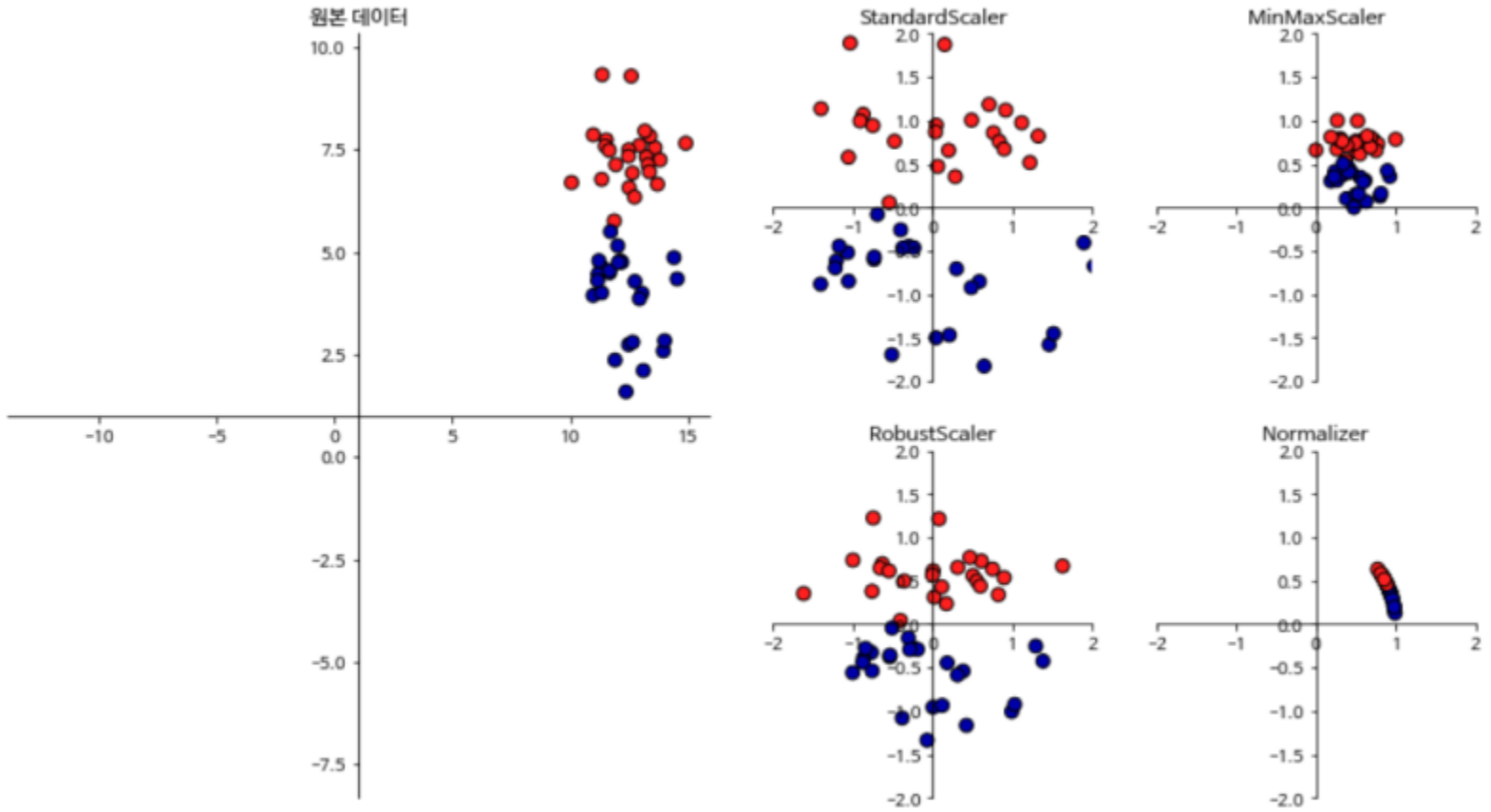
AI 2025-1 6주차

결정트리, 앙상블

KNN 수행하기 위해 필요한 것.



