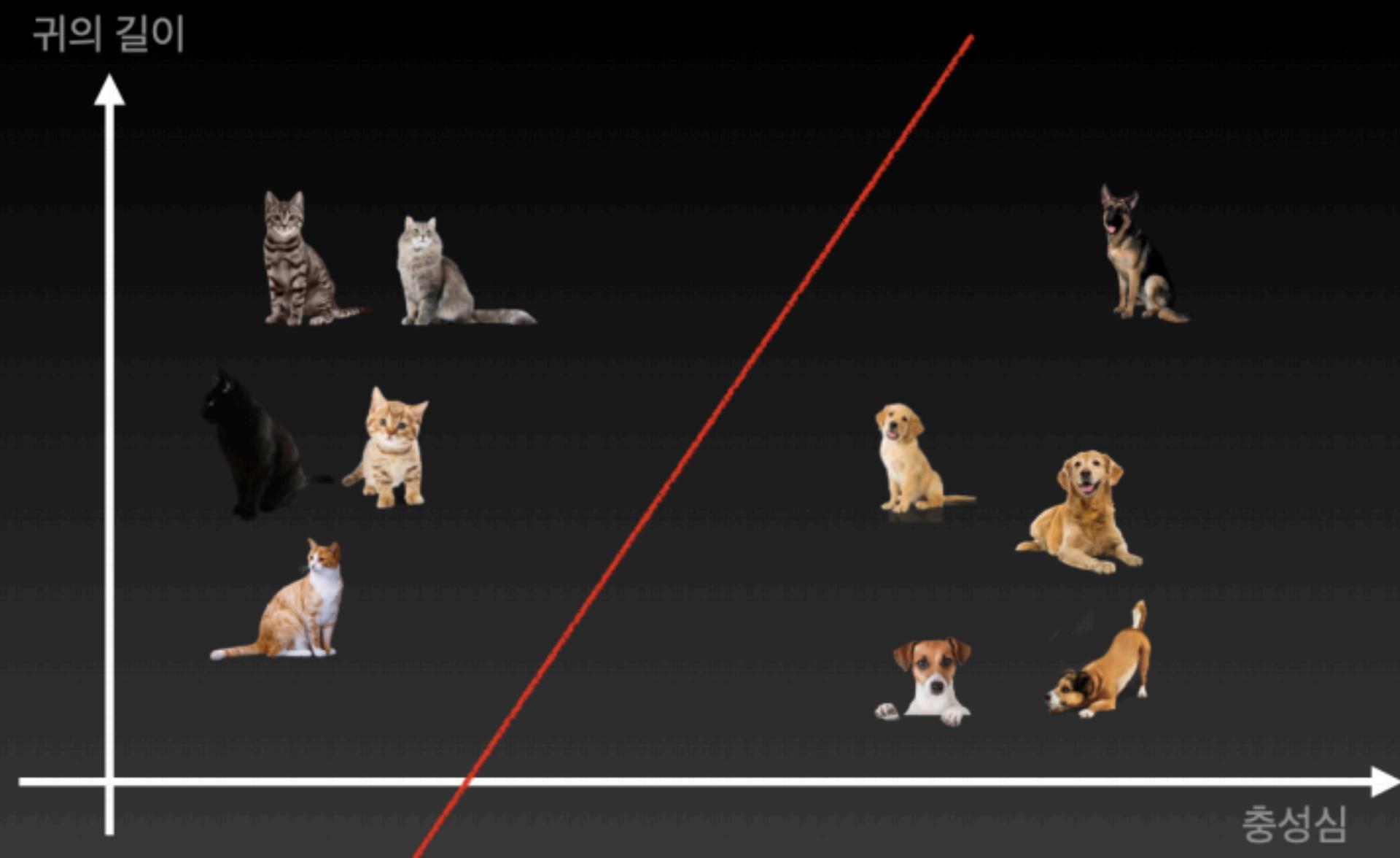
#01 머신러닝 이론 모델 평가

Q. 어떤게 더 잘 학습된 모델일까?



#01 머신러닝 이론 모델 사용

완성된 모델을 실제 문제 해결에 사용하는 단계



#01 머신러닝 이론 예시

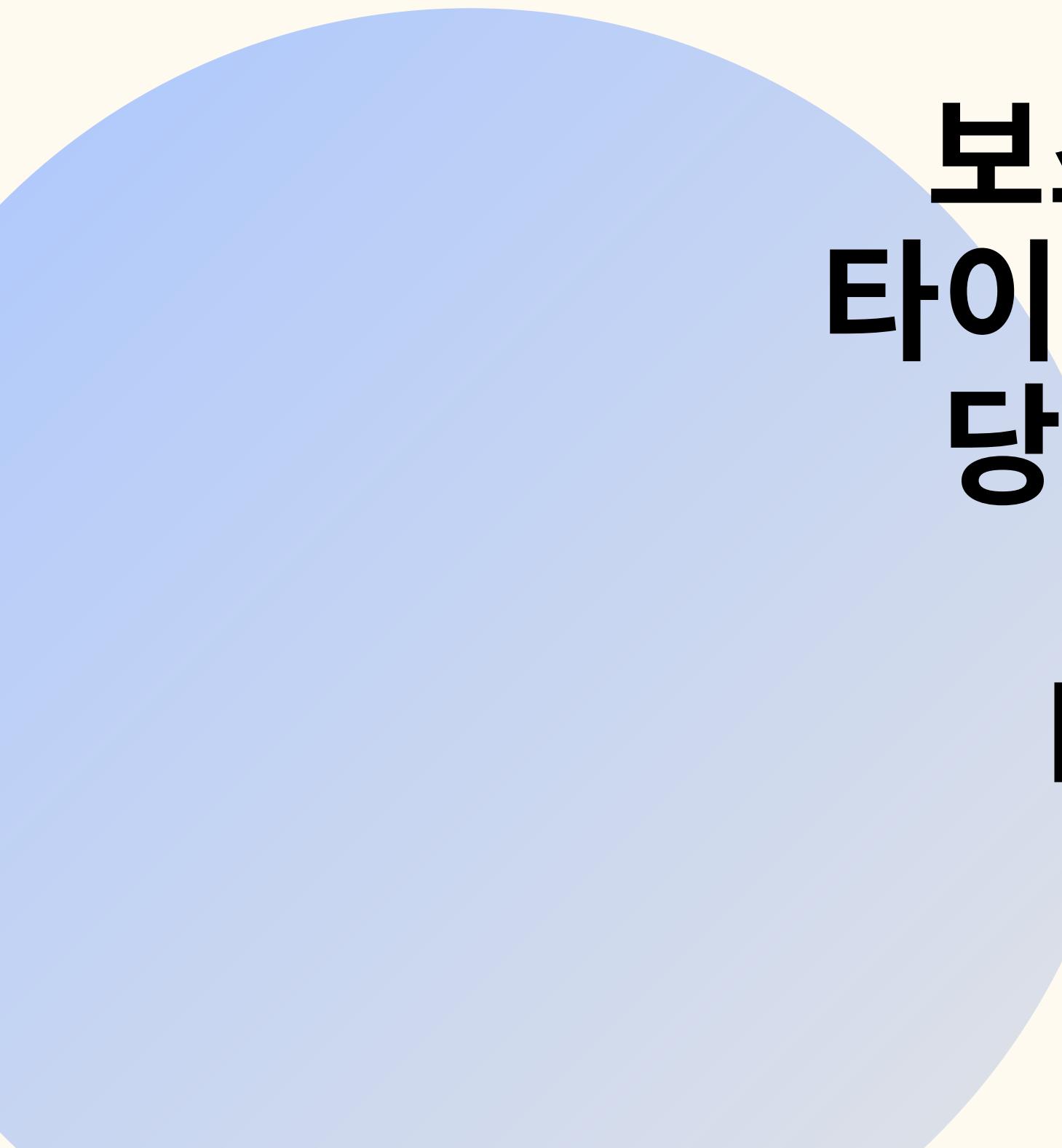
집 값 예측

피쳐 - 집의 크기, 방의 개수, 위치 등
레이블 - 집 값

목표 - 피쳐들을 바탕으로 집값을 예측하는 모델 만들기

#01 머신러닝 이론

대표 머신러닝 프로젝트



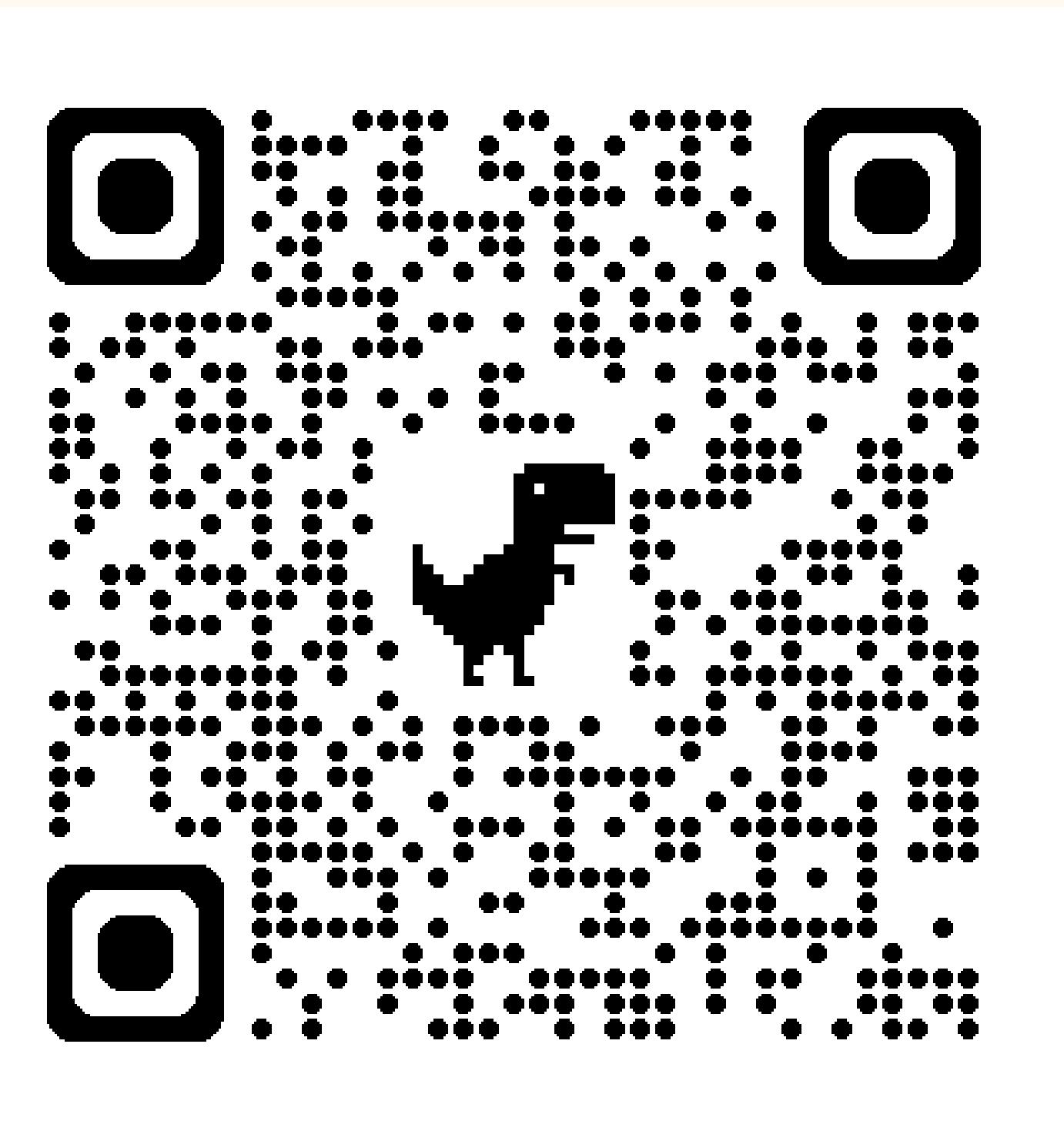
보스턴 집 값 예측
타이타닉 생존자 예측
당뇨병 환자 예측

kaggle.com
dacon.io

#01 머신러닝 이론 동아리 소개

인스타 아이디 dat.f_

대한민국 저출산 해결방안 제안
연말 인구 밀집 분석 및 해결방안 제안
마포구 불법 주정차 데이터를 활용한 주차요금 조정 제안
노인을 위한 사회복지시설 최적입지 선정
지하철역 혼잡도 예측 및 혼잡도 낮은 역 추천시스템 구축



#01 머신러닝 이론



머신러닝 이론
Q & A



#03 데이터전처리

-실습데이터 불러오기

- Google Colab에 실습 데이터 불러오기

The screenshot shows a Jupyter Notebook cell with the following code:

```
[29] import pandas as pd  
import numpy as np  
  
from google.colab import files  
  
# 파일 업로드  
uploaded = files.upload()
```

Below the code, there is a file upload dialog with a red box around the "파일 선택" button. The dialog also shows "선택된 파일 없음" and "Cancel upload".

Below the dialog, a file browser window is open, showing the file "high_diamond_ranked_10min.csv" in the "다운로드 > high_diamond_ranked_10min" folder. A mouse cursor is pointing at the file entry. The file details are shown in a tooltip:

- 이름: high_diamond_ranked_10min
- 날짜: 2024-06-07 오후 1:02
- 유형: Microsoft Excel 스크립트로 구분된 값 파일
- 크기: 1.37MB
- 수정한 날짜: 2024-06-07 오후 1:02

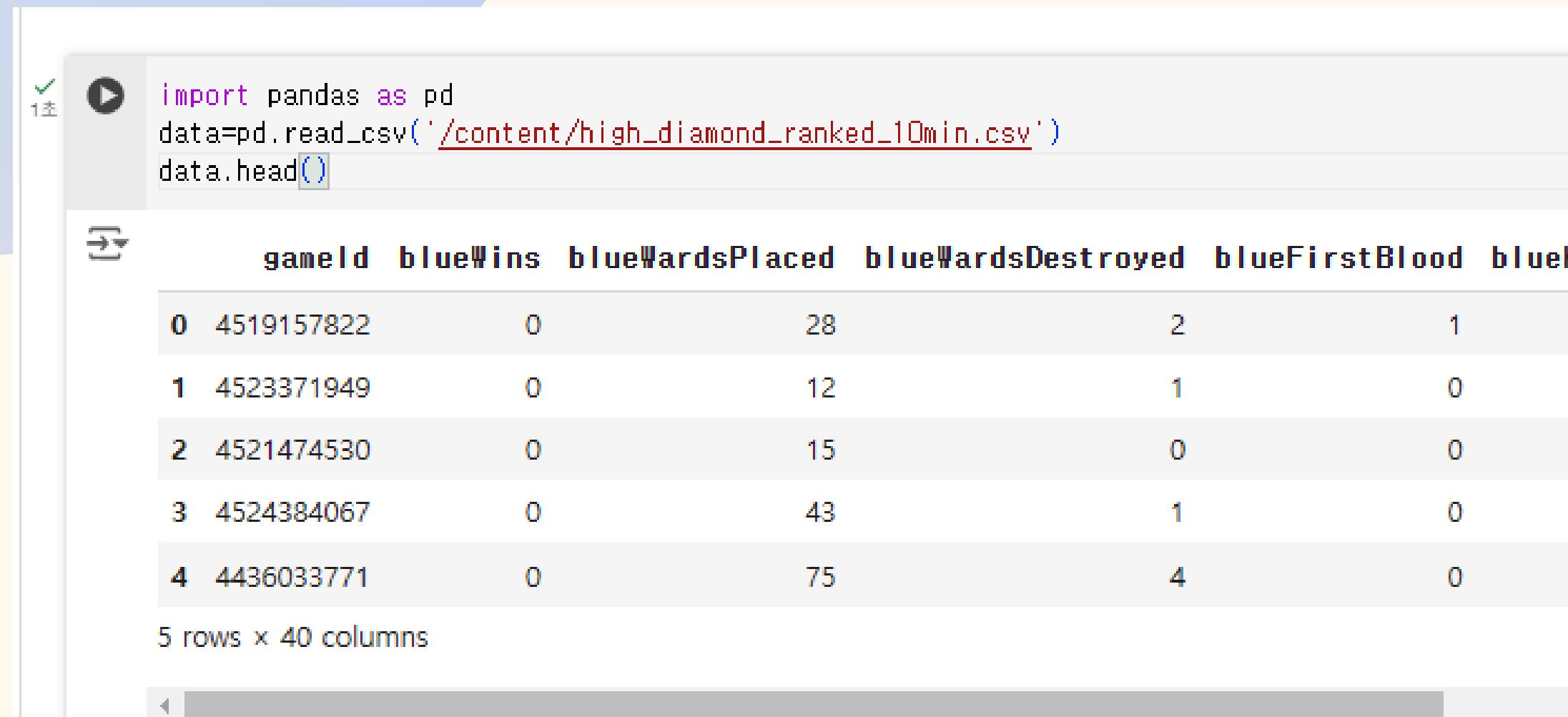
#03 데이터전처리

- 실습데이터 불러오기

pandas 패키지 불러오기

pd.read_csv()함수를 통해 데이터를 불러온 후 'data'에 저장하기

head()함수를 사용하여 데이터의 앞부분을 출력하기



The screenshot shows a Jupyter Notebook cell with the following code:

```
1초 ⏴ import pandas as pd  
data=pd.read_csv('/content/high_diamond_ranked_10min.csv')  
data.head()
```

Below the code, the output is displayed as a Pandas DataFrame:

| | gameId | blueWins | blueWardsPlaced | blueWardsDestroyed | blueFirstBlood | blueKills |
|---|------------|----------|-----------------|--------------------|----------------|-----------|
| 0 | 4519157822 | 0 | 28 | 2 | 1 | 1 |
| 1 | 4523371949 | 0 | 12 | 1 | 0 | 0 |
| 2 | 4521474530 | 0 | 15 | 0 | 0 | 0 |
| 3 | 4524384067 | 0 | 43 | 1 | 0 | 0 |
| 4 | 4436033771 | 0 | 75 | 4 | 0 | 0 |

5 rows × 40 columns

#03 데이터전처리

- info()함수
-> 행과 열의 수, 컬럼명, 컬럼의 데이터타입 등 데이터에 대한 전반적인 정보를 나타내줌

```
0초  data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9879 entries, 0 to 9878
Data columns (total 40 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   gameId          9879 non-null    int64  
 1   blueWins         9879 non-null    int64  
 2   blueWardsPlaced 9879 non-null    int64  
 3   blueWardsDestroyed 9879 non-null    int64  
 4   blueFirstBlood   9879 non-null    int64  
 5   blueKills        9879 non-null    int64  
 6   blueDeaths       9879 non-null    int64  
 7   blueAssists      9879 non-null    int64  
 8   blueEliteMonsters 9879 non-null    int64  
 9   blueDragons       9879 non-null    int64  
 10  blueHeralds      9879 non-null    int64  
 11  blueTowersDestroyed 9879 non-null    int64  
 12  blueTotalGold    9879 non-null    int64  
 13  blueAvgLevel     9879 non-null    float64
```

#03 데이터전처리

- 불필요한 컬럼 제거하기

[‘컬럼명’, ‘컬럼명’]의 형식을 통해 불필요한 컬럼의 집합을 cols에 저장하기

drop함수를 사용하여 cols에 저장된 열들을 제거한 후 data_drop에 저장하기

```
cols = ['gameId', 'redFirstBlood', 'redKills', 'redEliteMonsters', 'redDragons', 'redTotalMinionsKilled',  
'redTotalJungleMinionsKilled', 'redGoldDiff', 'redExperienceDiff', 'redCSPerMin', 'redGoldPerMin', 'redHeralds',  
'blueGoldDiff', 'blueExperienceDiff', 'blueCSPerMin', 'blueGoldPerMin', 'blueTotalMinionsKilled']  
  
data_drop = data.drop(cols, axis = 1)  
data_drop.head()
```

| | blueWins | blueWardsPlaced | blueWardsDestroyed | blueFirstBlood | blueKills | blueDeaths | blueAssists | blueEliteMo |
|---|----------|-----------------|--------------------|----------------|-----------|------------|-------------|-------------|
| 0 | 0 | 28 | 2 | 1 | 9 | 6 | 11 | |
| 1 | 0 | 12 | 1 | 0 | 5 | 5 | 5 | |
| 2 | 0 | 15 | 0 | 0 | 7 | 11 | 4 | |
| 3 | 0 | 43 | 1 | 0 | 4 | 5 | 5 | |
| 4 | 0 | 75 | 4 | 0 | 6 | 6 | 6 | |

5 rows × 23 columns

*drop함수 사용 형태 -> data.drop(['컬럼명','컬럼명'],axis=1)

