# Python을 활용한 머신러닝 실무

인선미

2026-02-05 ~ 02-10

# 분류

# 지도학습 : 분류 (Classification)

분류(Classification) 알고리즘은 주어진 데이터를 모델에 적용한 후
**범주**(카테고리) 중 하나의 값으로 분류하여 예측

사용자가 좋아요를 누른 데이터 (예)

| 관심 작품 | 장르 | 유형 | 좋아요 |
|---|---|---|---|
| 셜록 | 추리 | 드라마 | ♥ |
| 지정생존자 | 스릴러 | 드라마 | ♥ |
| 빨간 머리 앤 | 가족 | 드라마 | |
| 이웃집 토토로 | 가족 | 영화 | |
| 시그널 | 스릴러 | 드라마 | ♥ |

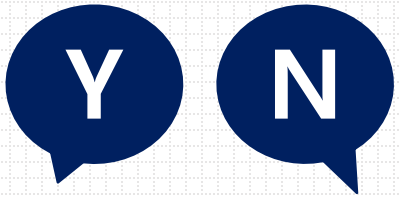
- 작품에 대한 장르와 유형에 따른 사용자의 좋아요 **데이터를 수집**
- 학습 데이터 내에 예측하고자 하는 결과 항목이 **명시적으로 표시되어 있음** (**labeled data**)
- 결과 예측 :
  - 입력값 : 장르 + 유형
  - 결과값(= 예측하고자 하는 항목) : 좋아요

  → **binary classification**

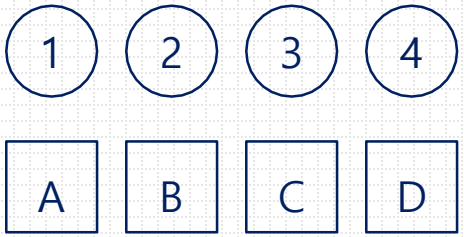
# 지도학습 : 분류 (Classification)

분류(Classification)는 예측하고자 하는 범주의 개수에 따라 **2진 분류**와 **다중 클래스 분류**로 구분됨.
스팸 메일 필터링, 질병 진단 등 다양한 분야에 적용되고 있음.

## 분류 (Classification)




- 2진 분류
- 2클래스 분류




- 다중 클래스 분류

## 예시

- 스팸 메일 필터링

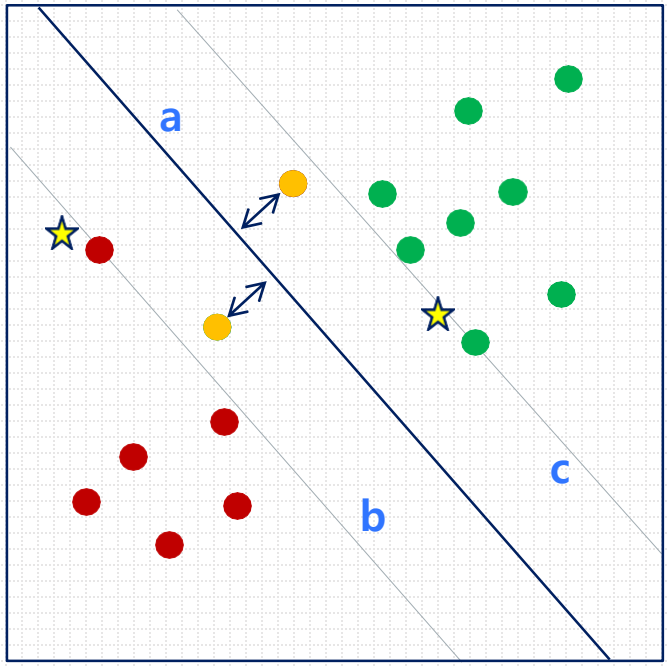
- 질병 진단

- 서포트 벡터 머신 (SVM)
- 의사 결정 나무 (Decision Tree)
- 로지스틱 회귀 등

# 지도학습 : 분류 (Classification)

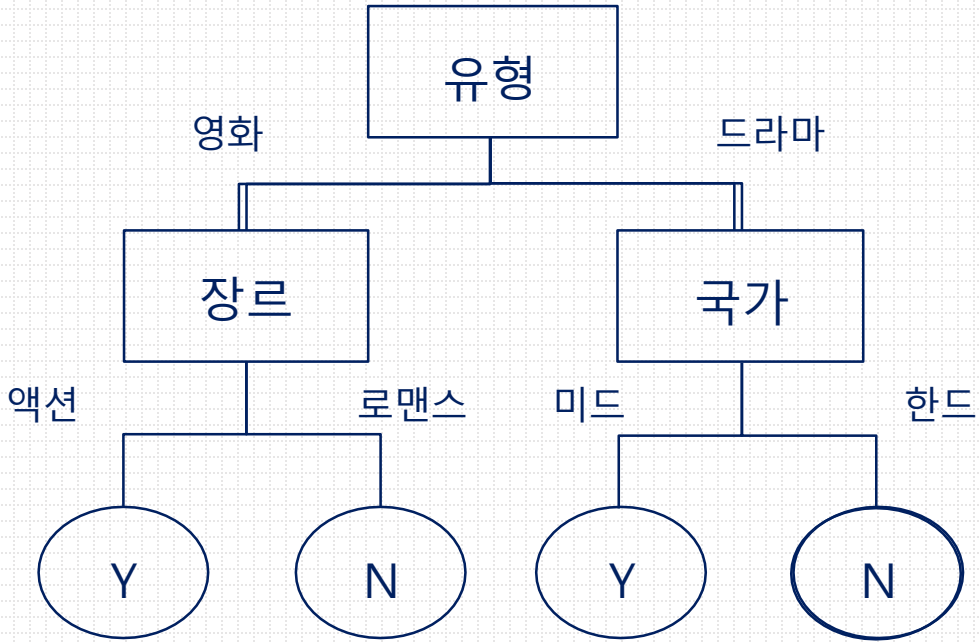
대표적 알고리즘 : 서포트 벡터 머신, 의사 결정 나무, 로지스틱회귀 등.

## 서포트 벡터 머신(SVM)




- 주어진 샘플 데이터들을 구분하는 **최적의 분할선** 탐색

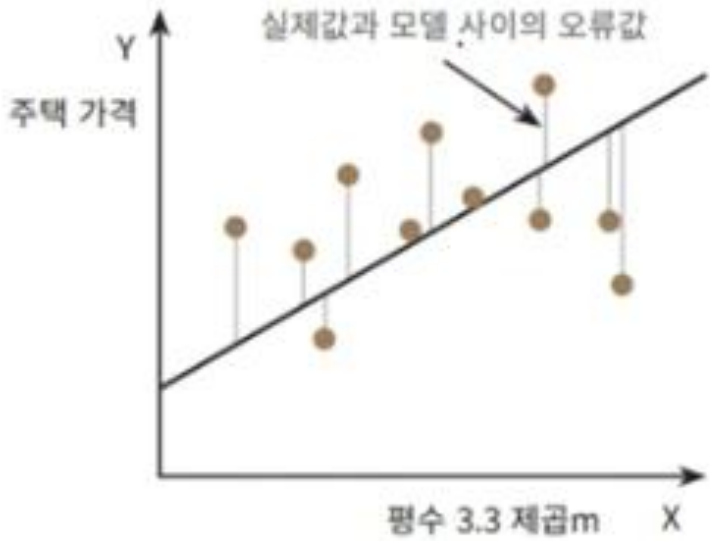
## 의사 결정 나무 (Decision Tree)