Overview

- 분류: 이산 값 대상 예측
) 이진 분류: 이진 값 대상 예측
) 다중 클래스 분류: 이산(> 2) 값 대상 예측

- 이진 분류의 예
) 다양한 증상의 존재 또는 부재를 감안하여 환자가 질병을 앓고 있는지 예측
) 이메일을 스팸 또는 비스팸으로 분류
) 금융 거래가 사기인지 예측

Overview

이진 선형 분류

분류: D차원 입력 $x \in R^D$가 주어지면 이산 값 대상을 예측합니다.

이진: 이진 대상 $t \in \{0, 1\}$을 예측합니다.

) $t = 1$인 학습 예제를 양성 예제라고 하고, $t = 0$인 학습 예제를 음성 예제라고 합니다. 죄송합니다.

) $t \in \{0, 1\}$ 또는 $t \in \{-1, +1\}$은 계산 편의를 위한 것입니다.

선형: 모델 예측 y는 x의 선형 함수이며, 그 뒤에 임계값 r이 붙습니다.

$$z = \mathbf{w}^T\mathbf{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq r \\ 0 & \text{if } z < r \end{cases}$$

Loss Functions

- 대신: 손실 함수를 정의한 다음 결과 비용 함수를 최소화하려고 시도합니다.

) 기억하세요: 비용은 훈련 세트에 대한 평균(또는 합산) 손실입니다.
- 겉보기에 명백한 손실 함수: 0-1 손실

$$L_{0-1}(y, t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases}$$

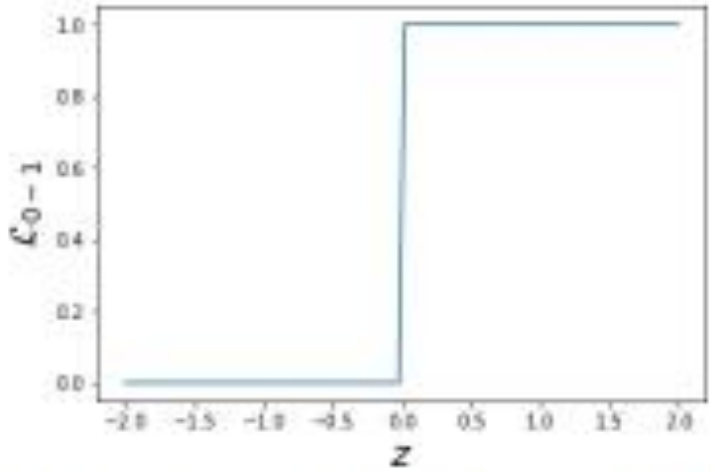
$$= \mathbb{I}[y \neq t]$$

Attempt 1: 0-1 loss

- 함수의 최소값은 임계점에 있을 것입니다. 0-1
- 손실의 임계점을 찾아보도록 합시다.
- 체인 규칙:

$$\frac{\partial L_{0-1}}{\partial w_j} = \frac{\partial L_{0-1}}{\partial z}\frac{\partial z}{\partial w_j}$$

- 하지만 ∂L0−1/∂z는 정의된 모든 곳에서 0입니다!



) ∂L0−1/∂wj = 0은 가중치를 아주 작은 양으로 변경해도 손실에 아무런 영향이 없다는 것을 의미합니다.

) 거의 모든 지점의 기울기가 0입니다!

Attempt 3: Logistic Activation Function

- [0, 1] 밖의 값을 예측할 이유는 분명히 없습니다. y를 이 구간에 압축해 보겠습니다.

- 로지스틱 함수는 일종의 시그모이드 또는 S자 모양 함수입니다.