#03 데이터전처리

- blue team의 특징 시각화하기

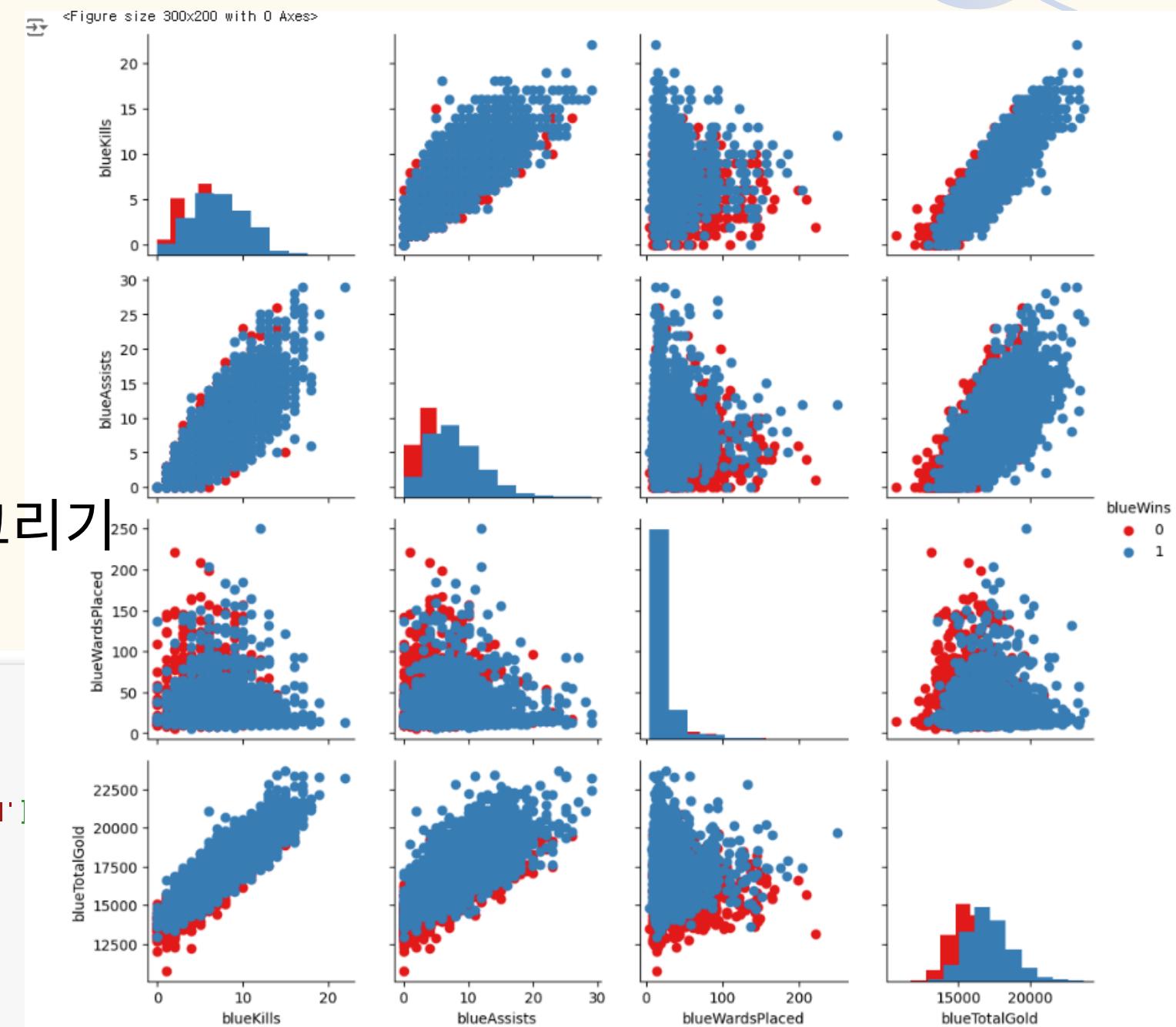
시각화 패키지 seaborn, matplotlib.pyplot 불러오기

seborn 패키지의 PairGrid()함수를 활용하여
각 변수에 따른 블루팀 승패여부의 pair그래프 그리기

대각선은 히스토그램으로 대각선을 제외한 영역은 산점도로 그리기

```
import seaborn as sns
import matplotlib.pyplot as plt

graph = sns.PairGrid(data=data_drop, vars=['blueKills', 'blueAssists', 'blueWardsPlaced', 'blueTotalGold'],
                      hue='blueWins', palette='Set1')
graph.map_diag(plt.hist)
graph.map_offdiag(plt.scatter)
graph.add_legend()
plt.figure(figsize=(3, 2))
```

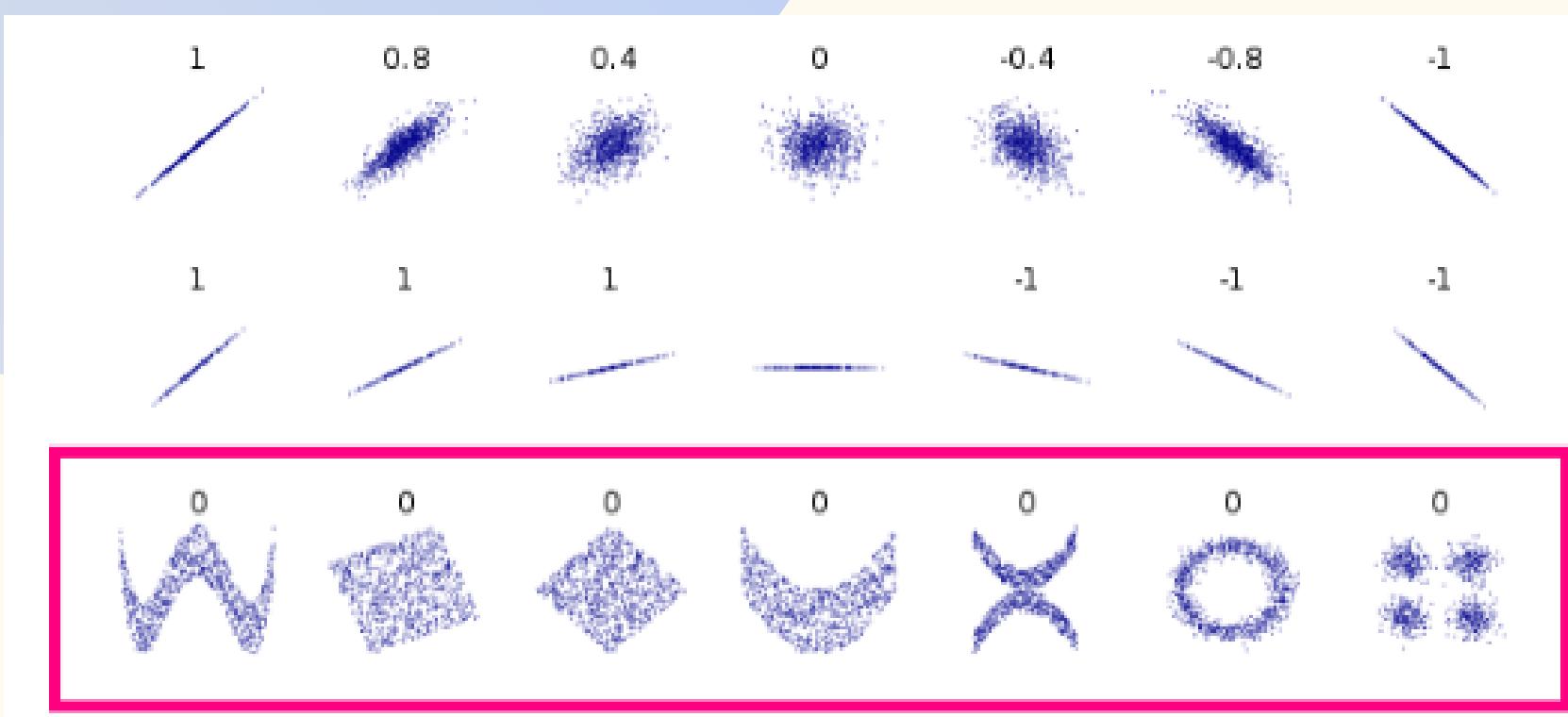


*PairGrid()함수-> 모든 열의 조합을 만들어주는 함수로 sns.PairGrid(data=데이터명, vars=변수집합,hue=명목형변수)의 형식을 가짐

#03 데이터전처리

- 공분산과 상관계수

비례 관계인 경우, 공분산은 양수의 값
반비례 관계인 경우, 공분산은 음수의 값



$$\text{피어슨 상관계수 } (r) = \frac{\text{공분산}}{\text{표준편차} \times \text{표준편차}}$$

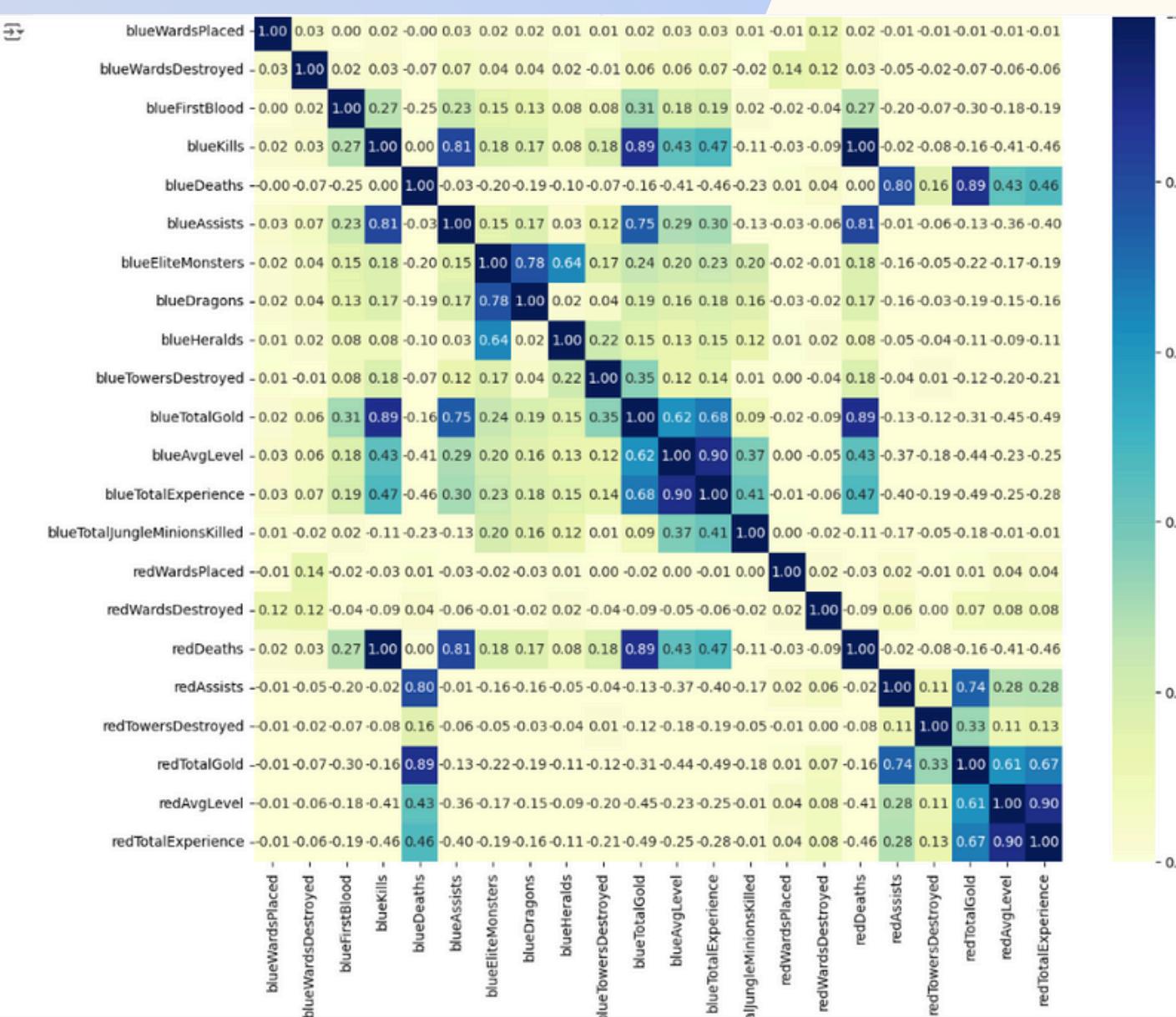
$$r_{XY} = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\frac{\sum_{i=1}^n (X_i - \bar{X})^2}{n-1}} \sqrt{\frac{\sum_{i=1}^n (Y_i - \bar{Y})^2}{n-1}}}$$

$$r_{XY} = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}$$

#03 데이터전처리

- 상관계수

```
[15] plt.figure(figsize=(16, 12))
    sns.heatmap(data_drop.drop('blueWins', axis=1).corr(), cmap='YIGnBu', annot=True, fmt='.2f', vmin=0);
```



seaborn 패키지의 heatmap함수를 사용하여
변수간의 상관관계 heatmap 그리기

모델을 돌리기 전에 상관계수가
높은 변수들은 제외하는 것이 좋음

*heatmap() 함수 -> heatmap(데이터명, corr(), cmap=컬러맵, annot=값 표시)

상관계수 히트맵을 그리겠다는 뜻

#03 데이터전처리

- 상관관계가 높은 변수들 제거

drop함수 사용하여 상관관계가 높은 변수들을 제거하고 최종 데이터 추출

```
✓ 0초  cols = ['blueAvgLevel', 'redWardsPlaced', 'redWardsDestroyed', 'redDeaths', 'redAssists', 'redTowersDestroyed',  
             'redTotalExperience', 'redTotalGold', 'redAvgLevel']  
    data_drop = data_drop.drop(cols, axis=1)  
    data_drop.head()  
  
→   blueWins  blueWardsPlaced  blueWardsDestroyed  blueFirstBlood  blueKills  blueDeaths  blueAssists  blueE  
  0      0            28                  2              1            9            6           11  
  1      0            12                  1              0            5            5           5  
  2      0            15                  0              0            7           11           4  
  3      0            43                  1              0            4            5           5  
  4      0            75                  4              0            6            6           6
```

#03 데이터처리

- 반복문 for

```
numbers = [1, 2, 3, 4, 5]
total = 0

for number in numbers:
    total += number

print(total)
```

정확히 10번, 100번 등 횟수로
반복하고 싶을 때

#03 데이터전처리

- 반복문 for

```
for i in range(1, 10):  
    print(2, 'x', i, '=', 2*i)
```

#03 데이터전처리

- 상관관계가 낮은 열들을 포함

```
if (corr_list[col]>0.2 or corr_list[col]<-0.2):
```

루프를 사용하여 상관관계가 0.2보다 크거나 -0.2보다 작은 열
(절대값 기준으로 0.2보다 작은) 이름을 cols 리스트에 추가

```
[12] corr_list = data_drop[data_drop.columns[1:]].apply(lambda x: x.corr(data_drop['blueWins']))
cols = []
for col in corr_list.index:
    if (corr_list[col]>0.2 or corr_list[col]<-0.2):
        cols.append(col)
cols
```

```
['blueFirstBlood',
 'blueKills',
 'blueDeaths',
 'blueAssists',
 'blueEliteMonsters',
 'blueDragons',
 'blueTotalGold',
 'blueTotalExperience']
```

```
[13] data_drop = data_drop[cols]
data_drop.head()
```

| | blueFirstBlood | blueKills | blueDeaths | blueAssists | blueEliteMonsters | blueDragons | blueTotalGold | blueTotalExperience |
|---|----------------|-----------|------------|-------------|-------------------|-------------|---------------|---------------------|
| 0 | 1 | 9 | 6 | 11 | 0 | 0 | 17210 | 17039 |
| 1 | 0 | 5 | 5 | 5 | 0 | 0 | 14712 | 16265 |
| 2 | 0 | 7 | 11 | 4 | 1 | 1 | 16113 | 16221 |
| 3 | 0 | 4 | 5 | 5 | 1 | 0 | 15157 | 17954 |
| 4 | 0 | 6 | 6 | 6 | 0 | 0 | 16400 | 18543 |

#03 데이터전처리

- 히스토그램 생성

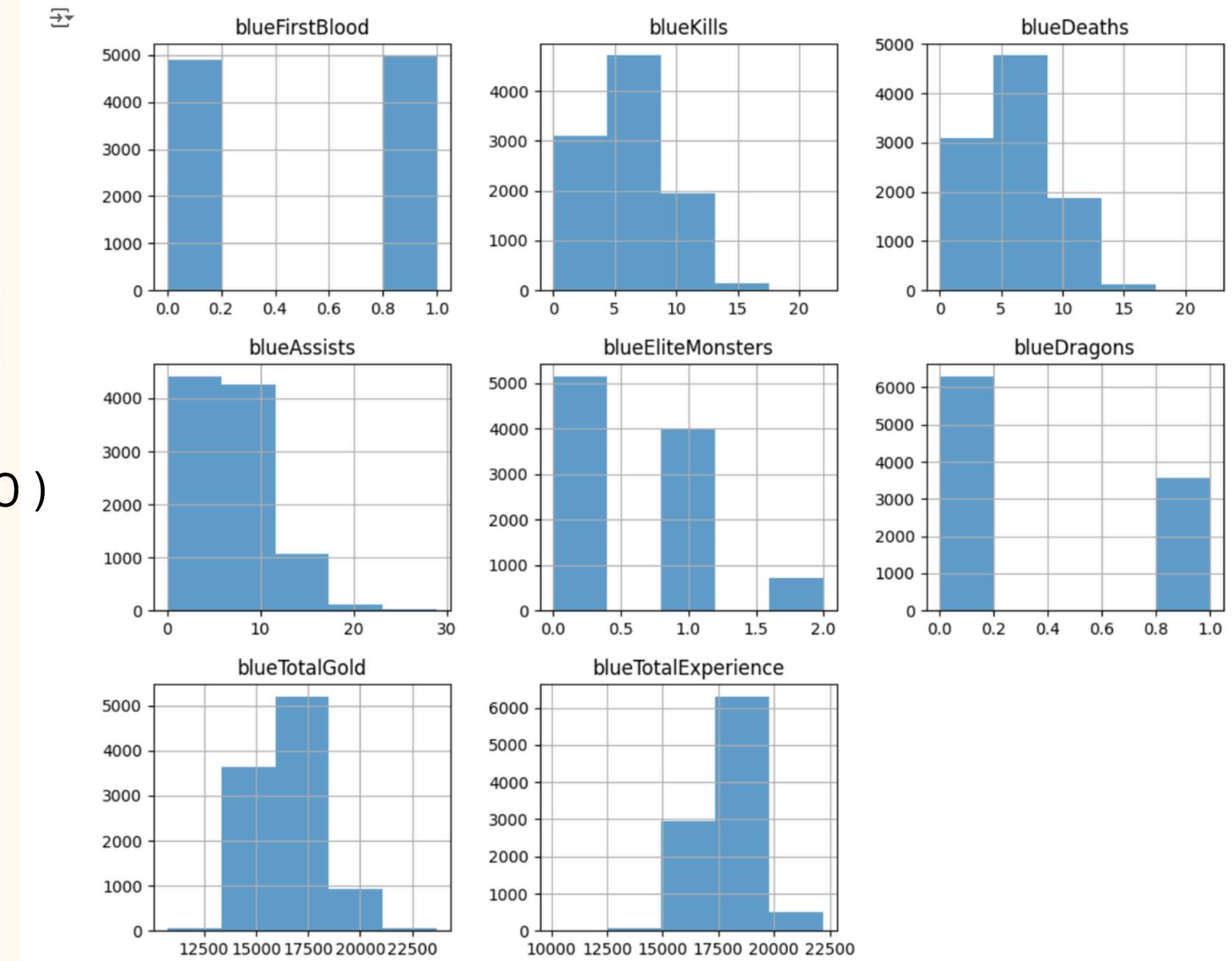
시각화 패키지 matplotlib.pyplot 불러오기

```
[14] data_drop.hist(alpha = 0.7, figsize=(12,10), bins=5);
```

‘alpha’ 매개변수는 히스토그램의 불투명도 설정 (0.0 ~1.0)

‘figsize’ 매개변수는 전체 그림의 크기 설정 (너비, 높이)