- 2클래스 질문에 따라 정답을 찾아 학습하는 알고리즘

# 로지스틱 회귀 Logistic Regression

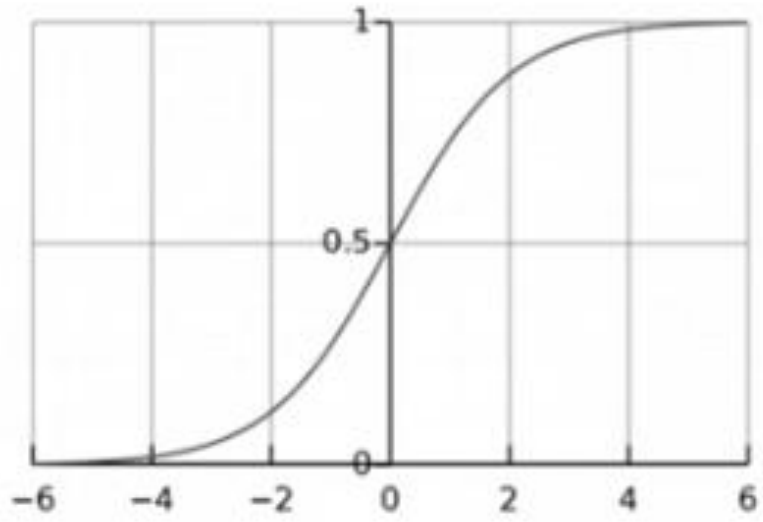
✓데이터가 어떤 범주에 속할 확률을 0에서 1 사이의 값으로 예측하고, 그 확률에 따라 특정 클래스로 분류하는 모델




〈 선형 회귀의 선형 함수 〉

시그모이드 함수 (Sigmoid Function)

$$f(z) = \frac{1}{1 + e^{-z}}$$



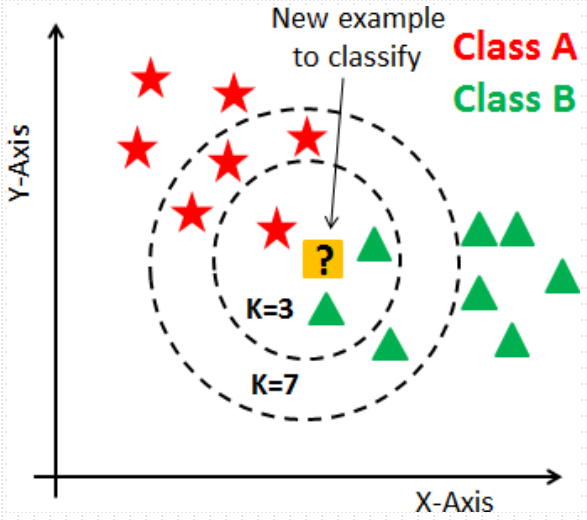

〈 로지스틱 회귀의 시그모이드 함수 〉

# K-Nearest Neighbors, KNN 알고리즘

✓기본 원리

✓학습 과정 없이 훈련 데이터에서 가장 가까운 K개의 이웃을 찾아 그들의 정보를 기반으로 분류나 회귀 수행

✓새로운 데이터 포인트로부터 기존 데이터들 사이의 거리를 계산
→ 가장 가까운 K개의 이웃 선택
→ 다수결(Majority Vote)로 결과 결정.




✓K값의 결정(Hyperparameter)

✓K가 너무 작을 때 (예: K=1): 모델이 너무 민감해져 과적합(Overfitting)될 위험이 크다. 노이즈에 취약

✓K가 너무 클 때: 결정 경계가 뭉툭해지며 과소적합(Underfitting)될 수 있다. 데이터의 세세한 특성을 무시

✓Tip: 일반적으로 데이터 개수의 제곱근($\sqrt{N}$) 근처에서 결정하거나, 짝수일 경우 투표 결과가 동점이 될 수 있어 홀수로 설정하는 것이 관례

# K-Nearest Neighbors, KNN 알고리즘

✓ KNN의 장단점

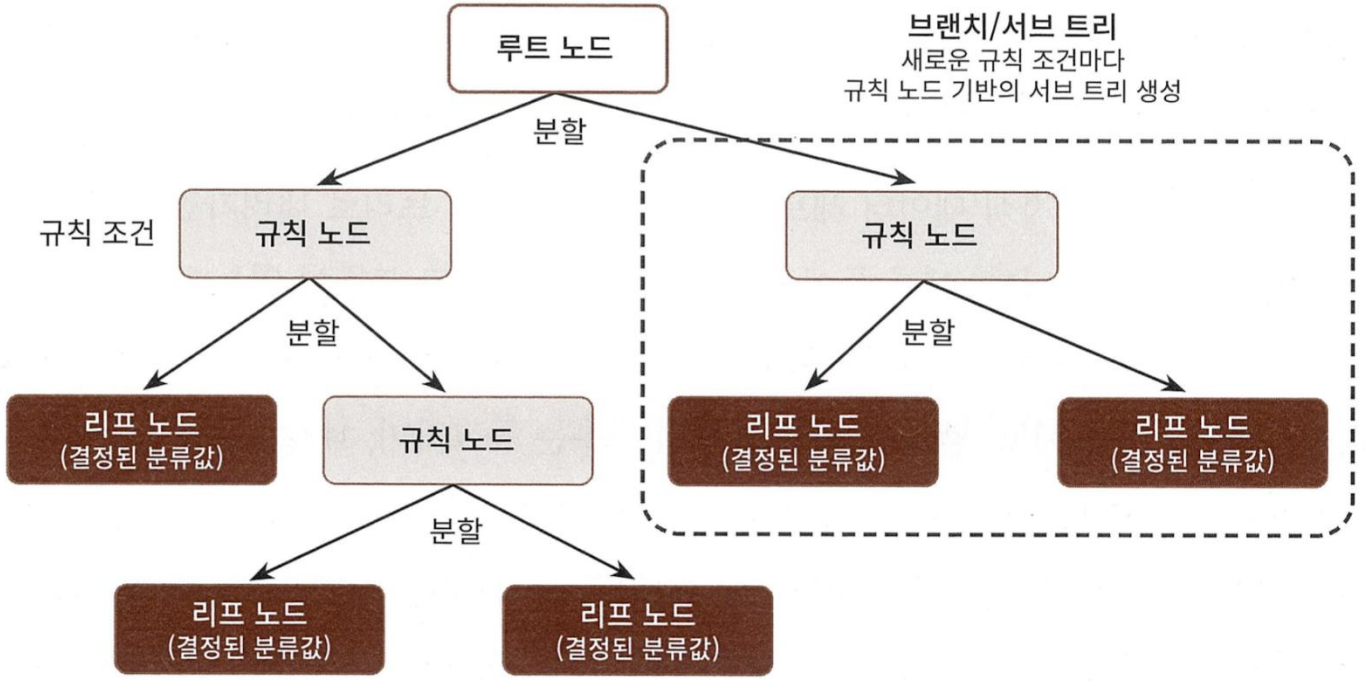
| 장점 | 단점 |
| --- | --- |
| 알고리즘이 매우 단순하고 이해하기 쉬움 | 데이터가 많아지면 거리 계산 속도가 급격히 느려짐 (메모리 집약적) |
| 수치형 데이터에 대해 별도의 가정이 필요 없음 | **차원의 저주:** 변수가 많아지면 성능이 저하됨 |
| 다중 분류도 자연스럽게 처리 가능 | 거리 기반이므로 데이터 스케일링(Normalization)이 필수적 |