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

- *$\sigma^{-1}(y) = \log(y/(1-y))$를 로짓이라고 합니다.로지스틱 비선형성을 가진*
- *선형 모델은 로그선형이라고 합니다.*

$$z = \mathbf{w}^T\mathbf{x}$$

$$y = \sigma(z)$$

$$L_{SE}(y, t) = \frac{1}{2}(y - t)^2.$$

- 이런 식으로 사용된 σ를 활성화 함수라고 한다.

Logistic Regression

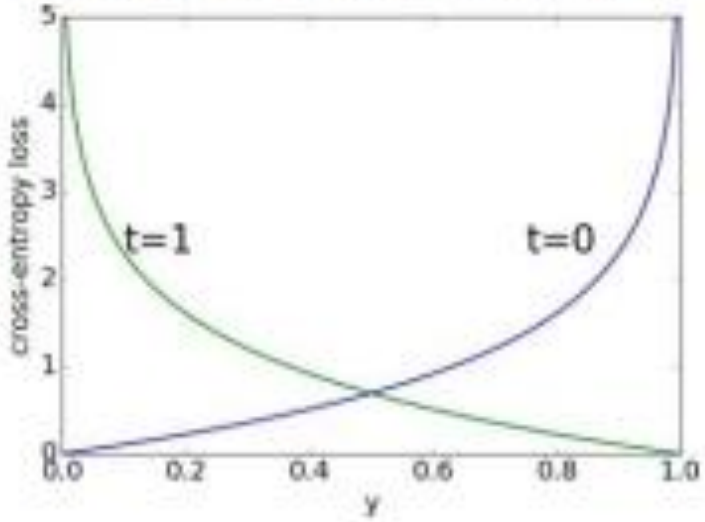
$y \in [0, 1]$이기 때문에 $t = 1$일 추정 확률로 해석할 수 있습니다. $t = 0$이면 $y \approx 1$에 큰 페널티를 주고 싶습니다.

클린턴이 이길 것이라고 99% 확신했던 전문가들은 90% 확신했던 전문가들보다 훨씬 더 틀렸습니다.

크로스 엔트로피 손실(일명 로그 손실)은 이러한 직관을 포착합니다.

$$L_{CE}(y, t) = \begin{cases} -\log y & \text{if } t = 1 \\ -\log(1 - y) & \text{if } t = 0 \end{cases}$$

$$= -t \log y - (1 - t) \log(1 - y)$$



라쏘와 릿지

L1 regularization on least squares:

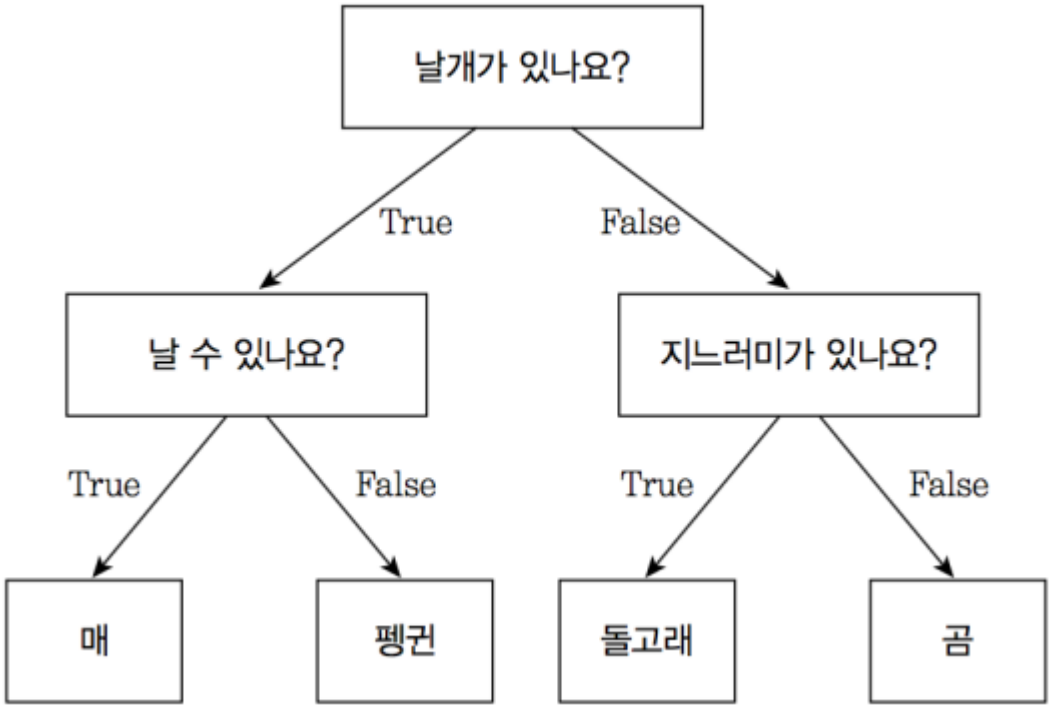
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