‘bins’ 매개변수는 각 히스토그램의 막대 개수 설정



데이터 분할

✓ [11] # 데이터셋 전처리, 훈련세트와 테스트 세트로 분할
0초

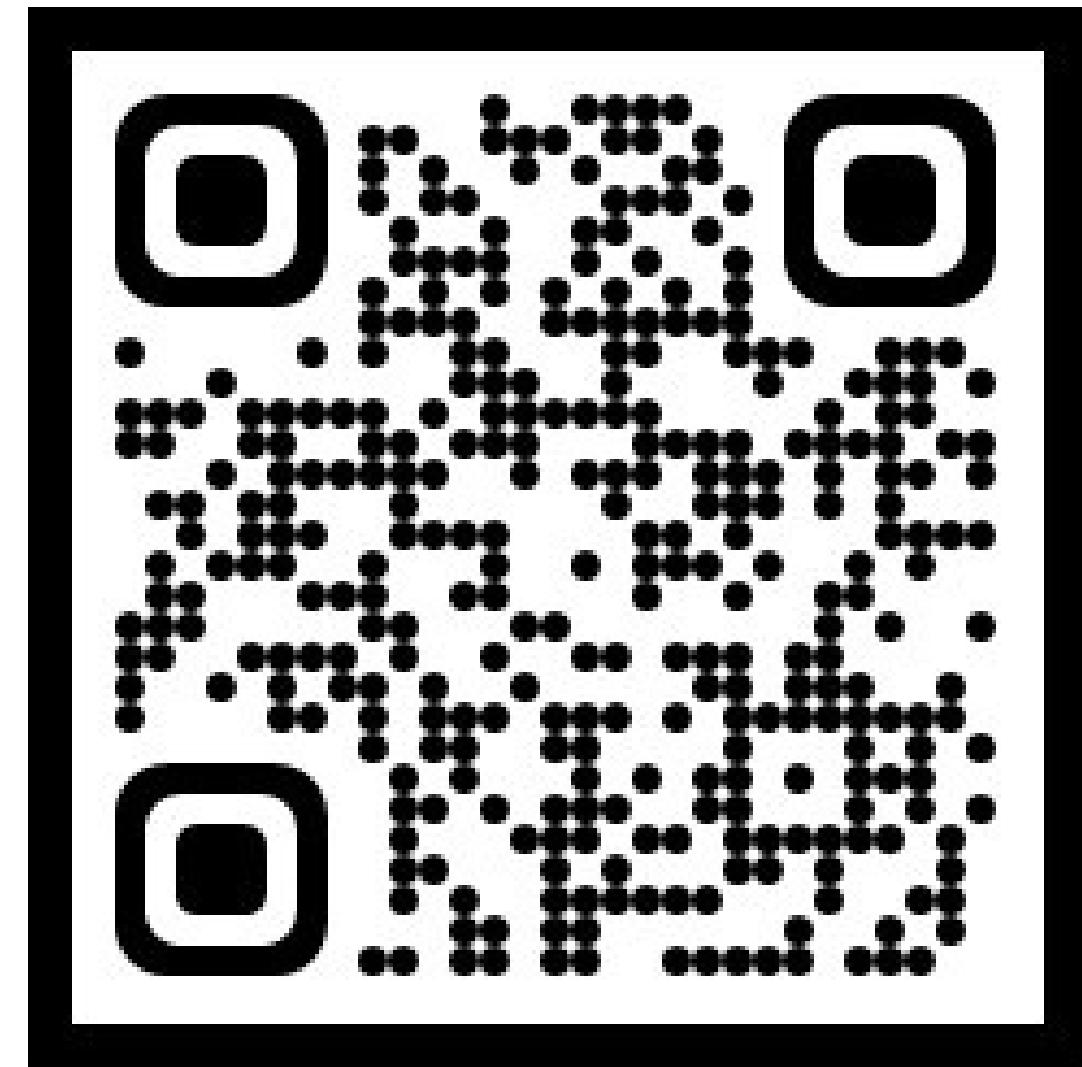
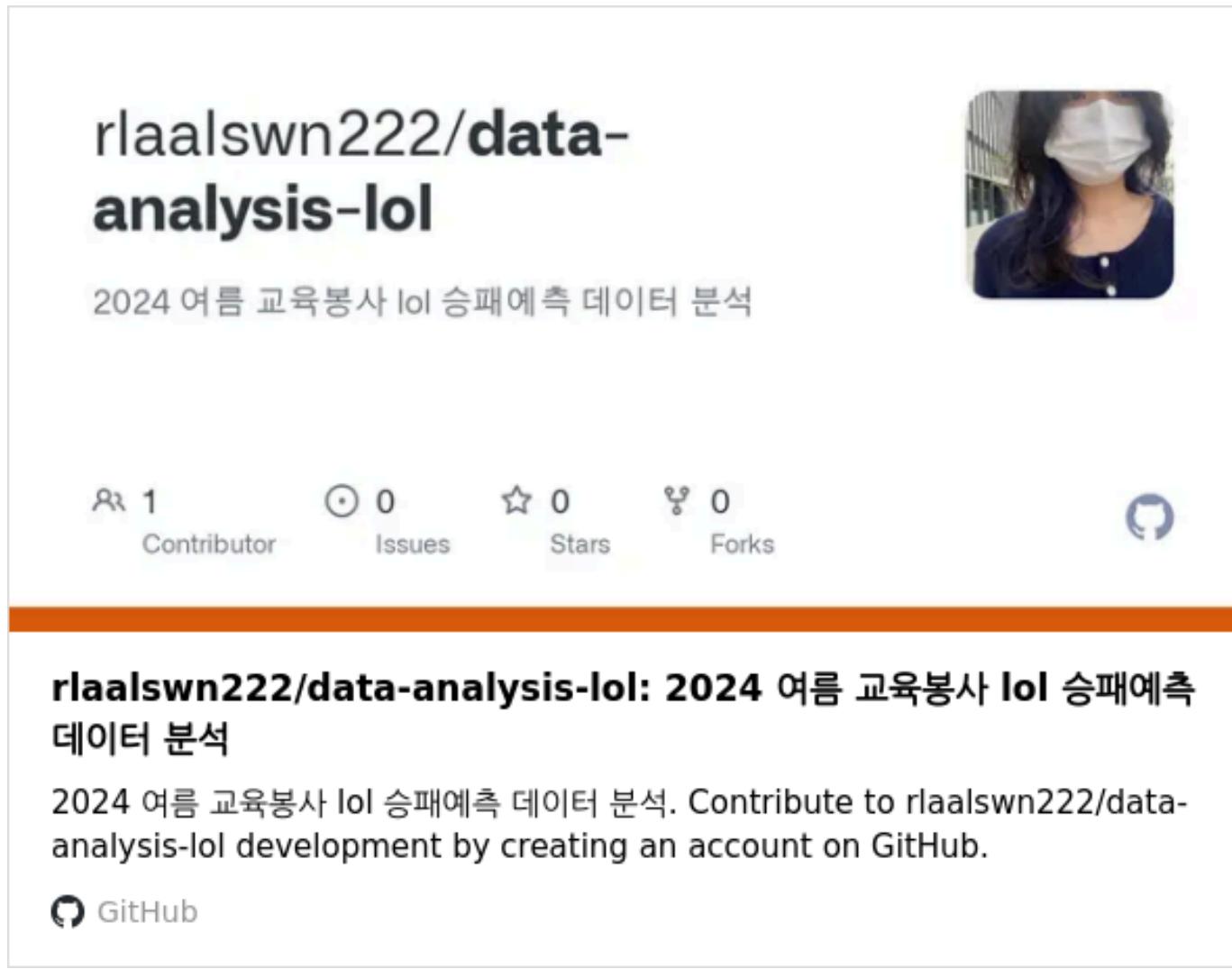
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
X = data_drop
y = data['blueWins']

#데이터 스케일링
scaler = MinMaxScaler() #MinMaxScaler를 사용하여 데이터의 스케일 조정
scaler.fit(X) #스케일러에 데이터를 맞춤
X = scaler.transform(X) #데이터를 스케일링

#데이터 분할
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

동영상 시청





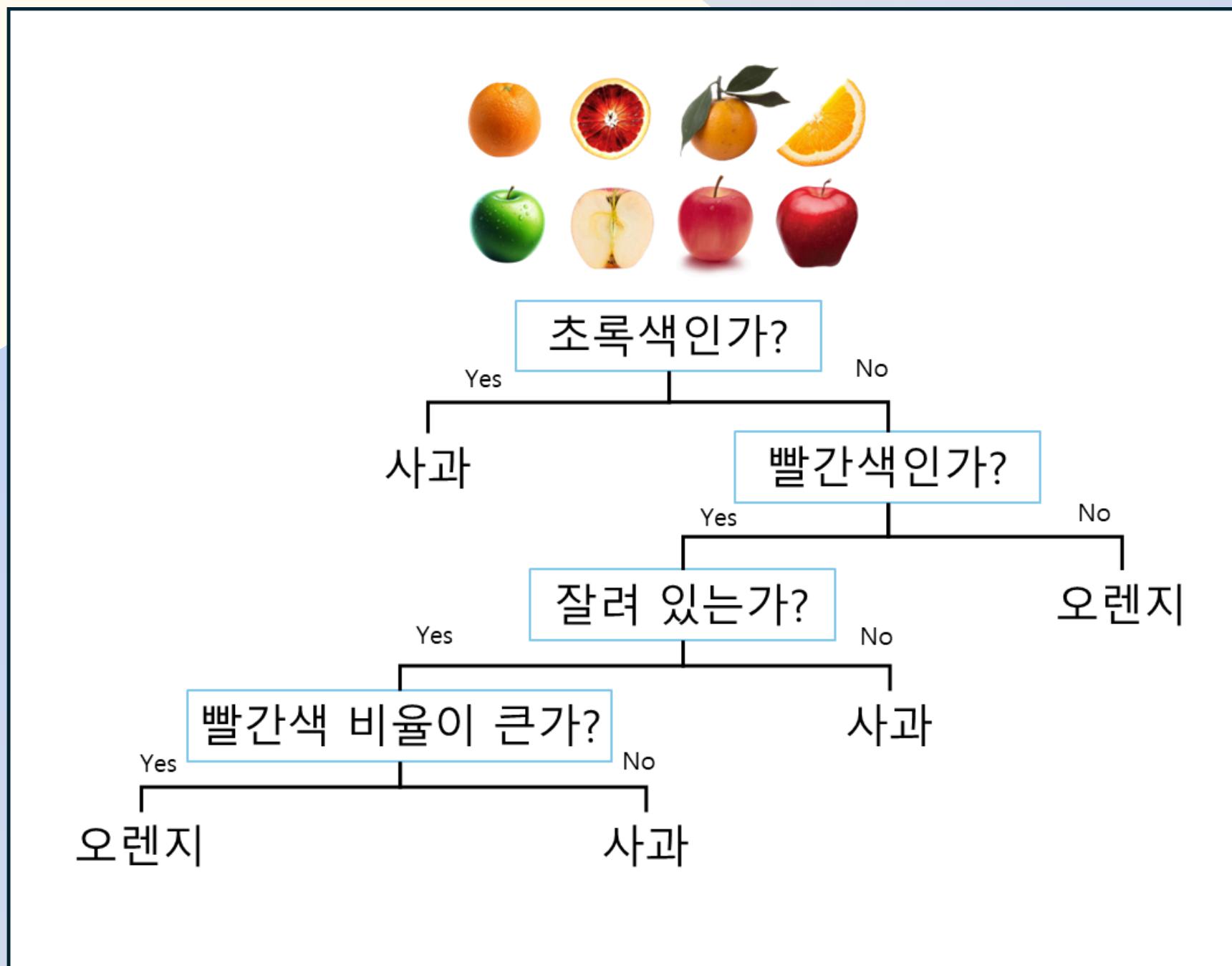
<https://github.com/reriaalsw222/data-analysis-lol>

Random Forest

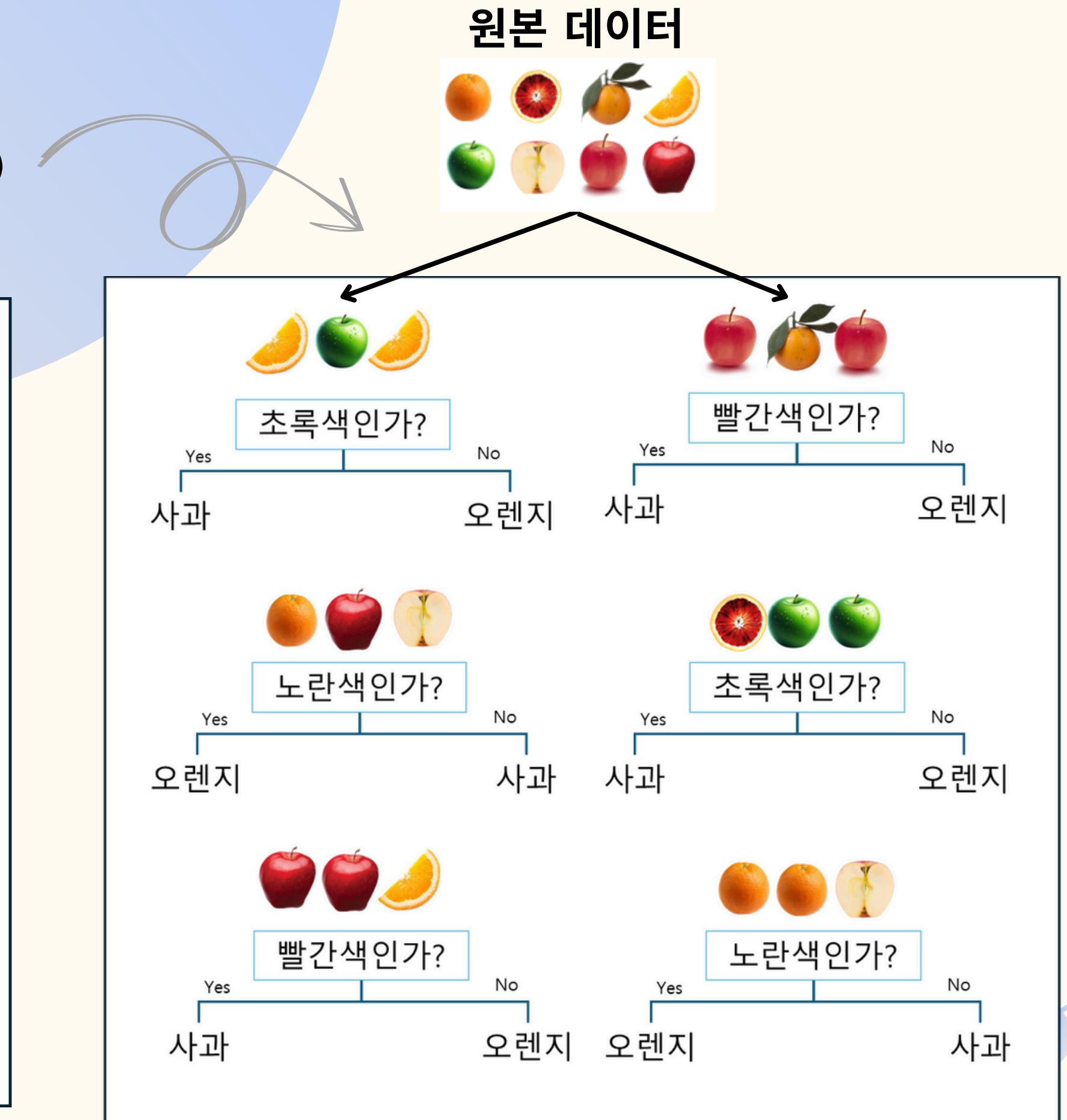


각기 다른 형태의 과일을 학습 시키려면 어떻게 해야할까?

데이터 샘플링 (부트스트랩)