# Decision Tree (의사결정 트리)

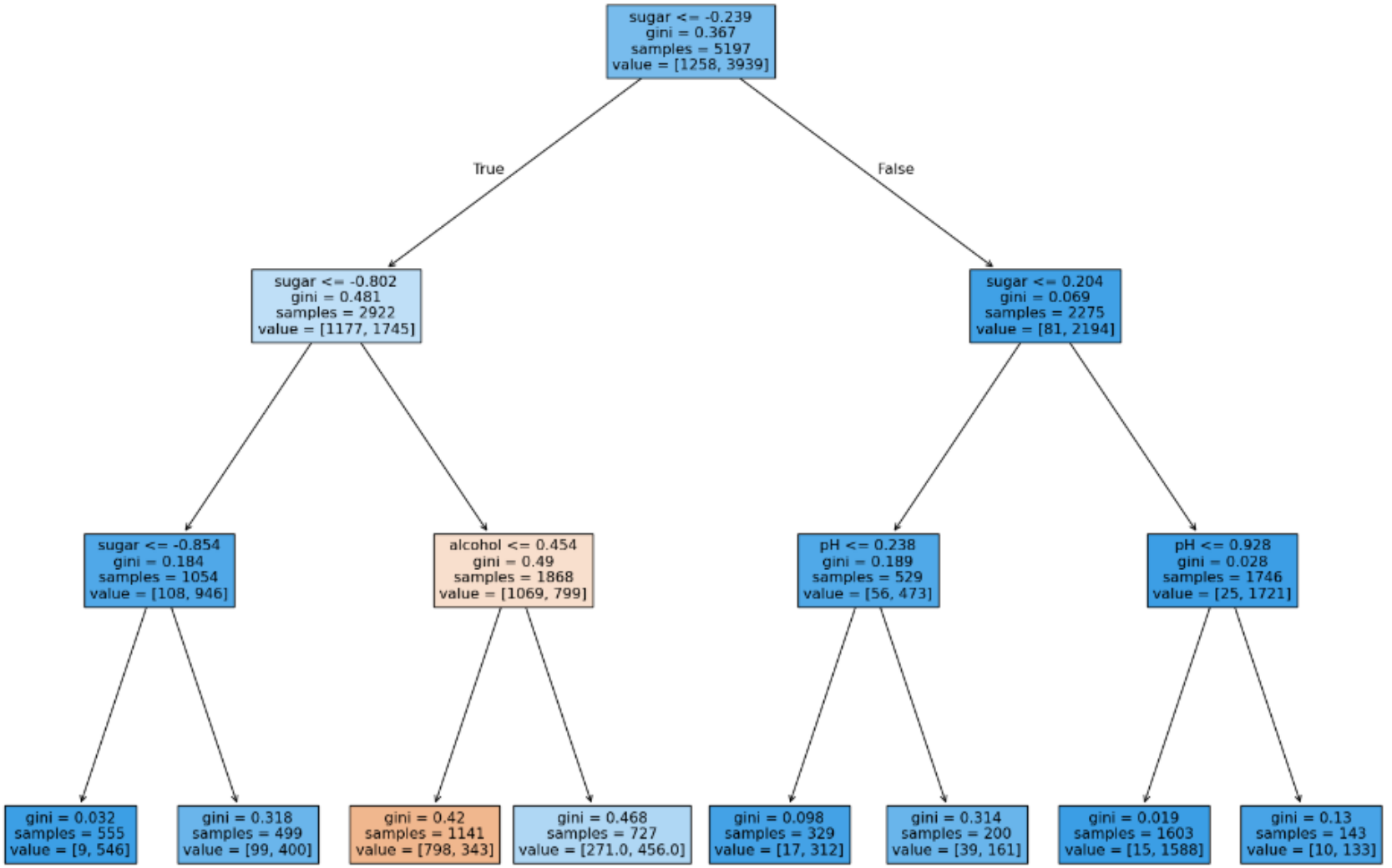
✓"스무고개"처럼 데이터에 질문을 던져 선택지를 좁혀가는 직관적인 알고리즘

✓데이터의 특성을 바탕으로 분기를 만들어 트리 구조로 의사결정 과정을 모델링

✓Root node에서 시작하여 조건에 따라 데이터를 분할하고, 이 과정을 반복하여 Leaf node에서 예측 결과를 도출



# Decision Tree (의사결정 트리)



# Decision Tree (의사결정 트리)

## ✓분할과 순수도

✓ 의사결정트리의 목표는 데이터를 가장 잘 구분할 수 있는 질문을 던져서, 결과적으로 나뉜 집단 내의 순수도(Purity)를 높이고 불순도(Impurity)를 낮추는 것

✓ **루트 노드(Root Node):** 전체 데이터가 시작되는 지점 (가장 중요한 질문).

✓ **리프 노드(Leaf Node):** 더 이상 나눠지지 않는 최종 결정 값(클래스 또는 수치).

✓ **분할 기준:** 어떤 질문이 좋은지 판단하기 위해 다음과 같은 지표 사용

✓ **불순도 측정 지표**

- **지니 계수 (Gini Index):** 데이터의 통계적 분산 정도를 측정. 0일 때 가장 순수. (Scikit-learn 기본값)
- **엔트로피 (Entropy):** 정보 이론에서의 무질서도를 측정, 정보 이득(Information Gain)이 최대화되는 방향으로 분할

# Decision Tree (의사결정 트리)

## ✓알고리즘의 주요 특징

### ✓장점

- **해석력(Explainability):** 결과가 시각화되어 있어 비전문가에게 설명하기 매우 좋음. (화이트박스 모델)
- **전처리 간소화:** 데이터 스케일링(Normalization)이나 정규화 작업이 성능에 거의 영향을 주지 않는다.
- **혼합 데이터 처리:** 수치형 변수와 범주형 변수를 동시에 처리 가능.

### ✓단점

- **과적합(Overfitting):** 트리가 너무 깊어지면 학습 데이터에만 과하게 맞춰져 새로운 데이터에 대한 성능이 떨어진다.
- **불안정성:** 데이터의 작은 변화에도 트리 구조가 크게 바뀔 수 있다.