L2 regularization on least squares:

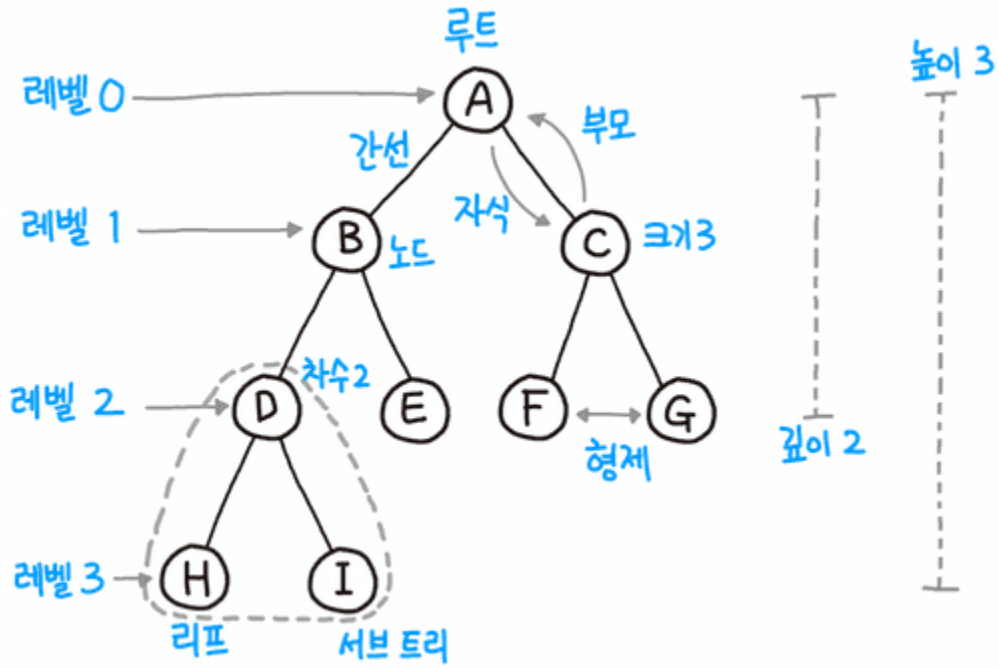
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

결정 트리?

- 스무고개를 하는 방식과 유사
- 다양한 질문을 통해 문제를 해결 하는 방식
- 1차원 적으로 그리는 것이 아닌, 다차원의 선분처럼 나타낼 수 있다.
  - 섬세하다



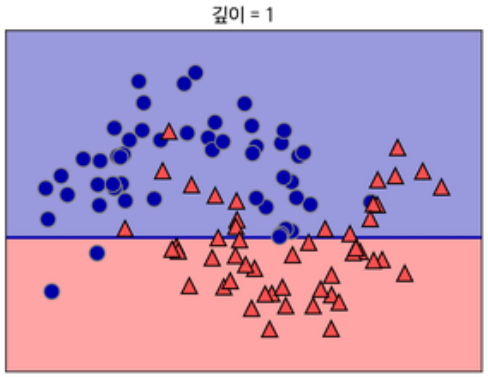
결정 트리의 구조는?