Decision Tree



Random Forest

원본 데이터

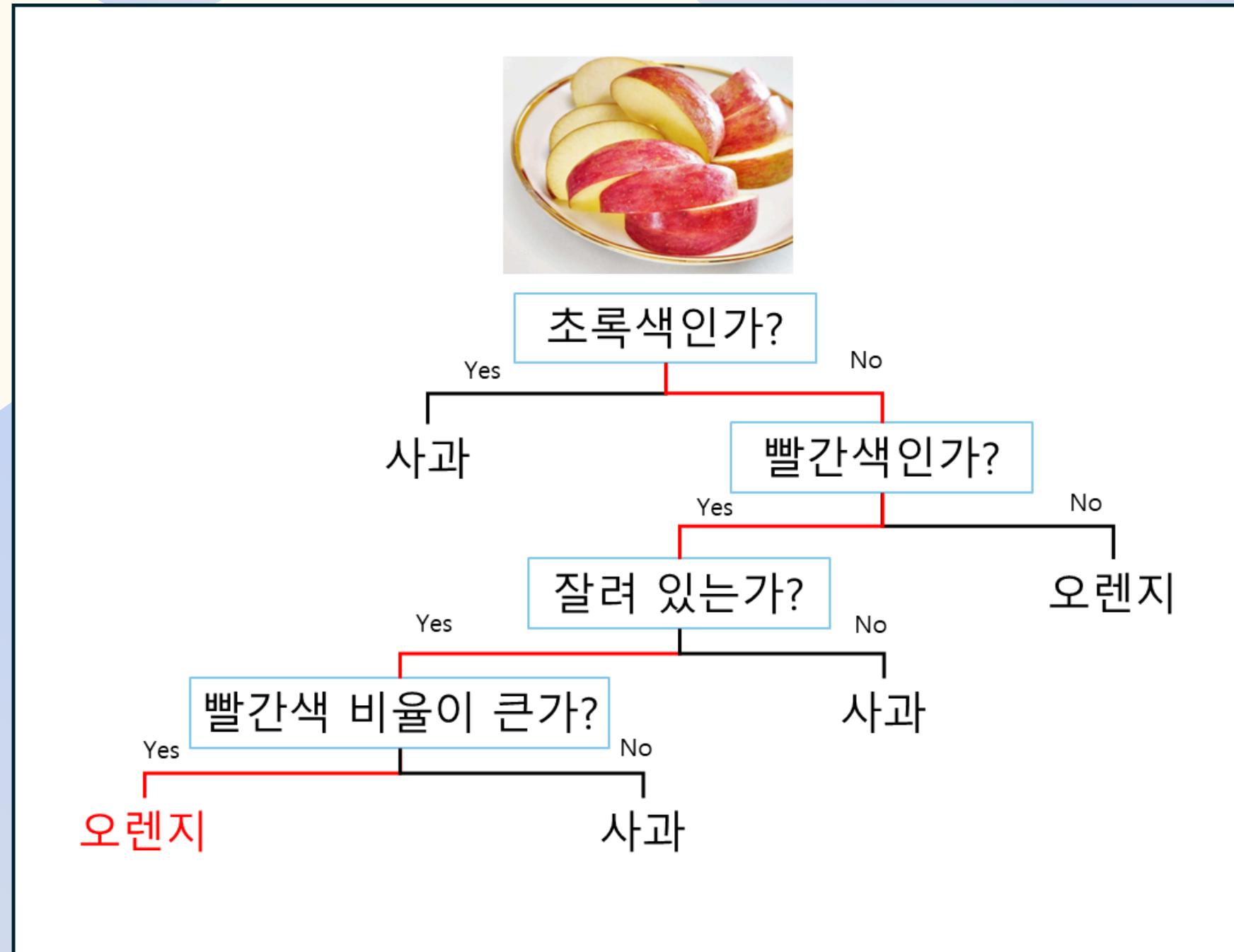


#04 머신러닝 모델 돌리기

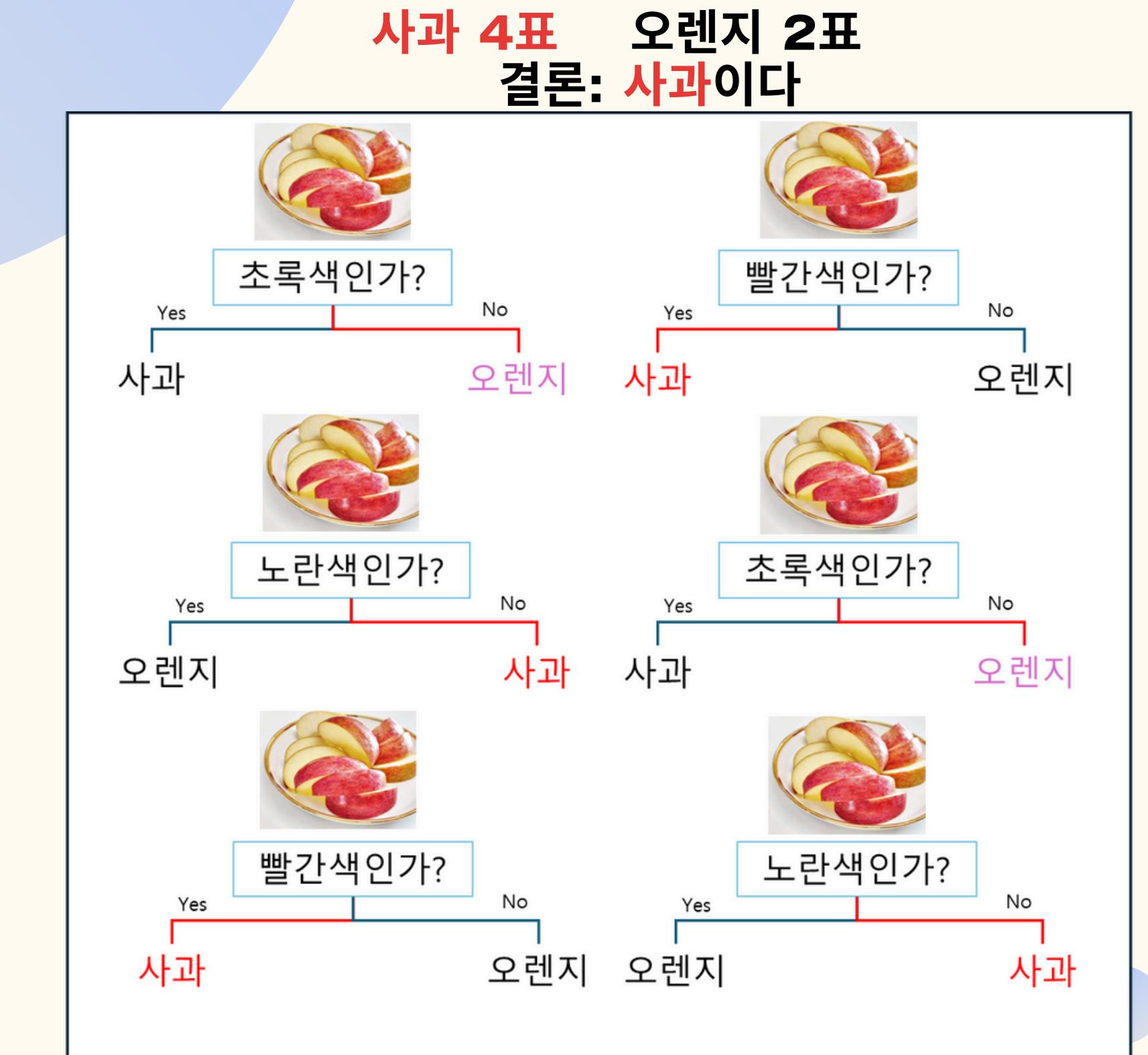


다음 사과 사진을 학습 시킨 트리를 통해 분류해 보자

#04 머신러닝 모델 돌리기



Decision Tree

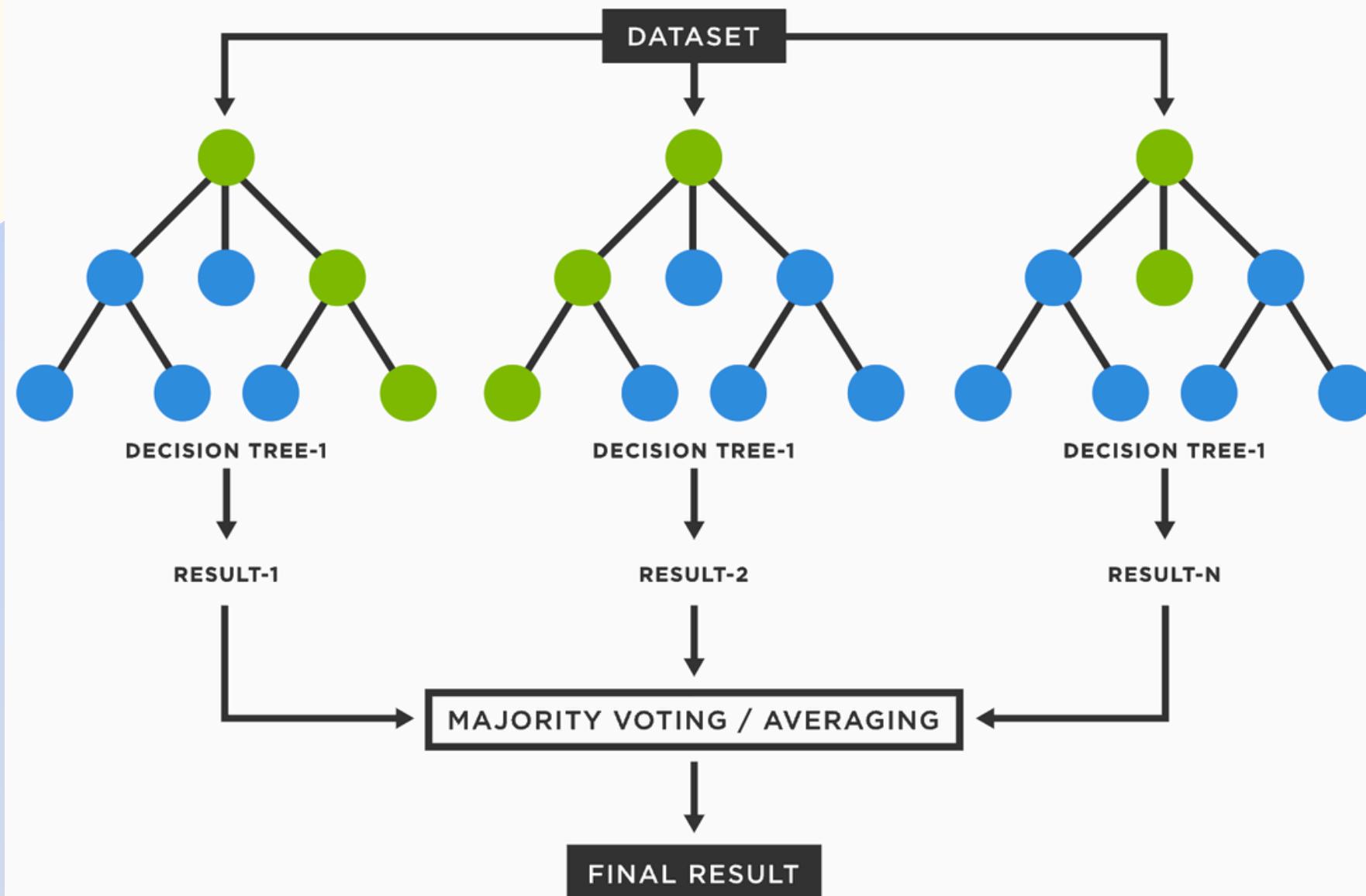


Random Forest

사과 4표 오렌지 2표
결론: 사과이다

#04 머신러닝 모델 돌리기

Random Forest



랜덤 포레스트(RandomForest)는 의사결정트리(Decision Tree)를 배깅(Vagging) 방식으로 만든 알고리즘이다.

Vagging:
부트스트랩(Bootstrap)을 통해 다양한 서브 데이터셋을 생성하고 여러 개의 의사결정나무가 각각의 데이터셋을 학습하고 평균/투표하여 최종 값을 결정한다.

단일 의사결정나무가 가질 수 있는 과적합 문제를 해결 할 수 있는 알고리즘이다.

RandomForest 주요 파라미터

n_estimators

생성하는 결정트리의 개수

지나치게 작은 값: 모델이 과소적합* 될 가능성이 크다.

지나치게 큰 값: 모델의 성능이 향상될 가능성이 크다.

하지만 모델을 실행하는 데에 드는 시간과 자원이 지나치게 커진다.

max_depth

생성하는 결정트리의 최대 깊이

지나치게 작은 값: 모델이 과소적합 될 가능성이 크다.

지나치게 큰 값: 모델이 과적합*될 가능성이 크다.

*과소적합 = 데이터의 복잡한 특징을 충분히 학습하지 못함

*과대적합(과적합) = 트리가 데이터의 잡음까지 학습하게 되어 일반화 성능이 떨어질 수 있다.

Random Forest 실습

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
rf = RandomForestClassifier()

# 최적 파라미터 찾기
grid = {'n_estimators': [10, 20, 30, 40, 50], 'max_depth': [2, 5, 10]}

clf_rf = GridSearchCV(rf, grid, cv=5)
clf_rf.fit(X_train, y_train)

pred_rf = clf_rf.predict(X_test)
# 정확도 계산
acc_rf = accuracy_score(pred_rf, y_test)
print(acc_rf)
```

모델의 정확도 = 1.0?

```
print(len(y_test))
```

1976

```
y_test.info()
```

```
<class 'pandas.core.series.Series'  
Index: 1976 entries, 4818 to 5118  
Series name: blueWins
```

```
Non-Null Count Dtype
```

```
-----  
1976 non-null    int64
```

여태까지 보지 못한 데이터에 대해 성능은?
= 일반화

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
rf = RandomForestClassifier()

# 최적 파라미터 찾기
grid = {'n_estimators': [10, 20, 30, 40, 50], 'max_depth': [2, 5, 10]}

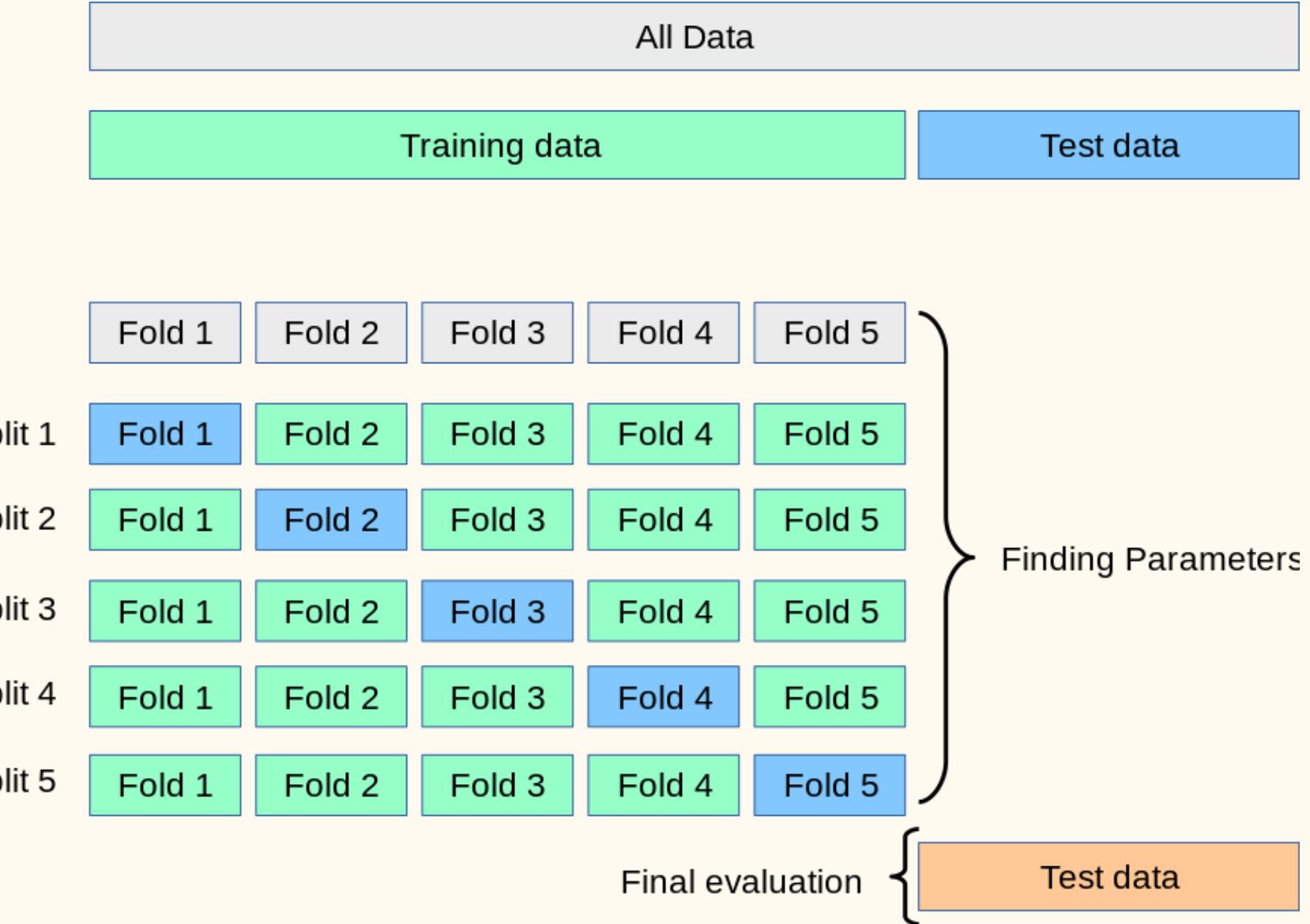
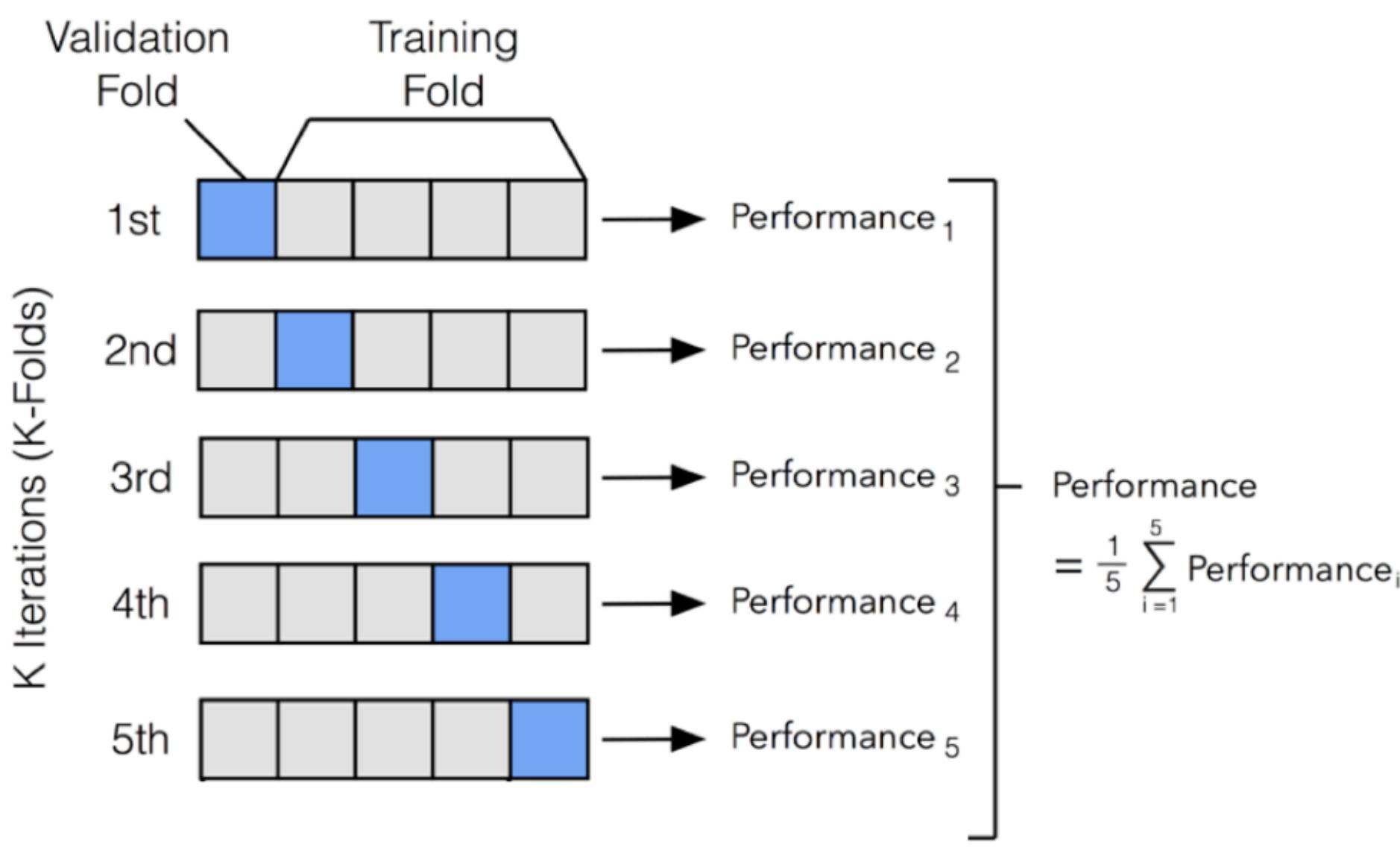
clf_rf = GridSearchCV(rf, grid, cv=5)
clf_rf.fit(X_train, y_train)

pred_rf = clf_rf.predict(X_test)
# 정확도 계산
acc_rf = accuracy_score(pred_rf, y_test)
print(acc_rf)
```

어제 Random Forest 실습 코드

Cross - Validation (교차 검증)

Test_data에 대한 과적합 해결



K-fold 교차검증

Time series data (시계열 데이터)

BTC/USDT Weekly Close Price



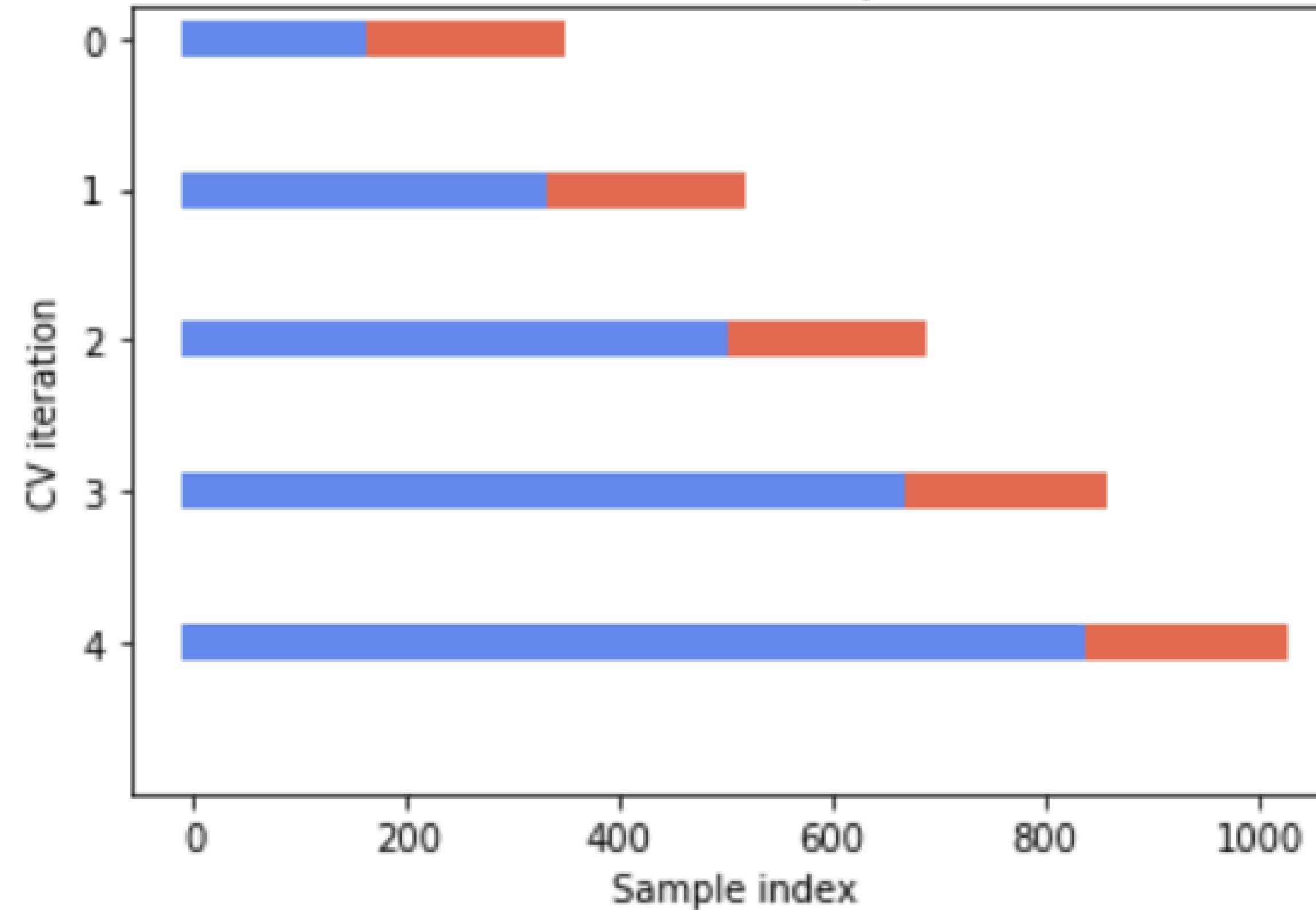
Time series data (시계열 데이터)