# Decision Tree (의사결정 트리)

- ✓가지치기 (Pruning): 과적합 해결책
  - ✓ 트리가 끝없이 자라지 않도록 제어하는 기법입니다.
  - ✓ 사전 가지치기(Pre-pruning): 트리의 최대 깊이(max_depth), 노드 분할을 위한 최소 샘플 수 등을 미리 제한
  - ✓ 사후 가지치기(Post-pruning): 일단 트리를 다 만든 후, 영향력이 적은 가지를 제거

# 앙상블 학습

✓다양한 분류 알고리즘 및 앙상블 기법 등을 검토하여 적용할 알고리즘을 결정

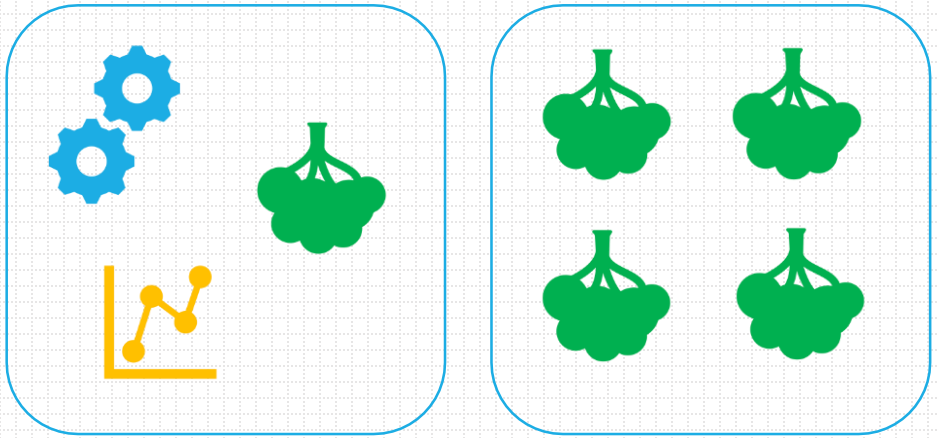
## 다양한 분류 알고리즘

**서포트 벡터 머신**
- 데이터 공간에서 최적의 분할선을 검색

**의사결정 나무**
- 연속적인 예/아니오 질문을 반복해 의사결정

**로지스틱 회귀**
- 로지스틱 함수를 적용하여 0과 1 사이의 값을 산출

## 앙상블 / 랜덤 포레스트



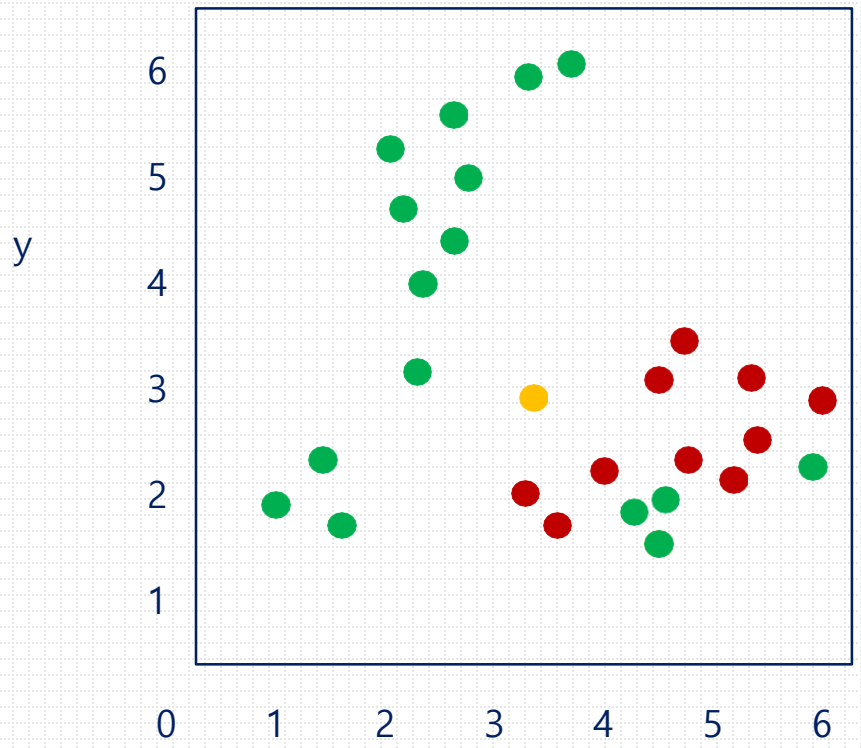
- 여러 머신러닝 모델을 적용
- 여러 의사결정트리를 적용

**랜덤 포레스트 : 단순하면서 강력한 분류 알고리즘으로 평가받고 있음**

# 앙상블 학습

✓앙상블 : 여러 개의 머신러닝 모델로부터 얻은 예측값을 종합하여 분류의 정확도를 높이는 방법 다양한 모델을 사용하여 과대적합 감소 효과 있음

**주어진 데이터 세트**




**앙상블 (예: 다수결 투표)**

# 하이퍼파라미터

✓모델링할 때 사용자가 직접 조정 및 세팅하는 값

✓ 예 : 의사결정나무의 깊이 지정, 샘플링 방식 등

의사결정나무의 최대 깊이는 몇으로 정할까?

랜덤포레스트에 의사결정나무를 몇 개 적용할까?

랜덤포레스트 샘플링 시 중복*을 허용할까?

리프 노드에 포함해야 할 최소 샘플의 수는?

- 의사결정나무 최대 깊이 (max depth) :
  학습 데이터에 과대 적합을 방지

- 랜덤포레스트 샘플링 시 중복 허용
  → 배깅 (bootstrap aggregating) 방식

# 하이퍼파라미터

| 파라미터 | 설명 | 권장 사항 |
|---|---|---|
| n_estimators | 결정 트리의 개수 | 많을수록 성능이 좋아질 수 있으나 속도가 느려짐<br>(보통 100~500) |
| max_depth | 트리의 최대 깊이 | 과적합 방지를 위해 적절히 제한 |
| max_features | 최적의 분할을 위해 고려할 피처 개수 | 보통 sqrt(전체 피처 수)를 사용 |
| n_jobs | 사용할 CPU 코어 개수 | -1로 설정하면 모든 코어를 사용하여 병렬 연산<br>(속도 향상) |