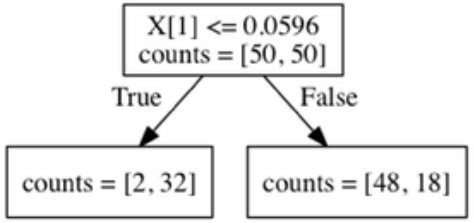


- 노드와 간선으로 이루어짐
  - 컴퓨터 자료구조 – 이진 트리
- 결정 트리는 노드를 질문,
- 간선은 노드 True or False 에 따라 다른 노드로 이동할 경로

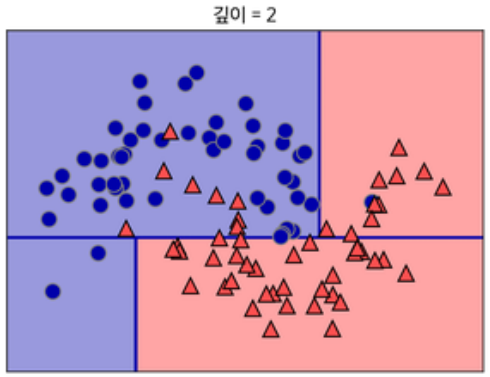
실제 결정 트리 구동예시

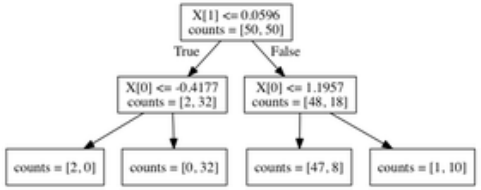
최대 깊이 : 1



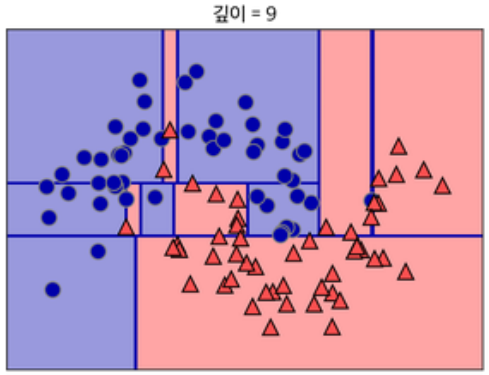


최대 깊이 : 2





최대 깊이 : 9



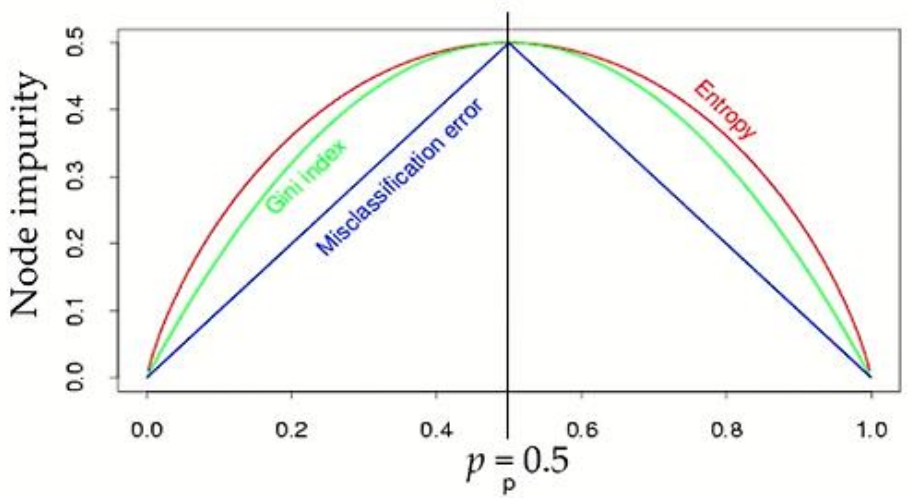
오차 나타내는 방법

- 분류 모델에서의 비용함수 (불순도 측정)