BTC/USDT Weekly Close Price

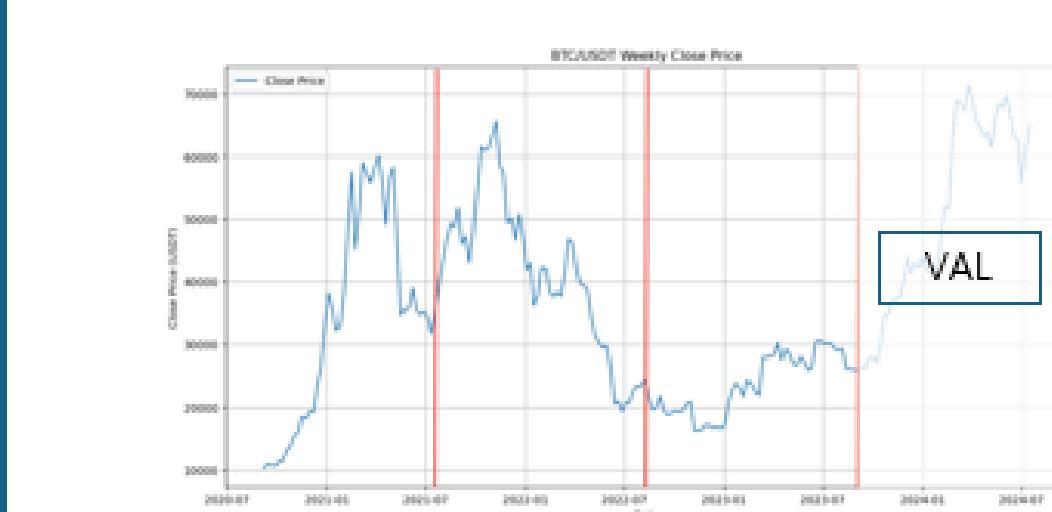
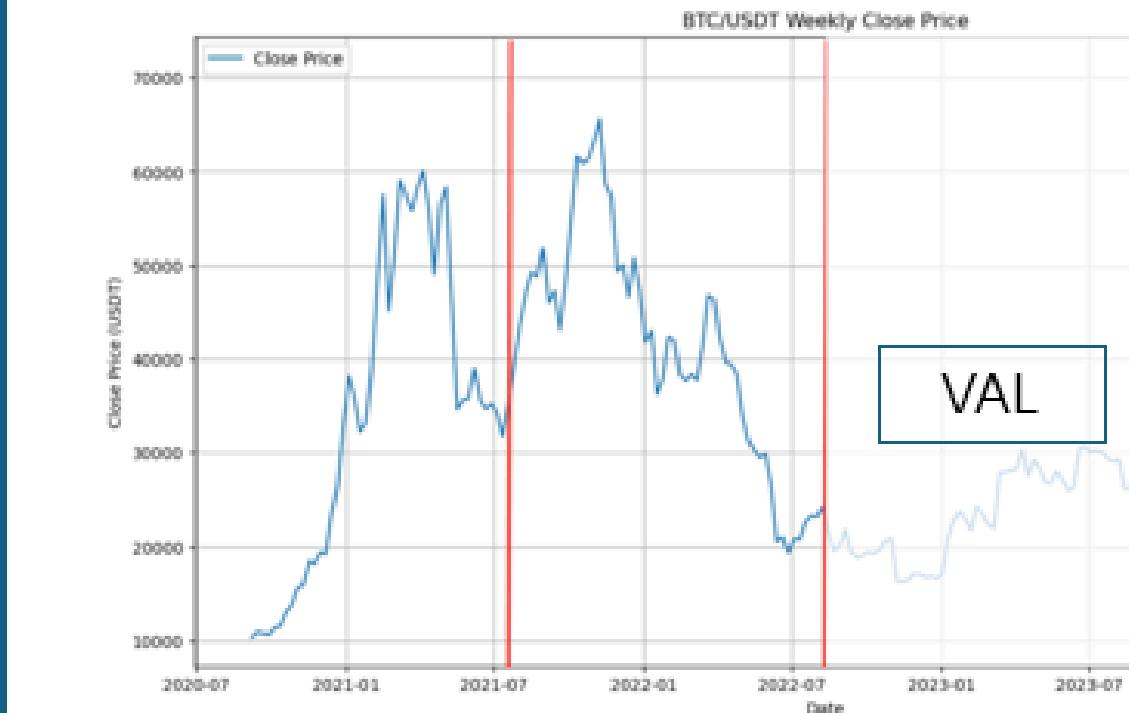
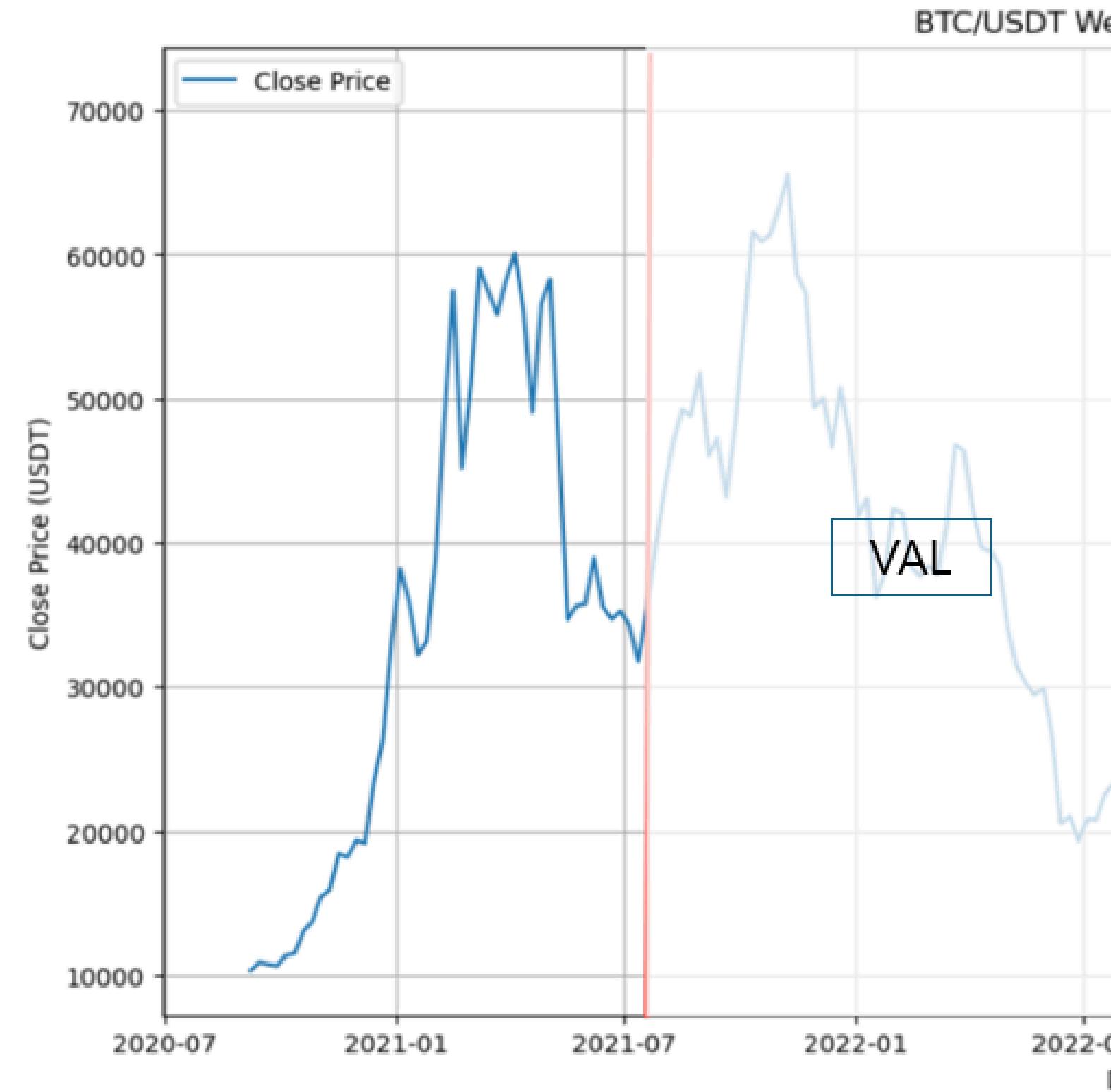


Time series data Cross - Validation (시계열 데이터 교차 검증)

TimeSeriesSplit



Time series data Cross - Validation (시계열 데이터 교차 검증)

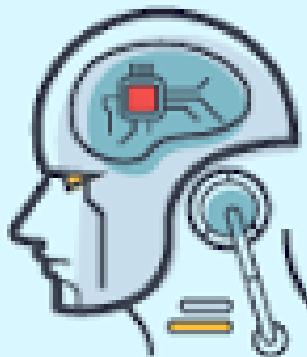


#04 머신러닝 모델 돌리기

Artificial Intelligence

인공지능

사고나 학습등 인간이 가진
지적 능력을 컴퓨터를 통해
구현하는 기술



Machine Learning

머신러닝

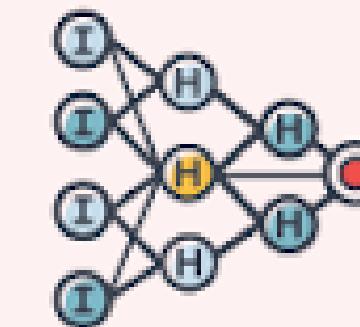
컴퓨터가 스스로 학습하여
인공지능의 성능을
향상 시키는 기술 방법



Deep Learning

딥러닝

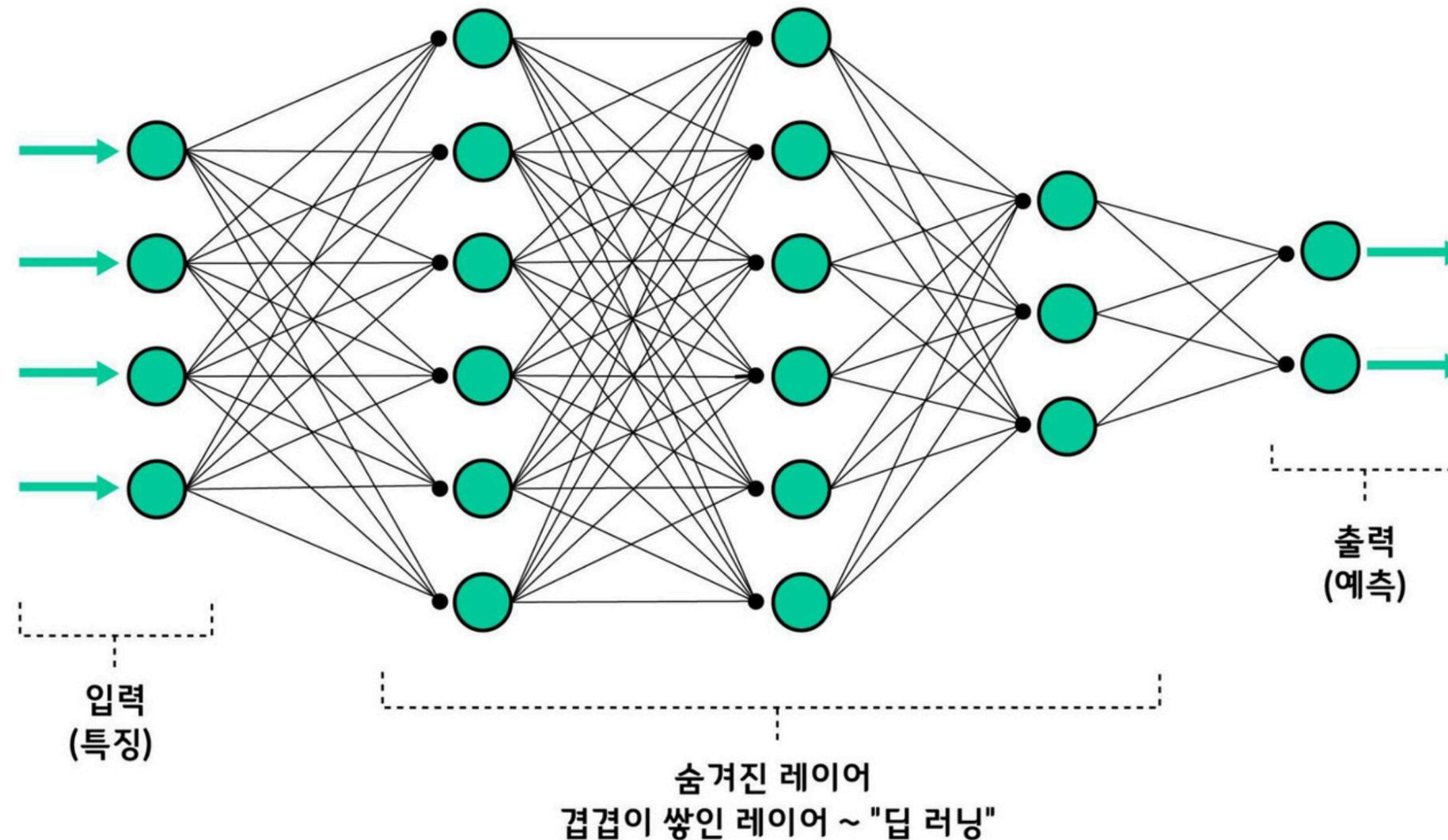
인간의 뉴런과 비슷한
인공신경망 방식으로
정보를 처리



머신러닝? 딥러닝?

#04 머신러닝 모델 돌리기

딥 러닝 (Deep Learning) - 인공 신경망의 구조



#04 머신러닝 모델 돌리기



#04 머신러닝 모델 돌리기

- 변수 중요도

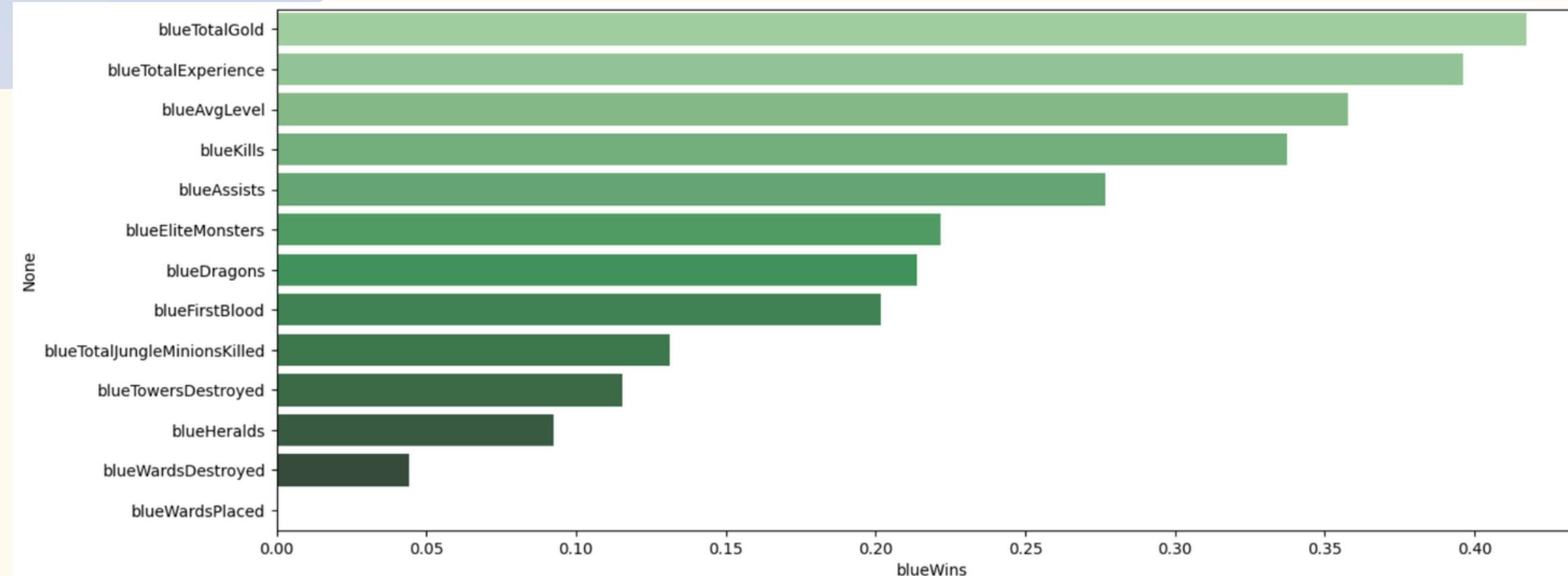
각 컬럼들과 목표 변수인 ‘blueWins’ 와의 상관관계

```
✓ 2초
▶ dfb = data_drop.loc[:,~data_drop.columns.str.startswith('red')]
plt.figure(figsize=(15,6))
dfw = dfb.corr()[['blueWins']].drop(['blueWins', 'blueDeaths'])
dfw = dfw.sort_values(ascending=False)

pal = sns.color_palette("Greens_d", len(dfw))
rank = dfw.argsort().argsort()

sns.barplot(y=dfw.index, x=dfw, palette=np.array(pal[::-1])[rank])

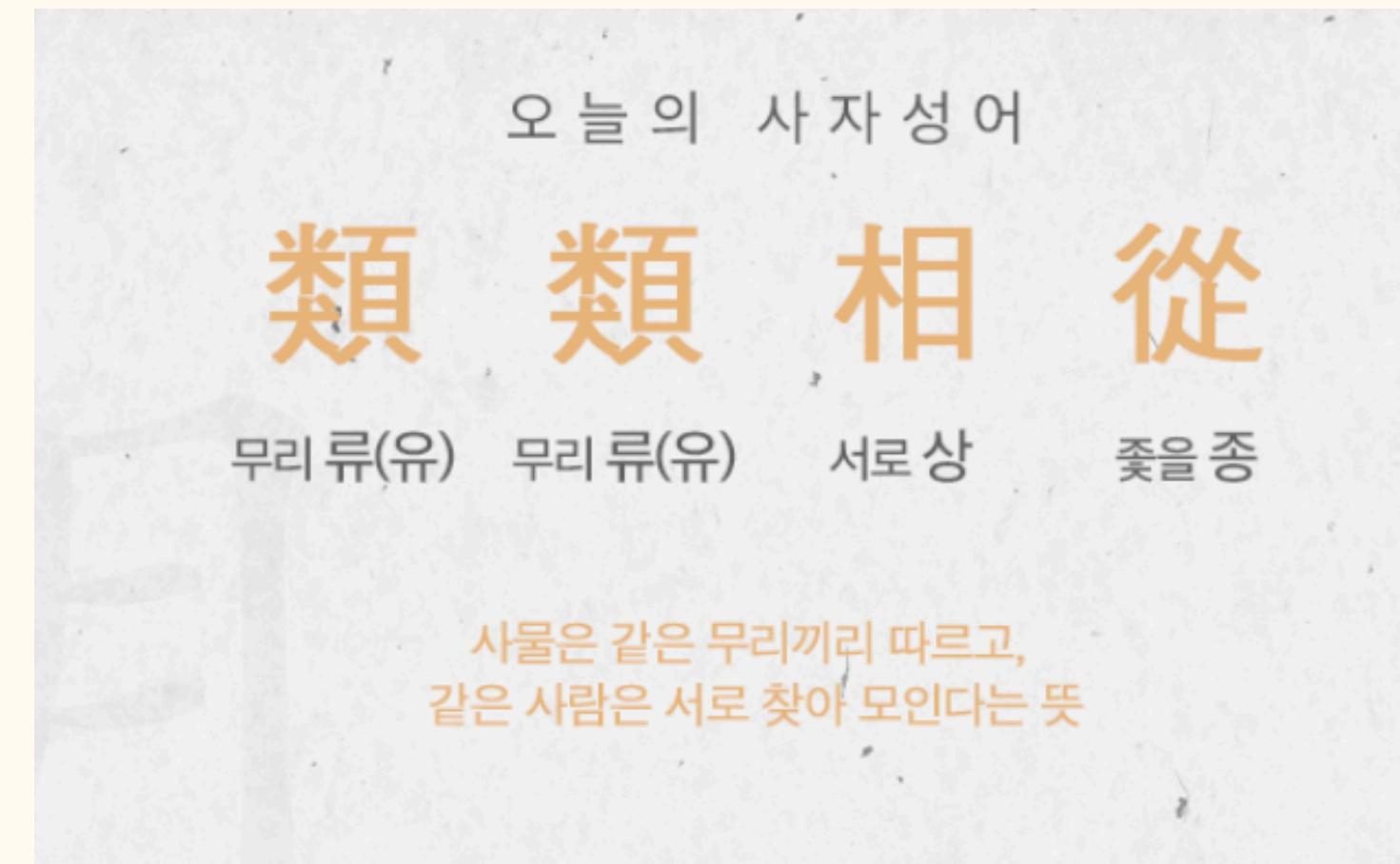
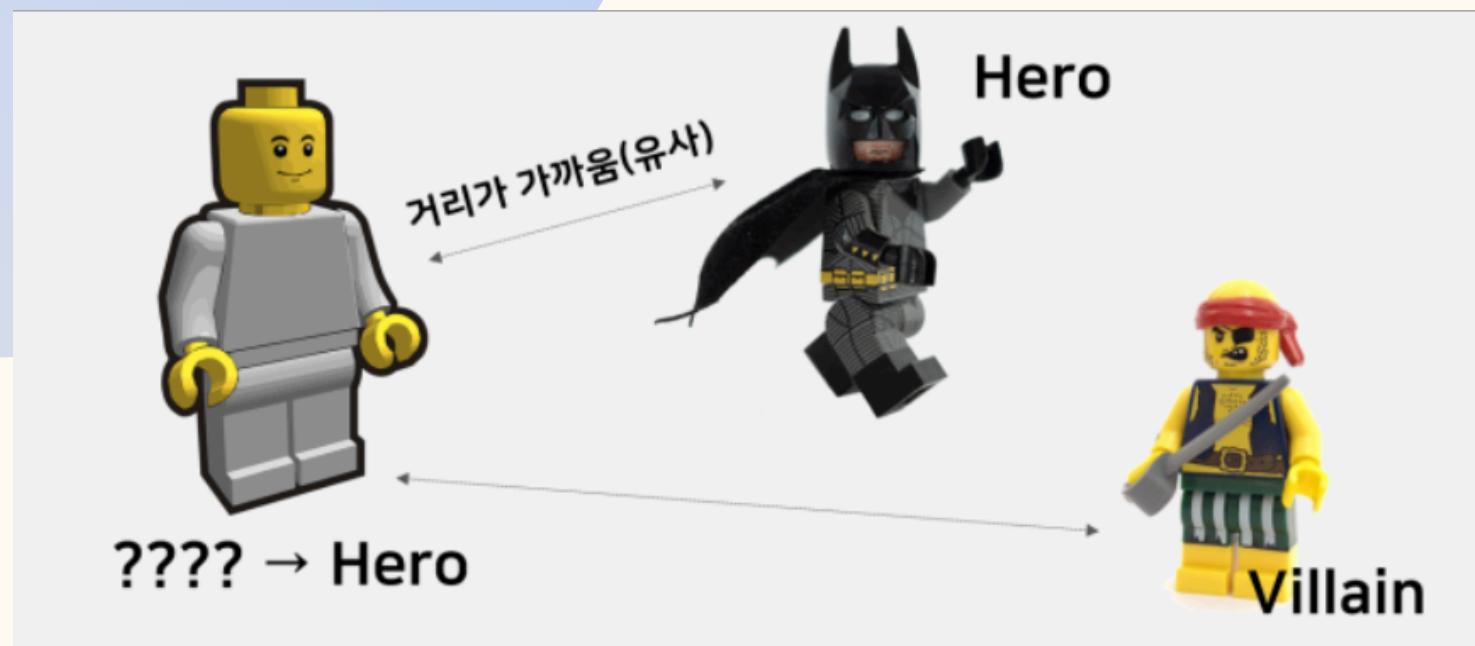
plt.show()
→ <ipython-input-26-b87f4a29dd4d>:9: FutureWarning:
```