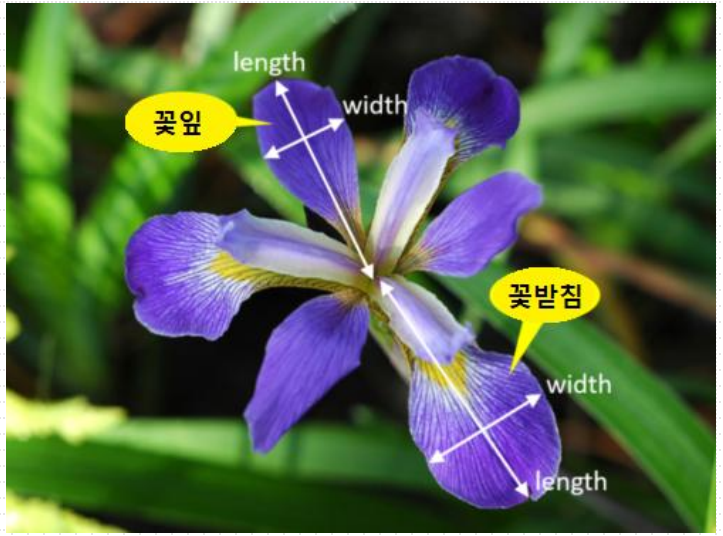
# 의사결정트리 vs 랜덤 포레스트

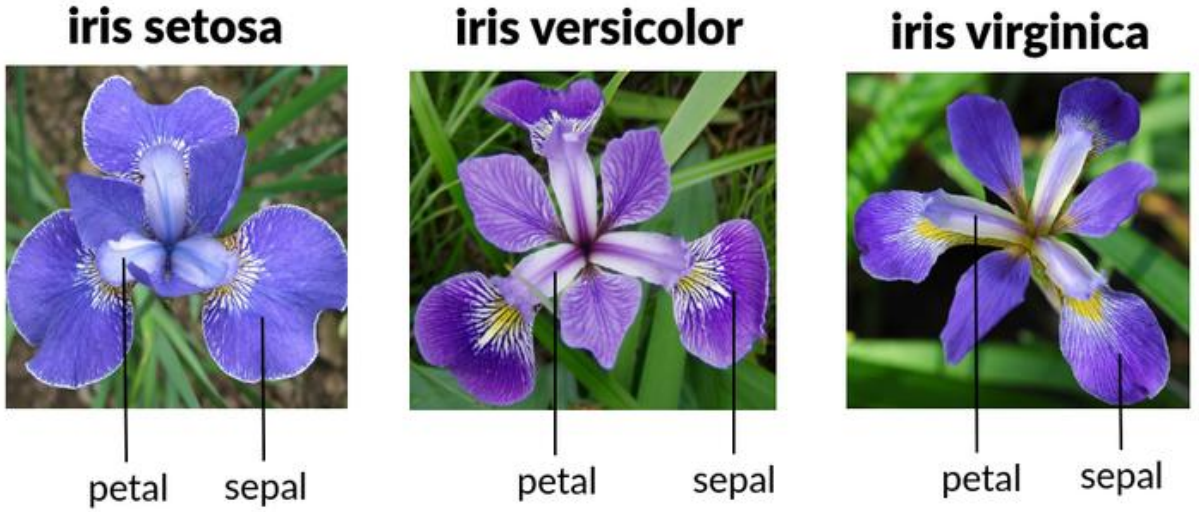
| 구분 | **의사결정트리** (Decision Tree) | **랜덤 포레스트** (Random Forest) |
|---|---|---|
| **모델 유형** | 단일 모델 | 앙상블 모델 (Bagging) |
| **분산**(Variance) | 높음 (데이터 변화에 민감) | 낮음 (여러 트리가 보완) |
| **해석 가능성** | 매우 높음 (시각화 가능) | 낮음 (여러 트리가 섞여 있어 복잡함) |
| **성능** | 보통 | 높음 |

# 사이킷런을 이용하여 머신러닝 모델 만들어보기

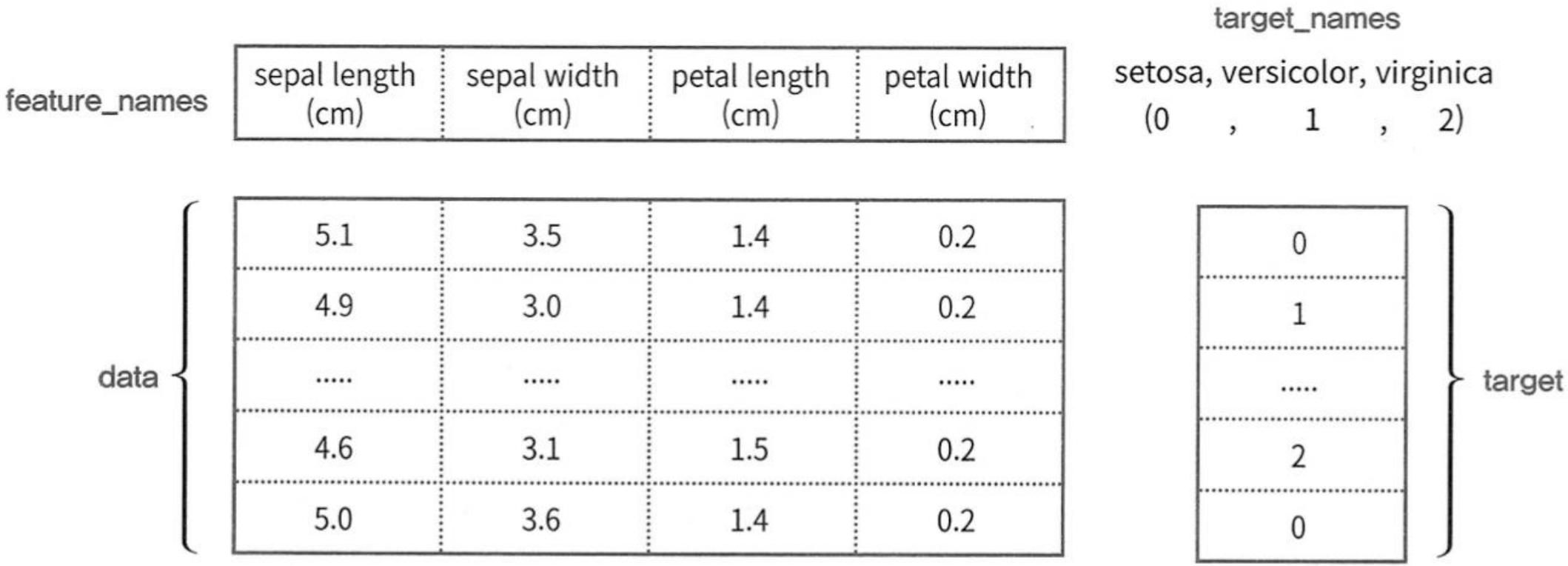
✓붓꽃 데이터 세트로 붓꽃의 품종을 분류(classification)

✓꽃잎의 길이와 너비, 꽃받침의 길이와 너비 피쳐를 기반으로 꽃의 품종을 예측



# 붓꽃 예측하기



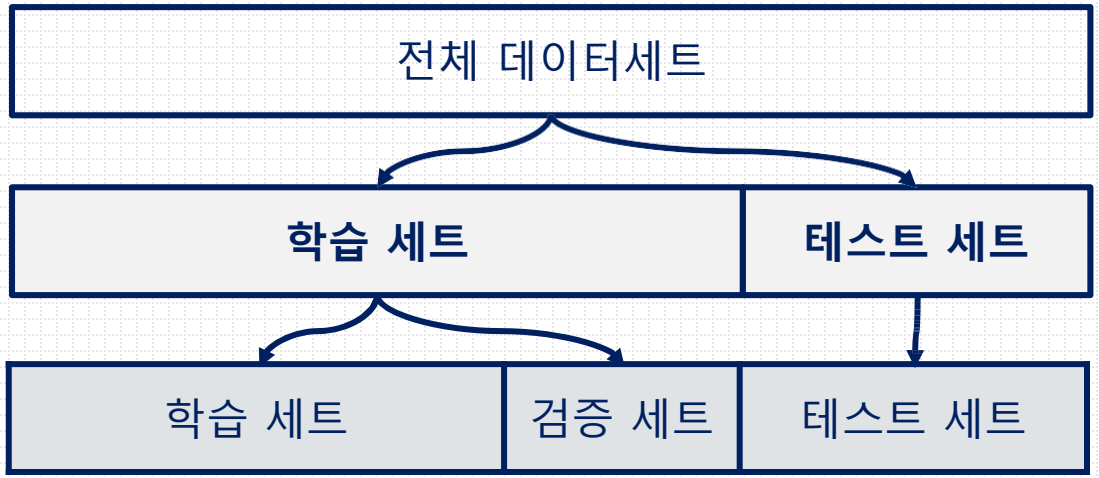
# train_test_split()

✓ 지도학습에서는 주어진 데이터를 데이터의 특징을 나타내는 입력(input, feature)과 예측하고자 하는 정답을 나타내는 타겟(target, label)으로 구분

데이터 세트 = 입력 (input) + 타겟 (target)