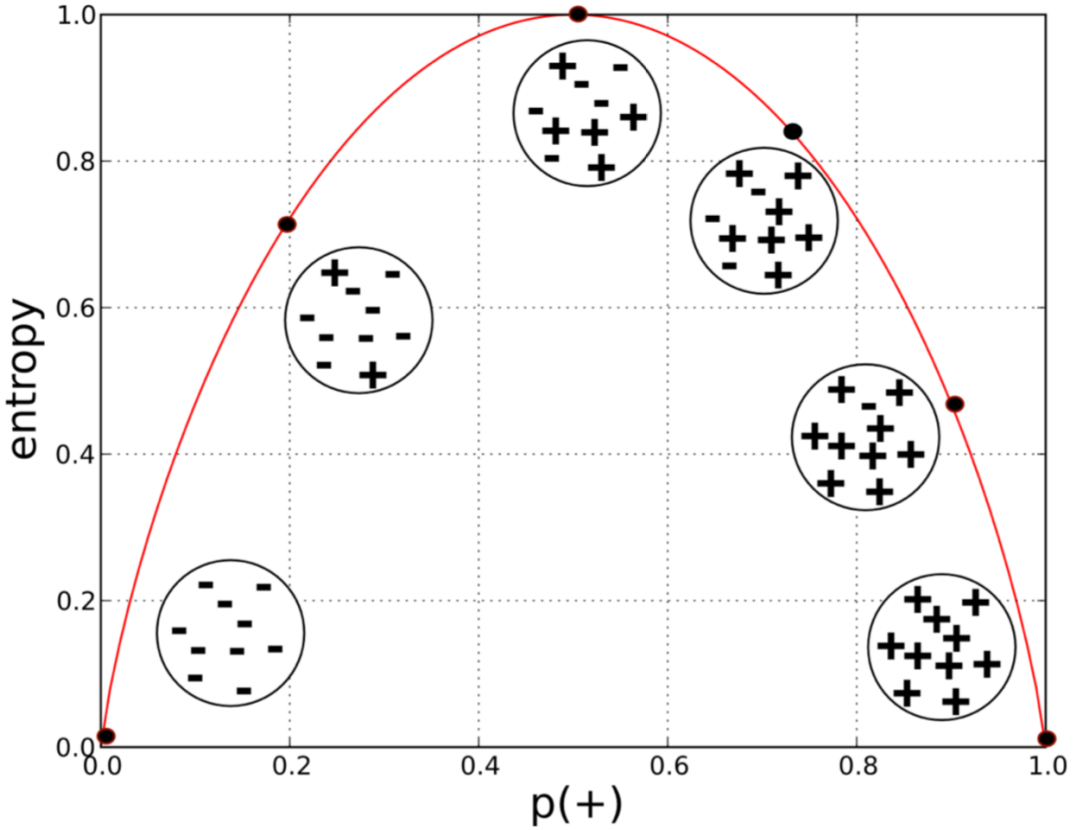
Misclassification rate: $\frac{1}{N_m}\sum_{i\in R_m} I\left(y_i \neq k(m)\right) = 1 - \hat{p}_{(mk)m}$

Gini Index: $\sum_{k\neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}\left(1-\hat{p}_{mk}\right)$

Cross-entropy : $-\sum_{k=1}^{K} \hat{p}_{mk}\, log\hat{p}_{mk}$



엔트로피 (불순물 비율)