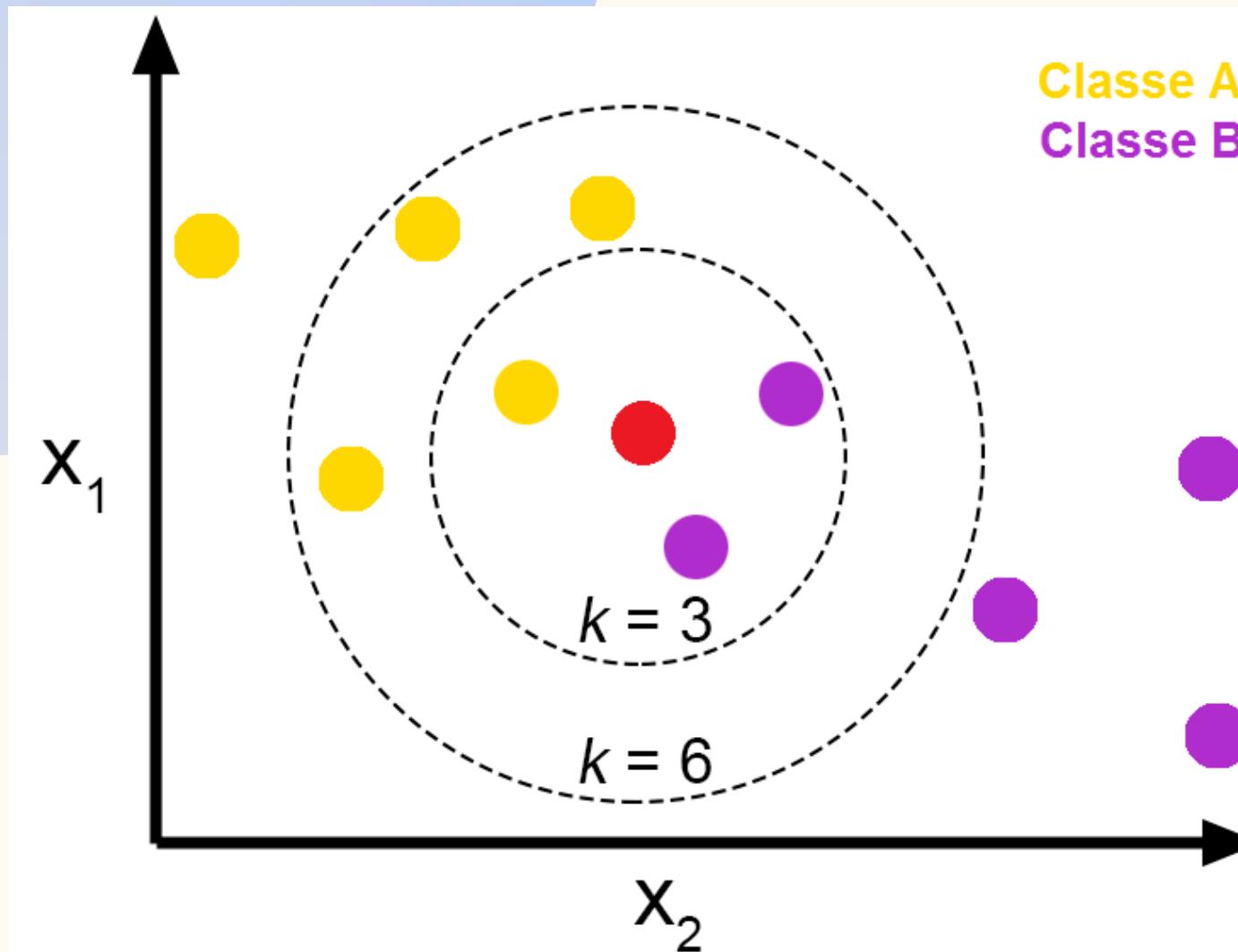
1. ‘blue TotalGold’
2. ‘blue Total Experience’
3. ‘blue AvgLevel’
4. ‘blue Kills’

#04 머신러닝 모델 돌리기

K-NN

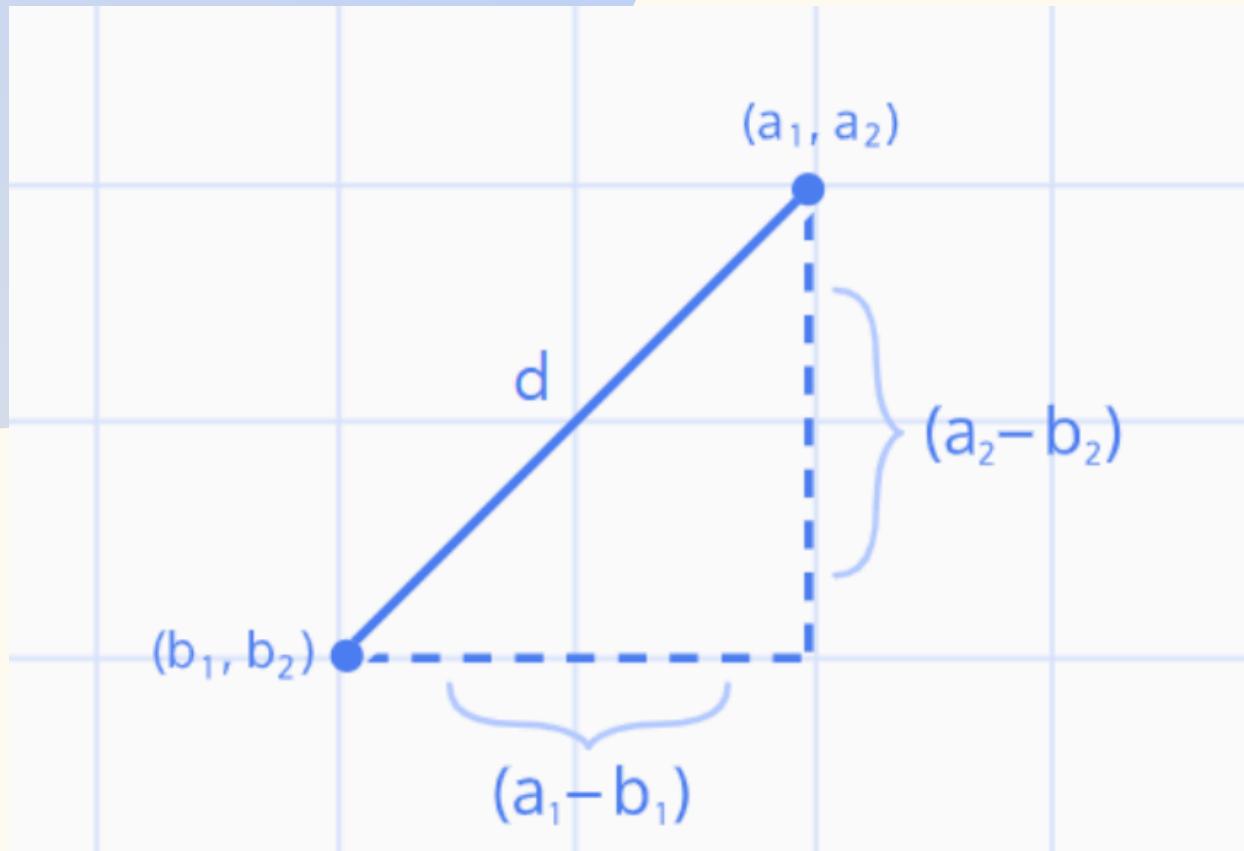


K-NN



현재 데이터와 가장 가까운 K 개의 데이터를 찾아 가장 많은 분류 값으로 현재의 데이터를 분류하는 알고리즘
= K 개의 이웃으로 나(현재 데이터)를 판단
 K 값을 정하는 것이 중요
 K 가 너무 작은 경우: 데이터의 지역적인 특징 과하게 반영
 K 가 너무 큰 경우: 모델이 과하게 정규화 됨

K-NN



유클리디안 거리(Euclidean distance)
피타고라스의 정리로 이해하기
= 두 점 사이의 거리 공식

KNN 실습

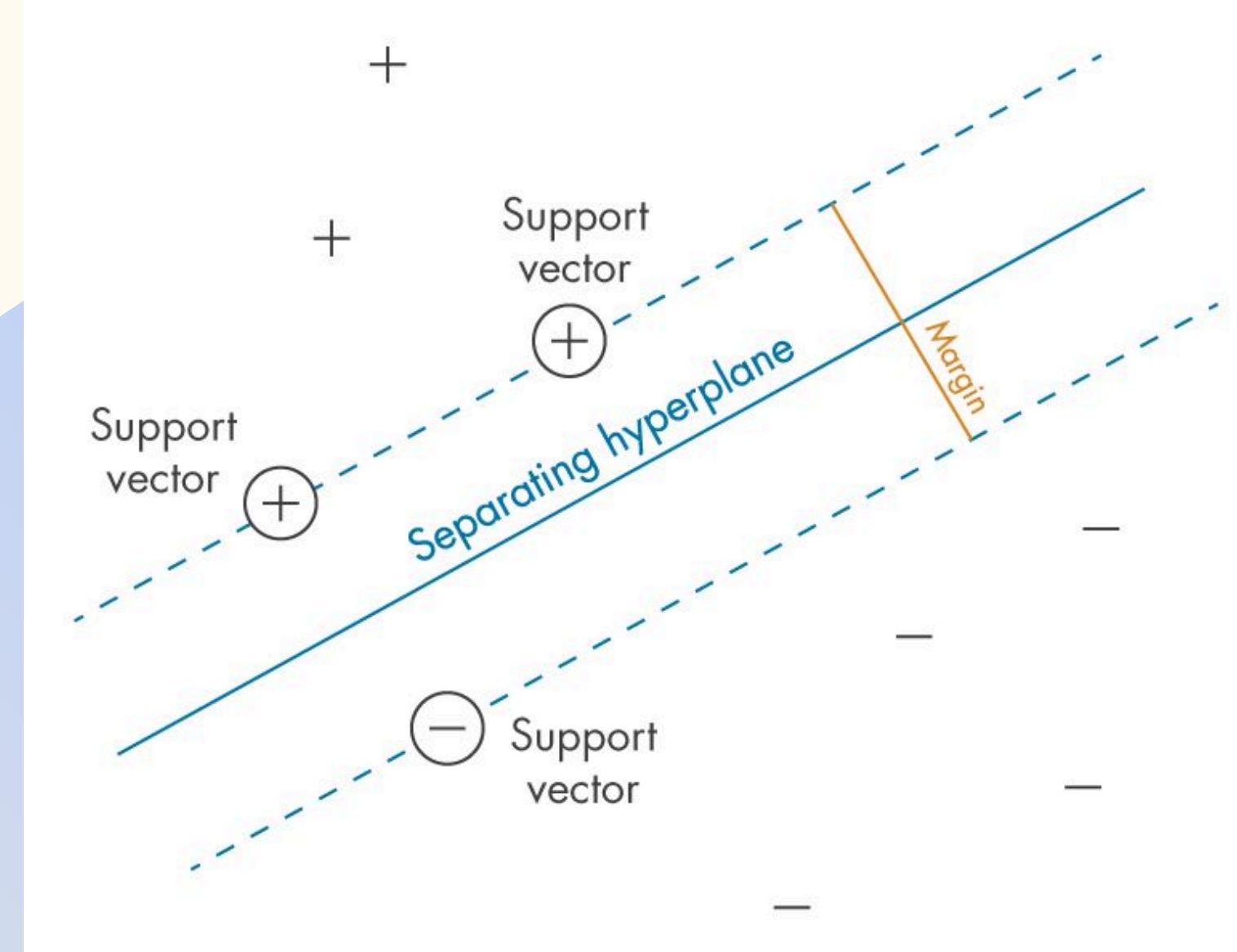
```
✓ [21] #knn
43초
from sklearn.neighbors import KNeighborsClassifier # 모듈 임포트
from sklearn.model_selection import GridSearchCV
knn = KNeighborsClassifier() # KNN 모델 불러오기

# search the best params
grid = {"n_neighbors":np.arange(1,50)}
clf_knn = GridSearchCV(knn, grid, cv=5) # 최적의 하이퍼파라미터 찾기
clf_knn.fit(X_train,y_train) # KNN 모델 학습하기

# get accuracy score
pred_knn = clf_knn.predict(X_test) # KNN 모델로 예측한 결과
acc_knn = accuracy_score(pred_knn, y_test) # 예측한 값과 실제 레이블 간의 비교를 통해 정확도 구하기
print(clf_knn.best_params_)
print(acc_knn)

→ {'n_neighbors': 48}
0.7171052631578947
```

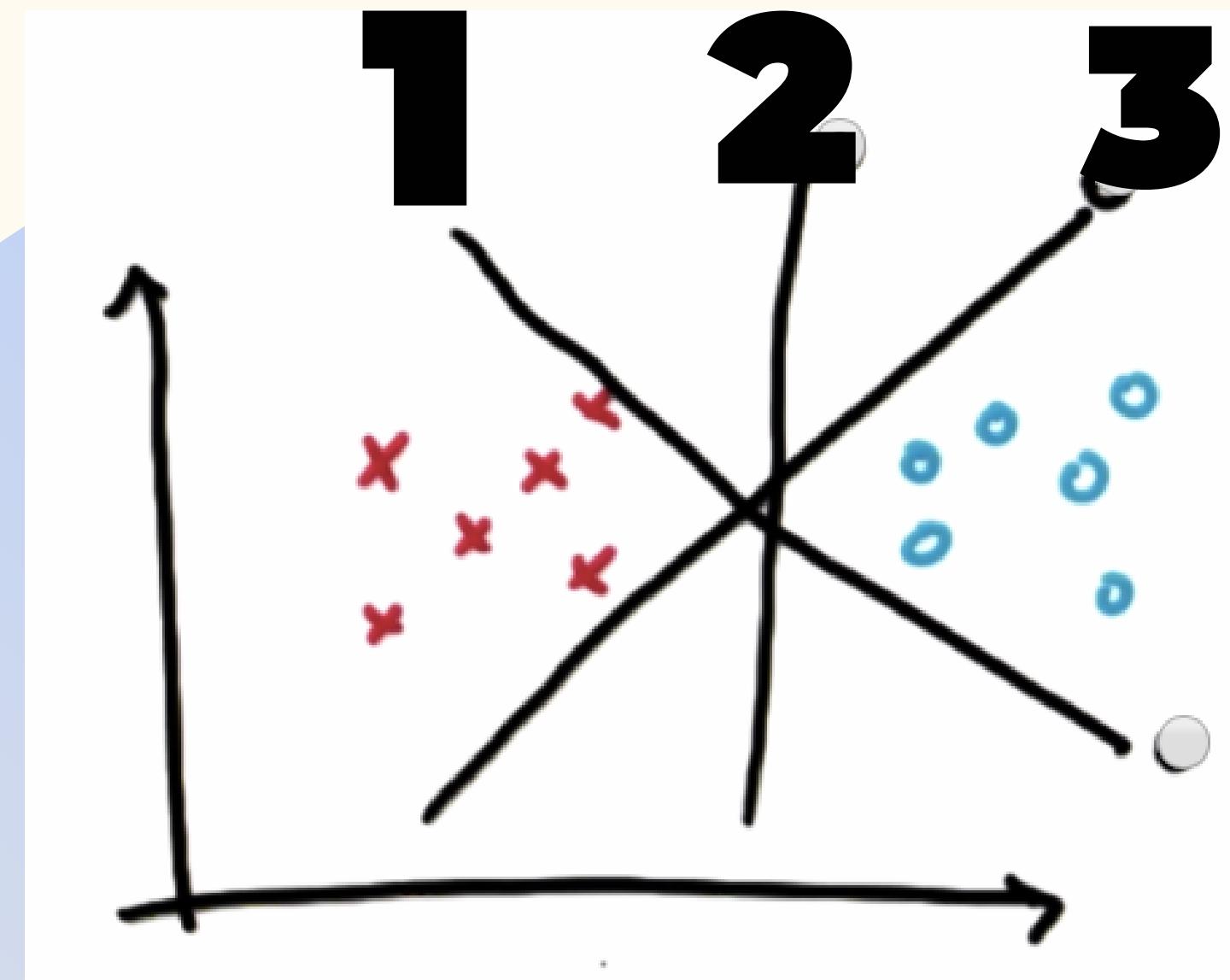
SVM



서포트 벡터 머신(Support Vector Machine 또는 SVM)은 지도 학습 알고리즘 중 하나로, 분류와 회귀 분석에 모두 사용

직선(선형모델)으로 그 둘을 구분 할 수도 있고 때에 따라서는 구불구불한 곡선으로(비선형모델)으로 둘을 구분

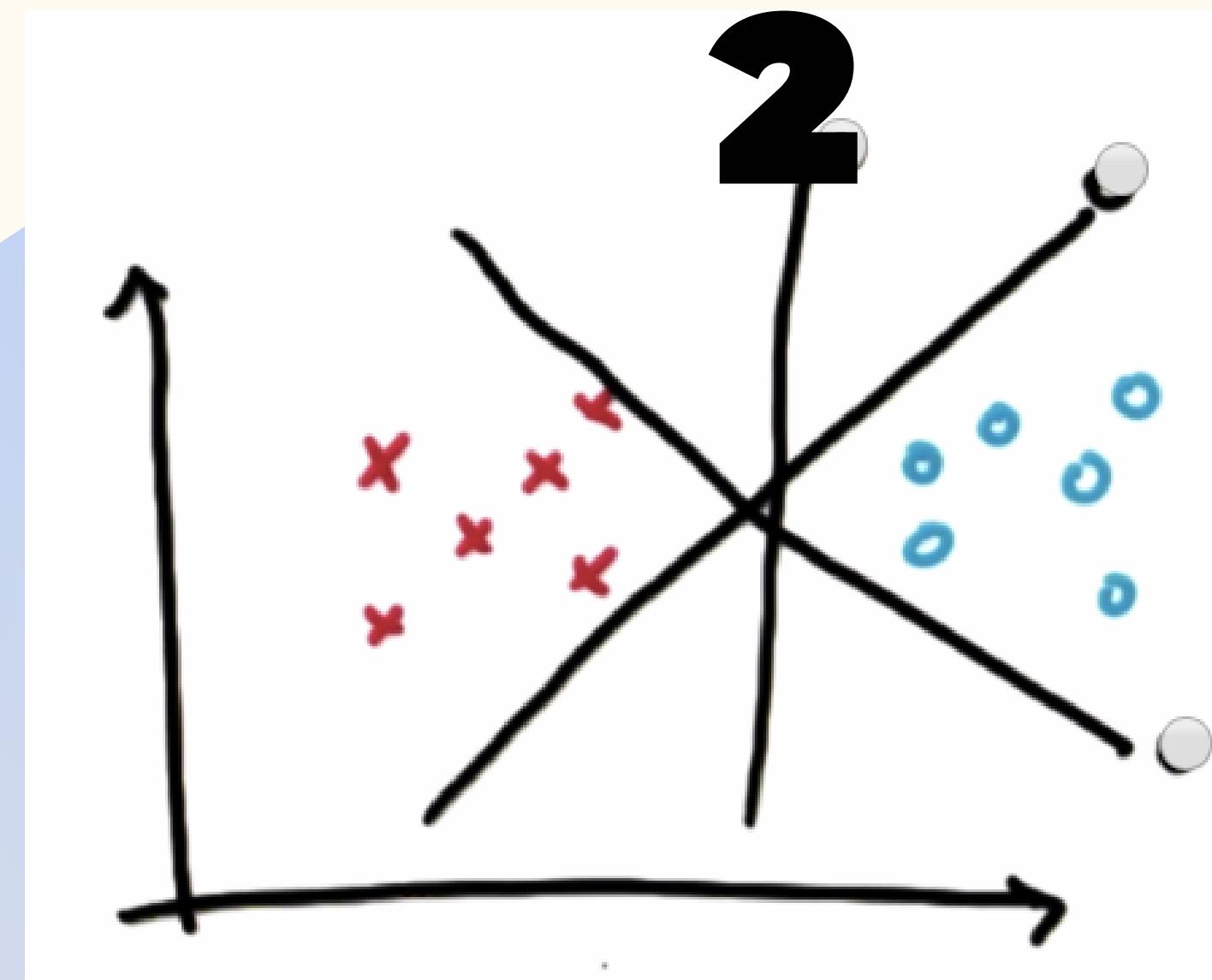
SVM



svm은 margin값이 가장 큰 Decision Boundary를 선택

무슨 선이 가장 큰 margin값을 갖고 있나요?