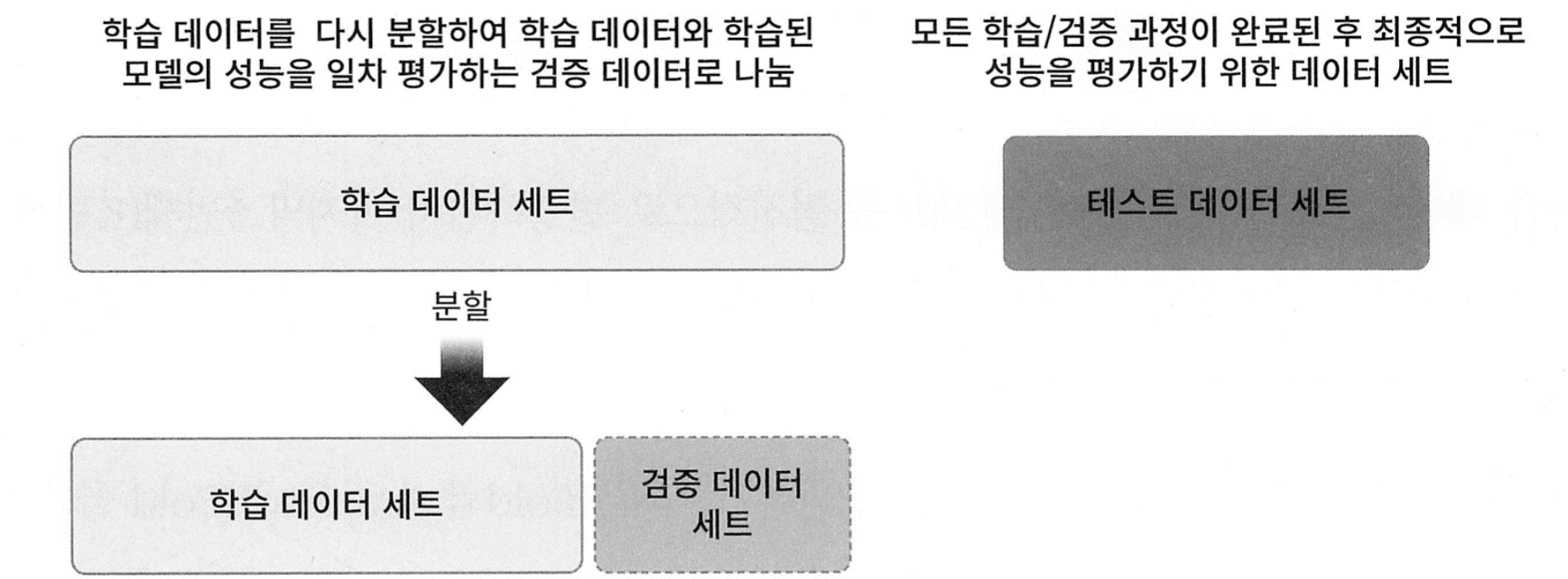
| 온도 | 강수량 | . | 발사 |
|---|---|---|---|
| 21.3 | 105 | ... | N |
| 18.6 | 30 | ... | N |
| 25.9 | 5 | ... | Y |
| 24.7 | 0 | ... | Y |
| 13.6 | 70 | ... | N |

- 지도학습에서는 주어진 데이터 전체 샘플을 데이터가 가진 개별 특성(feature)을 나타내는 입력(input)과 정답 데이터인 타겟(target)으로 구분

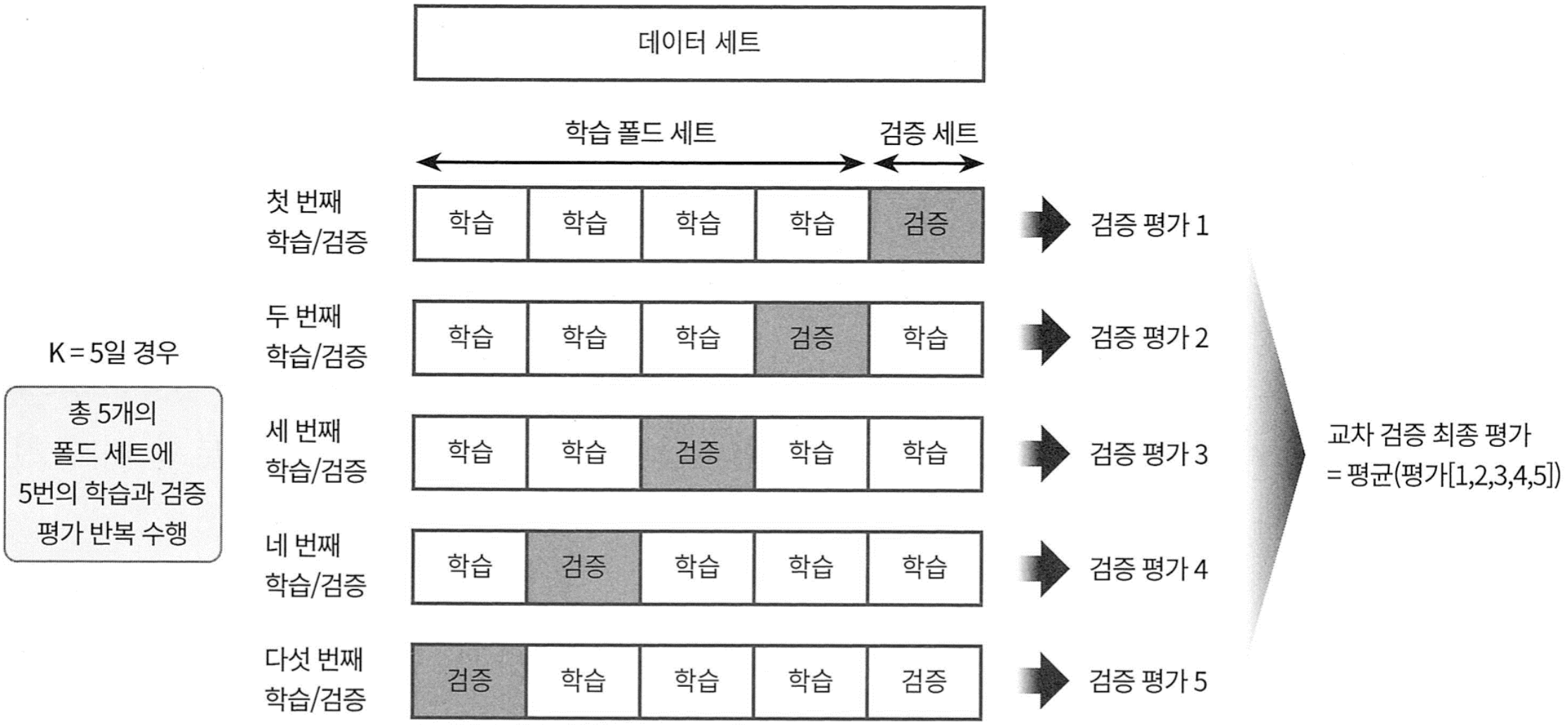


# 교차검증

✓교차 검증(Cross Validation)은 모델의 학습 과정에서 **데이터를 여러 번 나누어 학습과 평가를 반복**함으로써 모델의 일반화 성능을 높이고 과적합(Overfitting)을 방지하는 기법



# K 폴드 교차 검증



# cross_val_score()

- ✓ 교차 검증(Cross Validation)의 과정을 한 줄의 코드로 구현해주는 Scikit-learn의 편리한 도구
  - ✓ estimator: 알고리즘 클래스
  - ✓ X: 피처 데이터 세트
  - ✓ y: 레이블 데이터 세트
  - ✓ scoring: 예측 성능 평가 지표
  - ✓ cv: 교차 검증 폴드 수

# GridSearchCV

✓ 머신러닝 모델의 성능을 최적화하기 위해 필수적인 과정이 바로 **하이퍼파라미터 튜닝**, GridSearchCV는 이를 자동화해주는 가장 대표적인 도구

✓ **"격자(Grid) + 교차 검증(CV)"**

- ✓ **Grid Search (격자 탐색):** 사용자가 탐색할 파라미터 후보군들을 리스트로 전달하면, 가능한 **모든 조합**을 마치 격자판을 채우듯 하나씩 대입하며 테스트
- ✓ **Cross Validation (교차 검증):** 각 조합마다 데이터를 여러 번 쪼개어 학습과 검증을 반복(K-Fold 등). 이를 통해 특정 데이터셋에만 잘 맞는 과적합을 방지하고 모델의 **일반화 성능**을 측정

```python
from sklearn.model_selection import GridSearchCV

grid_search = GridSearchCV(
    estimator=model,            # 학습시킬 모델 (RF, SVM 등)
    param_grid=param_grid,      # 탐색할 파라미터 사전 (dict)
    cv=5,                       # 교차 검증 폴드 수 (K)
    scoring='accuracy',         # 평가 지표 (정확도, F1-score 등)
    n_jobs=-1                   # 모든 CPU 코어 사용 (병렬 연산)
)
```

# 성능 평가 지표(Evaluation Metric)

## ✓분류의 성능 평가 지표

- ✓정확도(Accuracy)
- ✓오차행렬(Confusion Matrix)
- ✓정밀도(Precision)
- ✓재현율(Recall)
- ✓F1 스코어
- ✓ROC AUC

# 성능 평가 지표(Evaluation Metric)

✓정확도는 실제 데이터에서 예측 데이터가 얼마가 같은지를 판단하는 지표

$$\text{정확도}(Accuracy) = \frac{\text{예측 결과가 동일한 데이터 건수}}{\text{전체 예측 데이터건수}}$$

# 성능 평가 지표(Evaluation Metric)

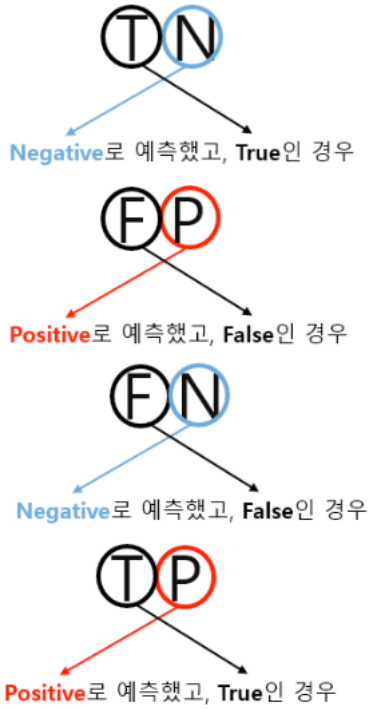
✓Confusion Matrix( 혼동 행렬, 오차 행렬)

✓학습된 분류 모델이 예측을 수행하면서 얼마나 헷갈리고(confused) 있는지를 보여주는 지표

$$\text{정확도} = \frac{\text{예측 결과와 실제 값이 동일한 건수}}{\text{전체 데이터 수}} = \frac{FN + TP}{TN + FP + FN + TP}$$

$$\text{정밀도} = \frac{TP}{FP + TP}, \quad \text{재현율} = \frac{TP}{FN + TP}$$

| Actual Class \ Predicted Class | Negative | Positive |
|---|---|---|
| Negative | **TN** (True Negative)<br>음성으로 예측했고 실제 범주는 음성인 경우의 수 | **FP** (False Positive)<br>양성으로 예측했지만 실제 범주는 음성인 경우의 수 |
| Positive | **FN** (False Negative)<br>음성으로 예측했지만 실제 범주는 양성인 경우의 수 | **TP** (True Positive)<br>양성으로 예측했고 실제 범주가 양성인 경우의 수 |



# Confusion Matrix( 혼동 행렬, 오차 행렬)

| | | Predicted: Negative (0) | Predicted: Positive (1) | |
|---|---|---|---|---|
| Actual | Negative (0) | **True Negative** **TN** | **False Positive** **FP** (Type I error) | **Specificity** $= \frac{TN}{TN + FP}$ |
| | Positive (1) | **False Negative** **FN** (Type II error) | **True Positive** **TP** | **Recall**, **Sensitivity**, True positive rate (**TPR**) $= \frac{TP}{TP + FN}$ |
| | | **Accuracy** $= \frac{TP + TN}{TP + TN + FP + FN}$ | **Precision**, Positive predictive value (**PPV**) $= \frac{TP}{TP + FP}$ | **F1-score** $= 2 \times \frac{Recall \times Precision}{Recall + Precision}$ |

# 분류 모델의 성능 평가 지표

✓ROC 커브 (Receiver Operating Characteristic Curve)

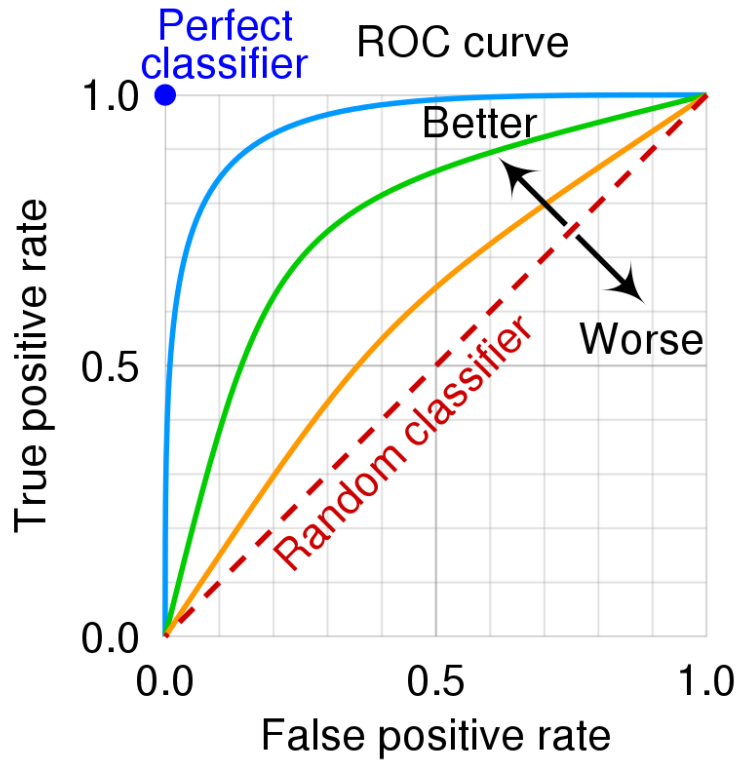
✓임계값(0~1)을 변화시킴에 따라 모델의 진짜 양성 비율(TPR)과 거짓 양성 비율(FPR)이 어떻게 변하는지를 나타낸 그래프