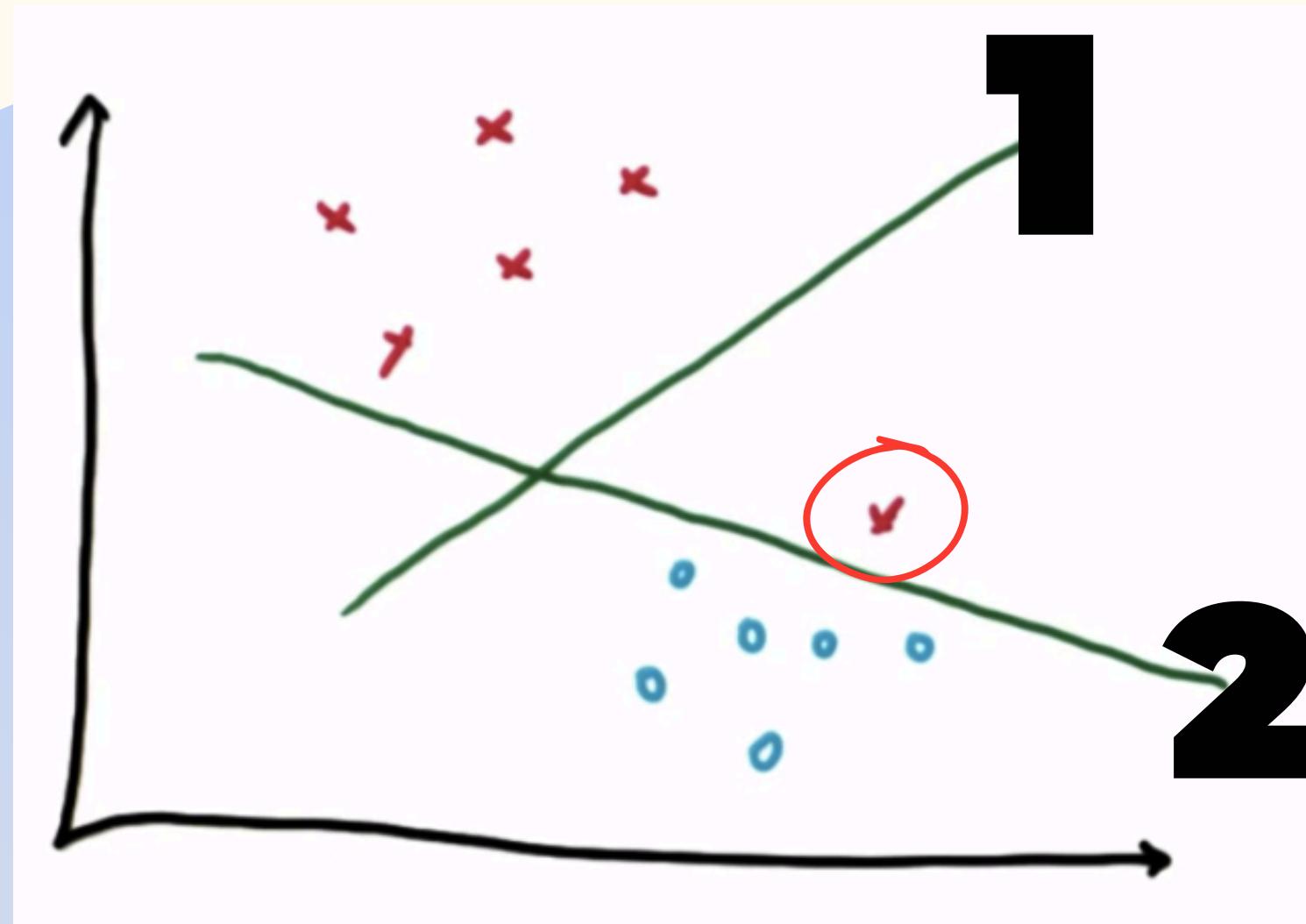
SVM



svm은 margin값이 가장 큰 Decision Boundary를 선택

따라서 2번 Decision Boundary를 선택하면
margin 값이 가장 큰 것

SVM



svm은 margin값이 가장 큰 Decision Boundary를 선택
하지만 무조건 margin값이 큰 선을 선택하면 오류 발생

1번 구분선이 margin값은 크지만
빨간 데이터 하나를 구분해내지 못함

데이터를 정확히 분류하는 범위를 먼저 찾고,
그 범위 안에서 Margin을 최대화하는 구분선을 택함

#04 머신러닝 모델 돌리기

✓ 8초



```
from sklearn.svm import SVC  
from sklearn.metrics import accuracy_score
```

SVM 모델 생성 및 학습

```
svm_model = SVC()  
svm_model.fit(X_train, y_train)
```

테스트 세트 예측

```
pred_svm = svm_model.predict(X_test)
```

모델 평가

```
acc_svm = accuracy_score(pred_svm, y_test)  
print(f'SVM Model Accuracy: {acc_svm}')
```



SVM Model Accuracy: 0.7317813765182186



SVM 주요 매개변수

C(Regularization Parameter)

오차 항의 허용 정도를 조절하는 매개변수

C값이 작으면,
결정 경계가 부드러워서 많은 오차를 허용

C값이 크면,
오차를 적게 허용하고 복잡한 결정 경계로
더 까다로움

기본값은 1.0

✓ 3초 [35] # SVM 모델 생성 및 학습
svm_model = SVC(C=0.8, kernel='rbf', gamma='scale')
svm_model.fit(X_train, y_train)

테스트 세트 예측
pred_svm = svm_model.predict(X_test)

모델 평가
acc_svm = accuracy_score(pred_svm, y_test)
print(f'SVM Model Accuracy: {acc_svm}')

→ SVM Model Accuracy: 0.7322874493927125

✓ 7초 [36] # SVM 모델 생성 및 학습
svm_model = SVC(C=2.0, kernel='rbf', gamma='scale')
svm_model.fit(X_train, y_train)

테스트 세트 예측
pred_svm = svm_model.predict(X_test)

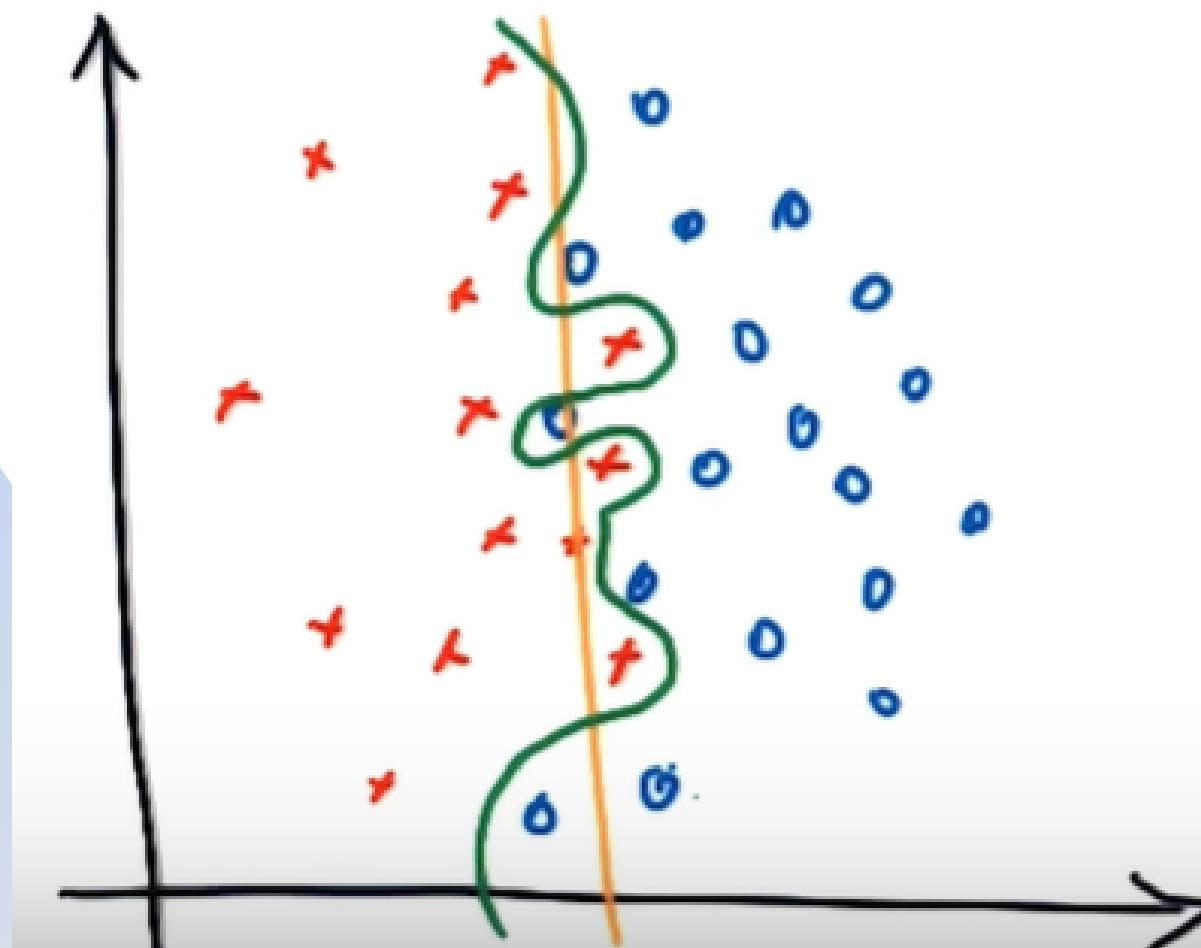
모델 평가
acc_svm = accuracy_score(pred_svm, y_test)
print(f'SVM Model Accuracy: {acc_svm}')

→ SVM Model Accuracy: 0.7302631578947368

SVM 주요 매개변수

C(Regularization Parameter)

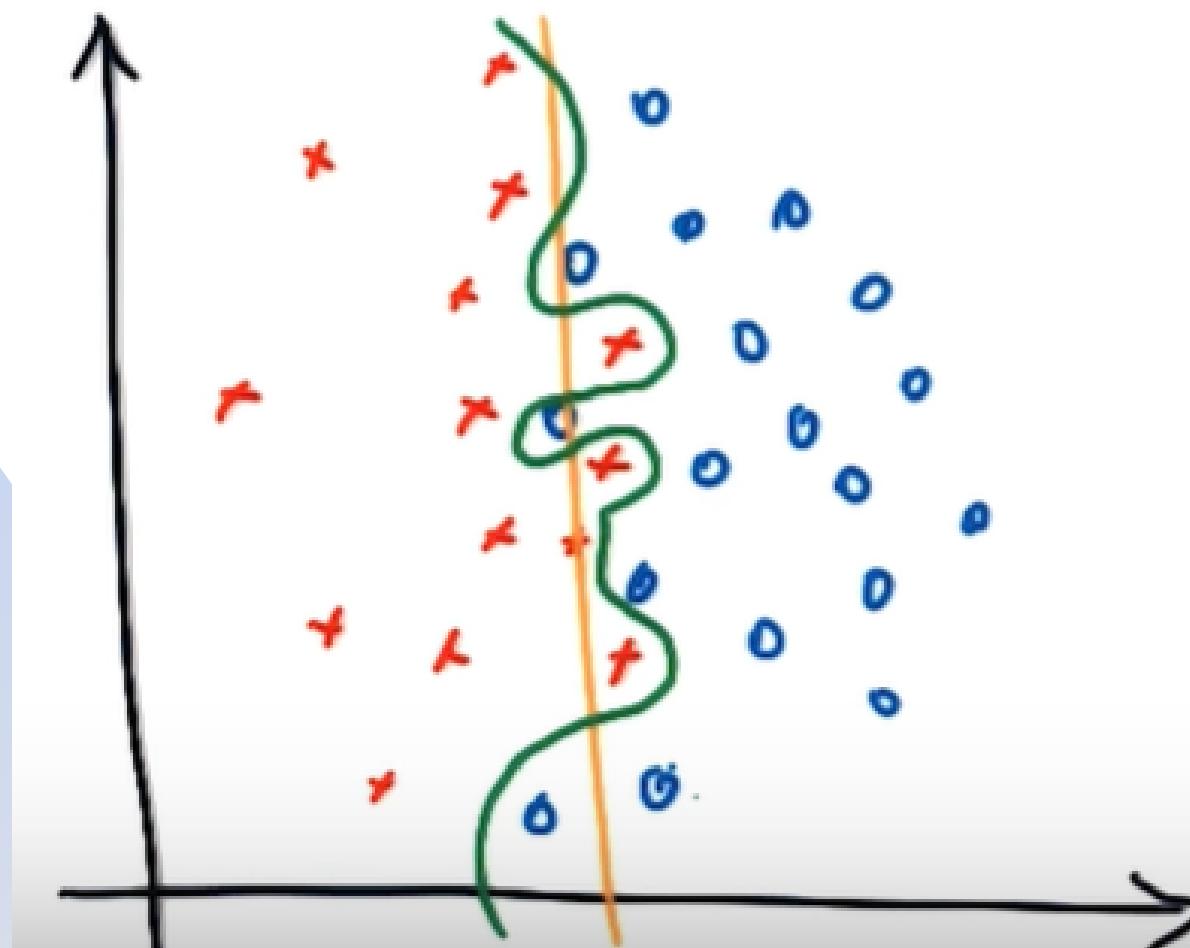
오차 항의 허용 정도를 조절하는 매개변수



초록색 구분선이 C값과
주황색 구분선이 C값 중
무슨 색의 구분선의 c 값이 큼까요??

SVM 주요 매개변수

C(Regularization Parameter)
오차 항의 허용 정도를 조절하는 매개변수



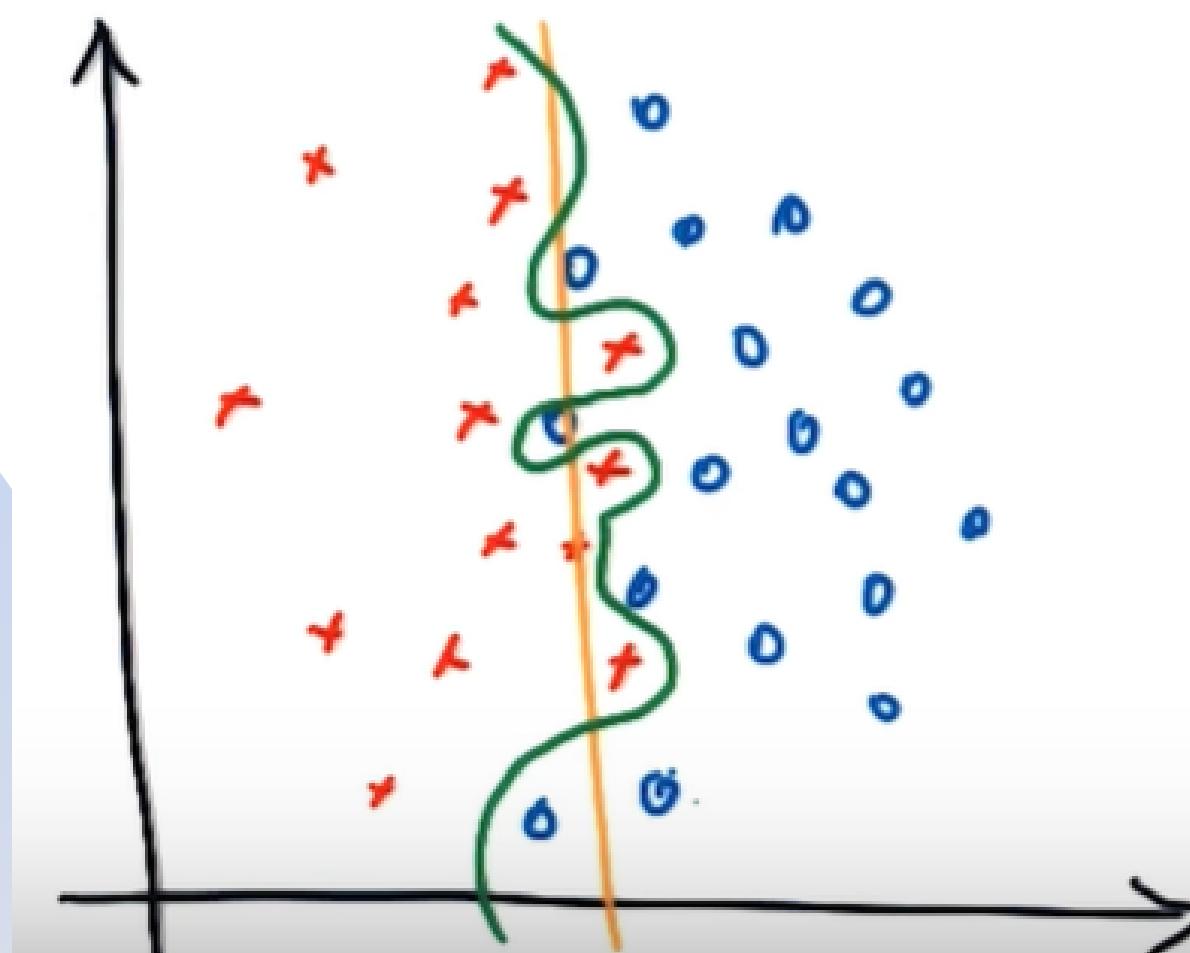
초록색 구분선이 c값과
주황색 구분선이 c값 중
무슨 색의 구분선의 c 값이 클까요??

hint!
작을수록 decision boundary은 직선에 가까워진다

SVM 주요 매개변수

C(Regularization Parameter)

오차 항의 허용 정도를 조절하는 매개변수



초록색 구분선이 c 값이 큰 것
주황색 구분선이 c 값이 작은 것
작을수록 decision boundary은 직선에 가까워진다

SVM 주요 매개변수

Gamma

데이터 포인트의 영향력을 조절

scale은 데이터의 특성 수와 데이터의 분산에 기반

auto는 데이터의 특성 수에 기반

특정 실수 값을 설정할 수도 있음

기본값은 scale

13초

```

from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# 데이터 준비: 이미 스케일링되고 분할된 데이터 사용
# X_train, X_test, y_train, y_test

# 'scale' gamma를 사용하는 SVM 모델 생성 및 학습
svm_model_scale = SVC(C=1.0, kernel='rbf', gamma='scale')
svm_model_scale.fit(X_train, y_train)
pred_svm_scale = svm_model_scale.predict(X_test)
acc_svm_scale = accuracy_score(pred_svm_scale, y_test)
print(f'SVM Model Accuracy with gamma="scale": {acc_svm_scale}')

# 'auto' gamma를 사용하는 SVM 모델 생성 및 학습
svm_model_auto = SVC(C=1.0, kernel='rbf', gamma='auto')
svm_model_auto.fit(X_train, y_train)
pred_svm_auto = svm_model_auto.predict(X_test)
acc_svm_auto = accuracy_score(pred_svm_auto, y_test)
print(f'SVM Model Accuracy with gamma="auto": {acc_svm_auto}')

# 사용자 정의 gamma 값을 사용하는 SVM 모델 생성 및 학습
svm_model_custom = SVC(C=1.0, kernel='rbf', gamma=0.1)
svm_model_custom.fit(X_train, y_train)
pred_svm_custom = svm_model_custom.predict(X_test)
acc_svm_custom = accuracy_score(pred_svm_custom, y_test)
print(f'SVM Model Accuracy with gamma=0.1: {acc_svm_custom}')