✓**X축: FPR (False Positive Rate)**

- 실제 음성인데 양성으로 잘못 예측한 비율
- $FPR = \frac{FP}{FP+TN}$

✓**Y축: TPR (True Positive Rate)**

- 재현율(Recall)과 같은 지표로,
  실제 양성을 모델이 양성으로 잘 맞춘 비율
- $TPR = \frac{TP}{TP+FN}$

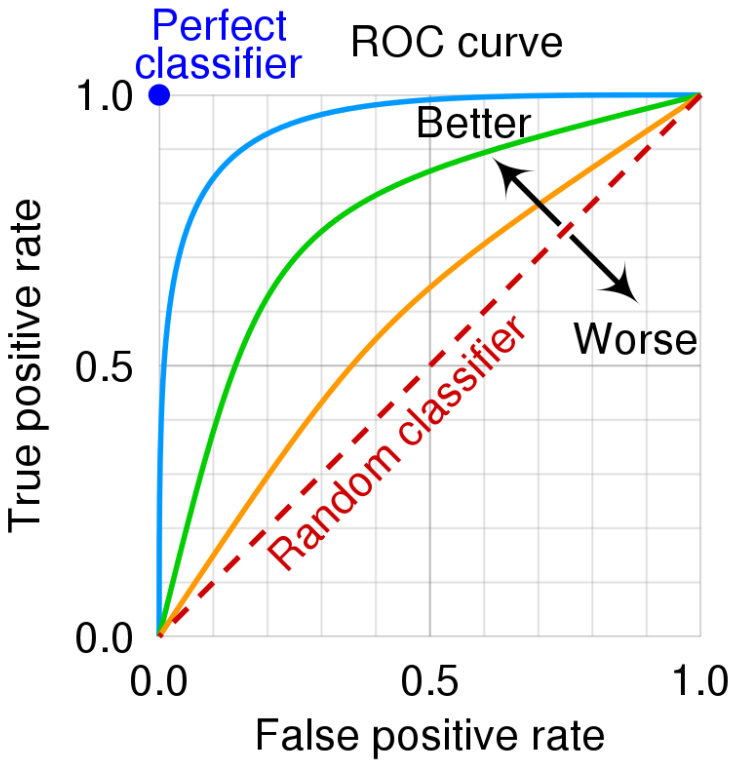



✓**핵심 원리:** 좋은 모델일수록 FPR을 낮게 유지하면서 TPR을 빠르게 높인다.
따라서 그래프가 왼쪽 상단 모서리(Perfect Classifier)에 가까워질수록 우수한 모델

# 분류 모델의 성능 평가 지표

## ✓AUC (Area Under the Curve)

- ✓ROC 곡선 아래의 면적을 수치화한 것. 모델의 성능을 하나의 숫자로 요약할 때 매우 유용.
- ✓**AUC = 1.0:** 완벽한 모델. 어떤 임계값에서도 양성과 음성을 완벽히 분리
- ✓**AUC = 0.5:** 랜덤 추측과 동일한 수준. (그래프상의 대각선 점선)
- ✓**AUC < 0.5:** 랜덤 추측보다 못한 경우
  (보통 모델의 예측을 반대로 뒤집으면 성능이 개선되는 상태).



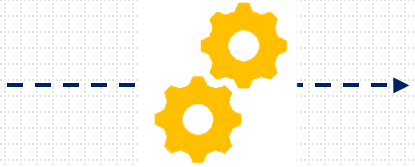
# 로켓 발사 예측 모델 구현

# 로켓 발사 예측 모델 구현

✓날씨 데이터를 활용하여 로켓 발사 예측 모델을 구현해보자

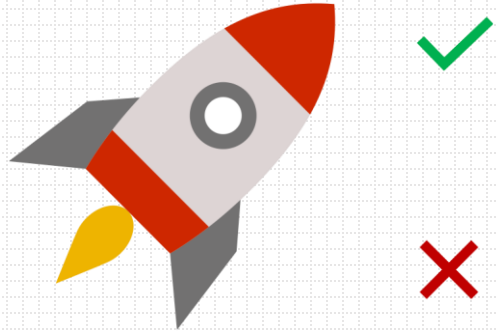
## 날씨 관련 데이터

- 온도, 강수량, 풍향, 풍속, 번개, 구름 등 날씨 데이터



머신러닝

## 로켓 발사 여부 결정



- 로켓 발사 / 발사 연기 의사 결정

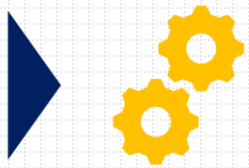
# 배경 : 로켓 발사와 날씨

로켓 발사 관련 데이터를 머신러닝 기법으로 분석.
→ 분석 결과를 활용하여 날씨에 따른 로켓 발사 가능 여부를 예측하고자 함.

**로켓 발사 관련 날씨 데이터**

| | A | B | E | F | G | H |
|---|---|---|---|---|---|---|
| 1 | Name | Date | Crewed or Uncrewed | Launched? | High Temp | Low Temp |
| 293 | | 26-May-20 | | | 85 | 74 |
| 294 | Space X Dragon | 27-May-20 | Crewed | N | 86 | 70 |
| 295 | | 28-May-20 | | | 87 | 71 |
| 296 | | 29-May-20 | | | 86 | 72 |
| 297 | | 28-May-20 | | | 87 | 71 |
| 298 | | 29-May-20 | | | 86 | 72 |
| 299 | Space X Dragon | 30-May-20 | Crewed | Y | 87 | 75 |
| 300 | | 31-May-20 | | | 87 | 72 |
| 301 | | 01-Jun-20 | | | 85 | 76 |

**로켓 발사 가능 여부 예측**



오늘 날씨에 로켓을 발사해도 괜찮을까?

- 로켓 발사일 근처의 온도, 강수량, 풍속 등의 관련 날씨 데이터 분석

# 로켓 발사 데이터

**기본**

- 이름 (Name)
- 날짜 (Date)
- 시간 (Time)
- 장소 (Location)
- 유무인 (Crewed / Uncrewed)
- **발사 여부 (Launched)**

**온도**

- 최고 온도 (High Temp)
- 최저 온도 (Low Temp)
- 평균 온도 (Ave Temp)
- 발사 시 온도 (Temp at Launch)
- 과거 최고 온도 (Hist High Temp)
- 과거 최저 온도 (Hist Low Temp)
- 과거 평균 온도 (Hist Ave Temp)

**강수량**

- 발사 시 강수량 (Precipitation)
- 과거 평균 강수량 (Hist Ave Precipitation)

**바람**

- 풍향 (Wind Direction)
- 최고 풍속 (Max Wind Speed)
- 발사 시 풍속 (Wind Speed at Launch)
- 가시성 (Visibility)
- 과거 평균 최고 풍속 (Hist Ave Max Wind Speed)
- 과거 평균 가시성 (Hist Ave Visibility)

**조건**

- 맑음 (Fair) / 약간 흐림 (Partly Cloudy)
- 많이 흐림 (Mostly Cloudy) / 흐림 (Cloudy)
- 천둥 번개 (T-storm)

**기타**

- 해수면 압력 (Sea Level Pressure)
- 과거 평균 해수면 압력 (Hist Ave Sea Level Pressure)
- 낮의 길이 (Day Length)

# 감사합니다 ^^