```

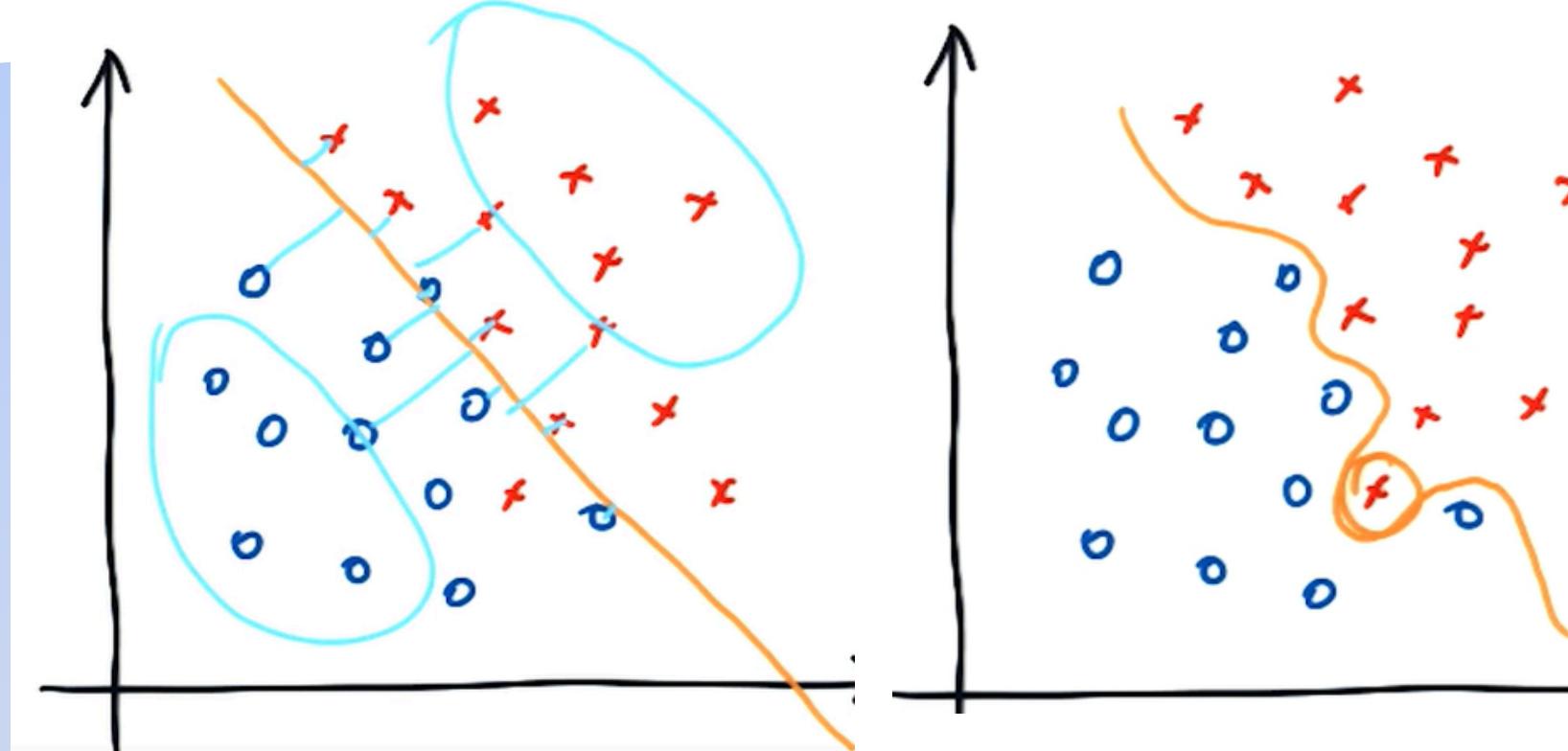
SVM Model Accuracy with gamma="scale": 0.7317813765182186
SVM Model Accuracy with gamma="auto": 0.7302631578947368
SVM Model Accuracy with gamma=0.1: 0.7302631578947368

SVM 주요 매개변수

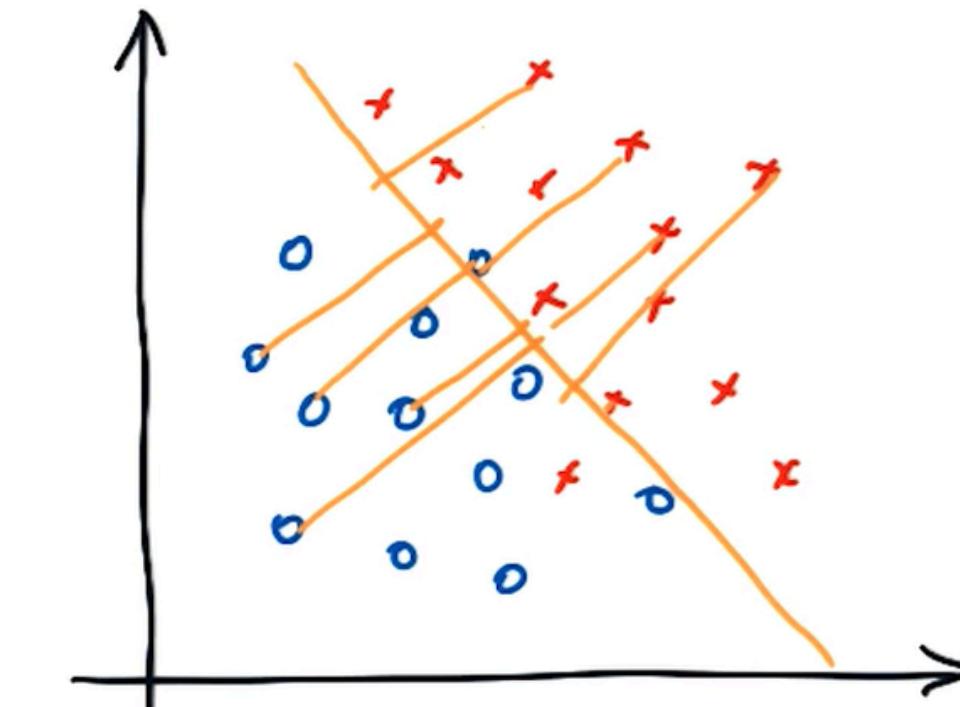
Gamma

decision boundary의 굴곡진 정도와 직선인 정도를 결정해줌

Gamma값이 크면 reach가 좁음
가까이 있는 포인트들만이 선의 굴곡에 영향
멀리 있는 포인트는 영향이 없으므로 선과 가까이 있는 포인트 하나하나의 영향이 상대적으로 크기 때문



Gamma값이 작으면 reach가 멀어짐
각 데이터 값들이 구분선에 영향을 주는 정도가 작아짐
따라서 거의 직선



SVM 과적합

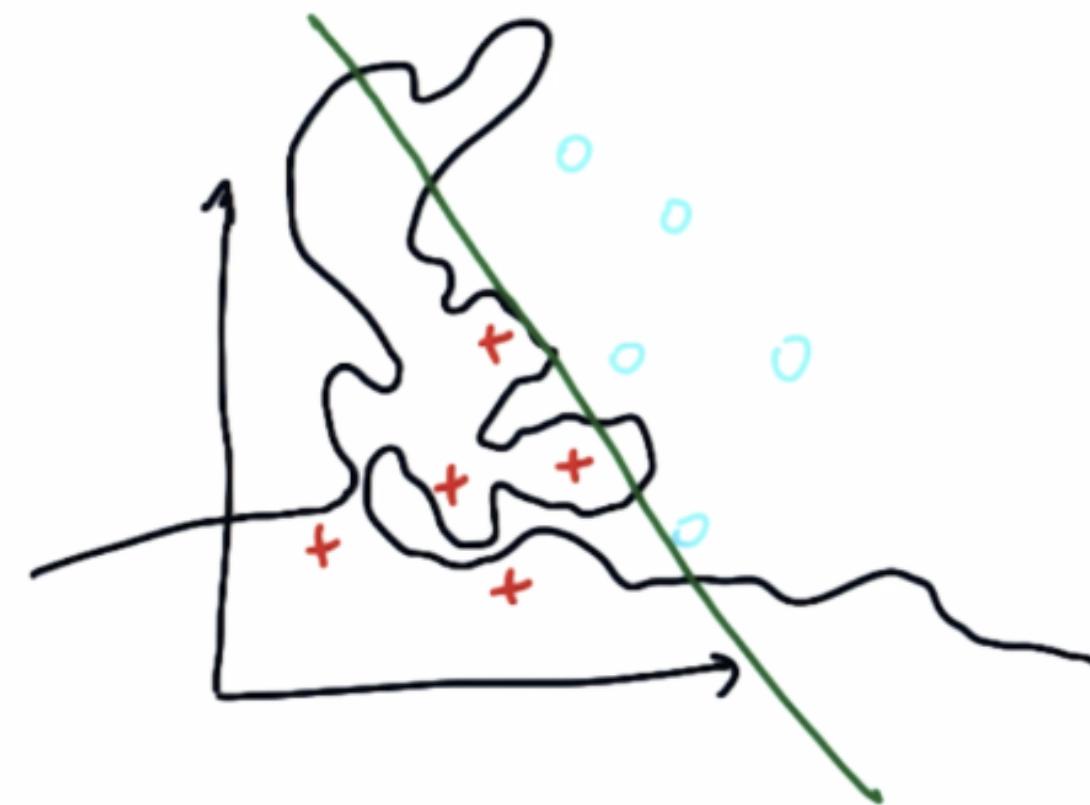
과적합 (Overfitting)

학습을 너무 과하게 해서 많이 구불구불한 구분선

C와 Gamma를 지나치게 높게 잡는다면 오버피팅

SVM은 training time이 길기 때문에 사이즈가 큰 데이터 셋에는 적합하지 않음

따라서 성능도 높이면서 과적합이 일어나지 않는 적당한 변수 값을 찾는 것이 중요



평가지표

1. 정확도(accuracy) : 전체 데이터에서 정답을 맞춘 비율(정답율)

=> “전체적으로 얼마나 맞췄는가”

2. 정밀도(precision) : 모델이 양성으로 예측한 것 중 실제 정답이 양성인 비율(양성 예측 비율)

=> “양성을 예측한 것이 얼마나 정확한가”

3. 재현율(recall) : 실제 정답이 양성인 것 중 모델이 양성이라고 예측한 비율(실제 양성 비율)

=> “실제 양성을 얼마나 놓치지 않았는가”

4. f1-score : 재현율과 정밀도의 조화평균(평균적인 변화율)

* f1 -score이 높을수록 모델 성능이 좋다고 할 수 있음

오분류표 (confusion matrix)

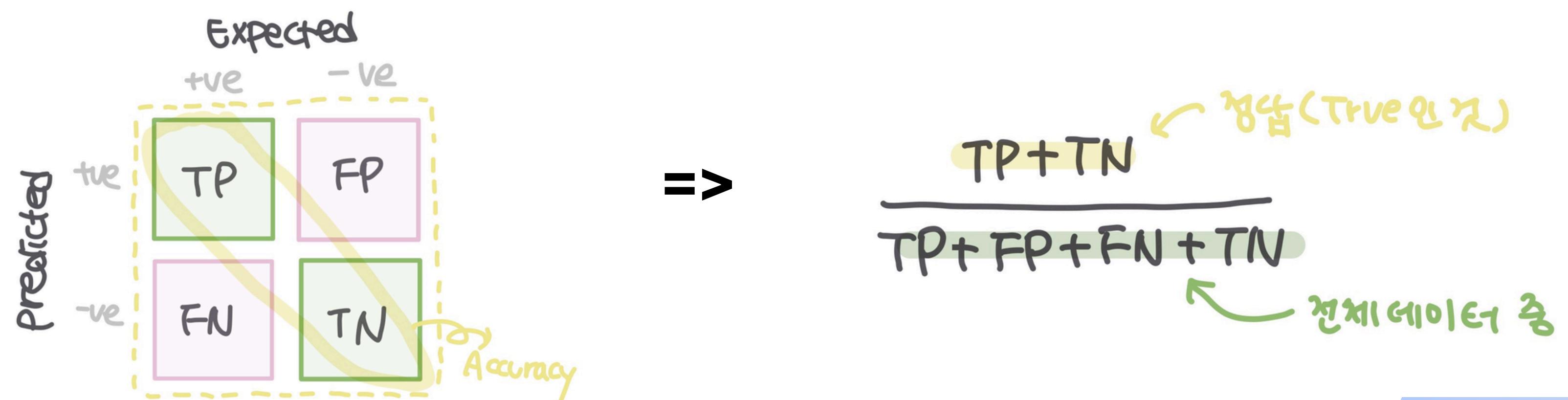
| 예측 | 실제(정답) | positive (실제 1) | negative (실제 0) |
|--------------------|--|--|--------------------|
| positive (예측 1) | TP (True Positive) 잘 예측한 경우 ($1 = 1$) | FP (False Positive) 잘못 예측한 경우 ($1 \neq 0$) | |
| negative (예측 0) | FN (False Negative) 잘못 예측한 경우 ($0 \neq 1$) | TN (True Negative) 잘 예측한 경우 ($0 = 0$) | |

*True : 정답, False : 오답 / Negative(음성), Positive(양성)

1. 정확도 (Accuracy)

: 전체 데이터에서 정답을 맞춘 비율

- TN (True Negative) : 실제가 음성인데 음성으로 **잘 예측**한 것
- FP (False Positive) : 실제가 음성인데 양성으로 **잘못 예측**한 것
- FN (False Negative) : 실제가 양성인데 음성으로 **잘못 예측**한 것
- TP (True Positive) : 실제가 양성인데 양성으로 **잘 예측한** 것



2. 정밀도(Precision)

: 모델이 양성으로 예측한 것 중
실제 정답이 양성인 비율

TN (True Negative) : 실제가 음성인데 음성으로 **잘 예측**한 것
 FP (False Positive) : 실제가 음성인데 양성으로 **잘못 예측**한 것
 FN (False Negative) : 실제가 양성인데 음성으로 **잘못 예측**한 것
 TP (True Positive) : 실제가 양성인데 양성으로 **잘 예측**한 것



$$\Rightarrow \frac{\text{TP}}{\text{TP} + \text{FP}}$$

↑ 정답
↑ 양성으로 예측한 것 중(_P)

3. 재현율(Recall)

: 실제 정답이 양성인 것 중
모델이 양성이라고 예측한 비율



- TN (True Negative) : 실제가 음성인데 음성으로 **잘 예측**한 것
- FP (False Positive) : 실제가 음성인데 양성으로 **잘못 예측**한 것
- FN (False Negative) : 실제가 양성인데 음성으로 **잘못 예측**한 것
- TP (True Positive) : 실제가 양성인데 양성으로 **잘 예측한** 것

$$\frac{TP}{TP+FN}$$

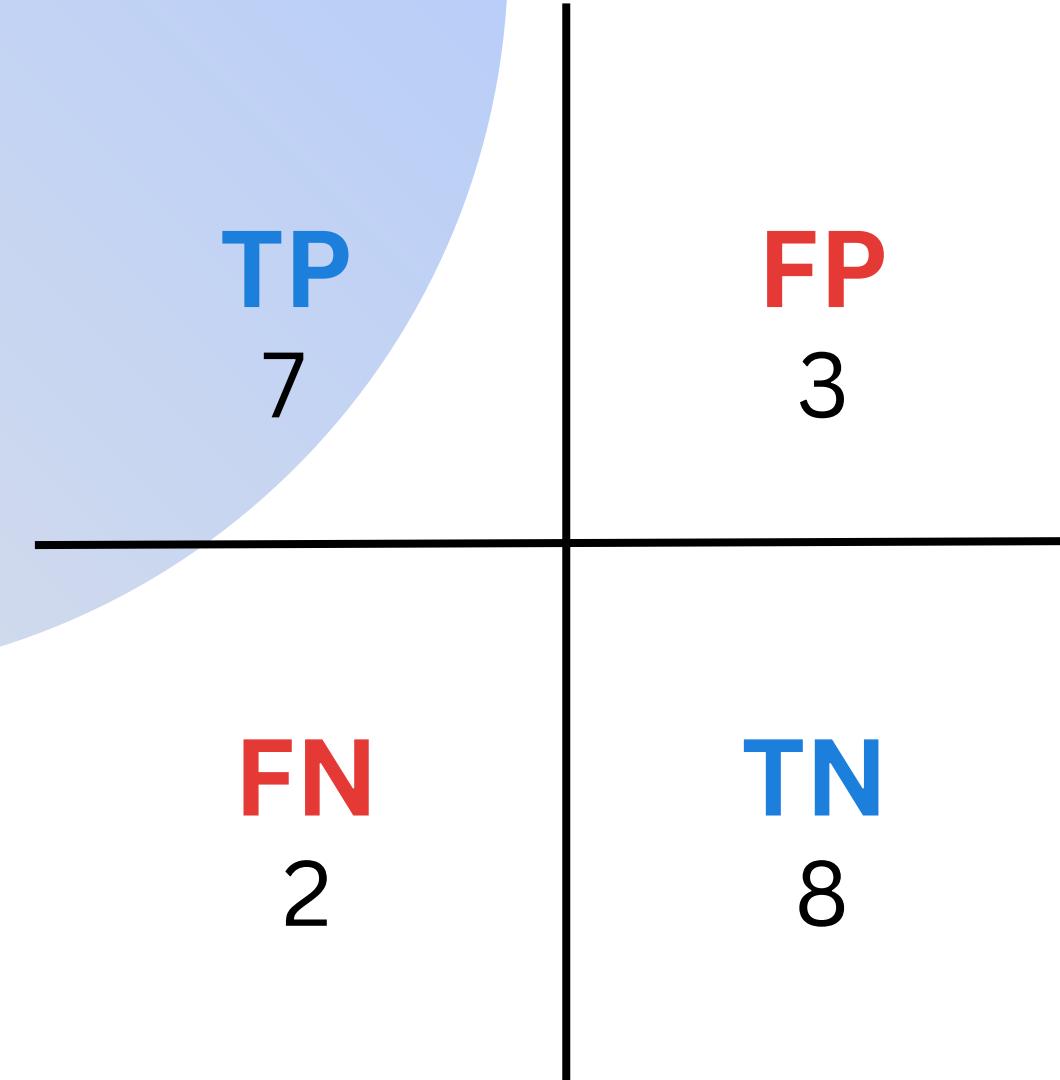
TP ← 양성이라고 예측
실제로 양성인 것 중

4. F1-score

: 재현율과 정밀도의 조화평균

- TN (True Negative) : 실제가 음성인데 음성으로 **잘 예측**한 것
- FP (False Positive) : 실제가 음성인데 양성으로 **잘못 예측**한 것
- FN (False Negative) : 실제가 양성인데 음성으로 **잘못 예측**한 것
- TP (True Positive) : 실제가 양성인데 양성으로 **잘 예측한** 것

$$\begin{aligned}
 \text{F1-Score} &= 2 \times \frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{Recall}}} \\
 &= 2 \times \frac{\text{precision} \times \text{Recall}}{\text{precision} + \text{Recall}} = 2 \times \frac{\text{정밀도} \times \text{재현율}}{\text{정밀도} + \text{재현율}}
 \end{aligned}$$



퀴즈 !

1. 정확도는 ?

(hint : 전체 데이터에서 정답을 맞춘 비율)

2. 정밀도는 ?

(hint : 모델이 양성으로 예측한 것 중 실제 정답이 양성인 비율)

3. 재현율은?

(hint : 실제 정답이 양성인 것 중 모델이 양성이라고 예측한 비율)

정확도(accuracy)

가장 널리 사용되는 평가지표지만,
데이터가 불균형할 시 모델의 실제 성능을 과대평가하거나 과소평가 할 수 있음



정밀도와 재현율

trade-off 관계(한 지표를 높이려고 하면, 다른 지표가 낮아지는 경향이 있기 때문에
어떤 것을 더 중요시할지 결정해야 함)



F1 score

정밀도와 재현율을 얼마나 균형있게 고려하고 있는지를 나타내는 지표로,
이 지표값이 높을 수록 정밀도와 재현율 모두 높아 모델 성능이 우수함을 의미함

즉, **모델의 목적에 따라 적절한 평가지표를 선택하고,**
그 각 지표가 가지는 의미와 한계를 이해하는 것이 중요함!

그럼 어떤 상황에서 어떤 평가지표를 사용하는 것이 좋을까?

정확도가 중요지표인 경우

데이터 불균형이 크지 않은 일반적인 경우

정밀도가 중요지표인 경우

실제 음성을 양성으로 잘못 판단하게 되면 업무상 큰 영향이 발생하는 경우
(ex) 일반메일을 스팸메일로 분류할 경우 필요한 메일을 받지 못하는 업무상 차질이 생김)

$TP/TP+FP$

재현율이 중요지표인 경우

실제 양성을 음성으로 잘못 판단하게 되면 업무상 큰 영향이 발생하는 경우
(ex) 암환자(양성)을 음성으로 진단했을 경우 생명이 달린 문제가 발생)

$TP/TP+FN$

여태까지 배운 단어들로 빙고 게임을 해봅시다.

사이트 : <https://pushoong.com>

4X4 빙고판으로 만드세요!

머신러닝, 그룹화, 정규화, 표준화, 이상치, 결측값, 상관계수, 히스토그램, 전처리, 머신러닝, 랜덤포레스트, svm, knn, 정확도, 정밀도, 재현율, f1-score, TP, TN, FN, FP, 평가지표, 감마, 마진, 파라미터, 과적합, 피쳐, 레이블, 훈련데이터, 테스트데이터

Thank You

+123-456-7890

123 ANYWHERE ST., ANY CITY

REALLYGREATSITE.COM