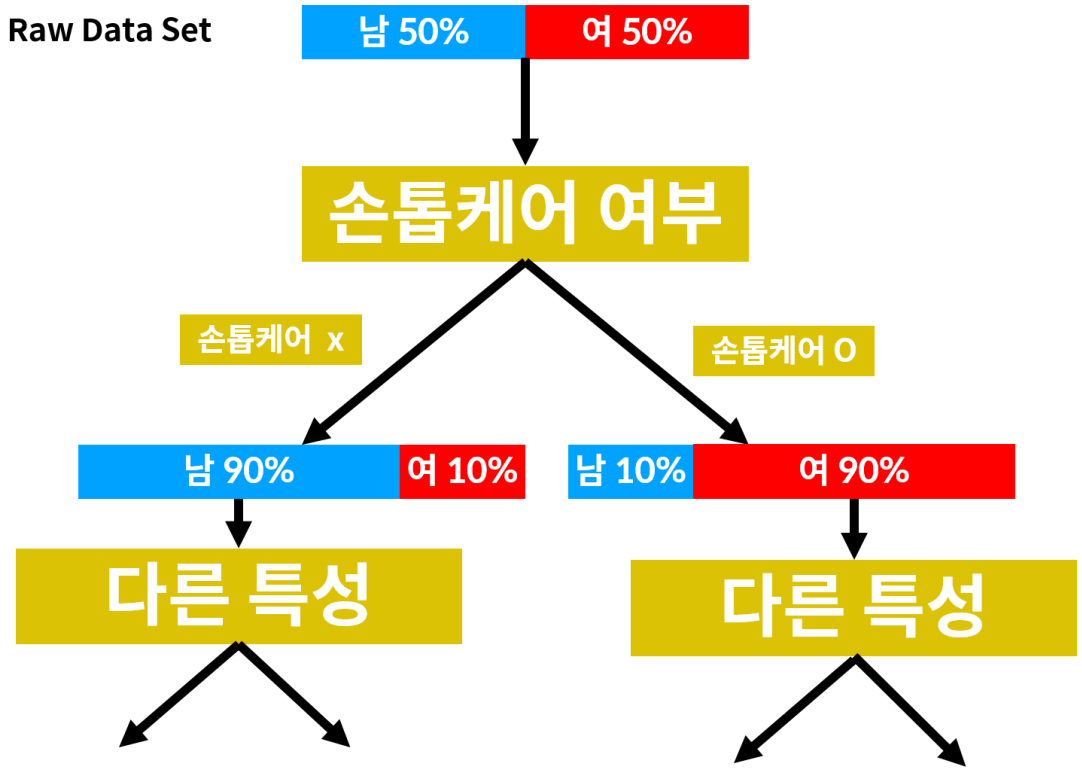


$$\text{Entropy} = -\sum_i (p_i) \log_2 (p_i)$$

- 얼마나 데이터가 섞여 있는지 (불순물 수치) 확인할 수 있다.
- 엔트로피가 작아야 해당 문제를 잘 푸는 것과 동일하다.

대표 x 값	엔트로피 y
0	0
0.5	1
1	0

데이터 예측하는 방안 개선(정보 획득)



$$IG(D, A) = I(D) - \sum \left(\frac{|D_j|}{|D|} \times I(D_j) \right)$$

여기서:

- $IG(D, A)$는 데이터 집합 D에 대한 속성 A의 정보 이득입니다.
- $I(D)$는 원래 데이터 집합의 불순도입니다.
- $\frac{|D_j|}{|D|}$는 분할된 데이터 집합의 크기 비율입니다.
- $I(D_j)$는 분할된 데이터 집합의 불순도입니다.

실습
와인 품질 분류 문제 해결해보기

Wine Quality Dataset | Kaggle

와인 품질 데이터 변수 요소

변수	설명	비고
fixed acidity	고정 산도	
volatile acidity	휘발성 산도	
citric acid	구연산	
residual sugar	잔여 설탕	
chlorides	염화물	
free sulfur dioxide	유리 이산화황(?)	
total sulfur dioxide	총 이산화황	
density	밀도	
pH	산성 수치	
sulphates	환산염	
alcohol	알코올	
quality	품질 수치	값의 범위는 0~10

데이터 불러오기

- import pandas as pd
- df = pd.read_csv('http://bit.ly/3JOzDZH')

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	0
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5	1
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5	2
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6	3
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6	1592
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	9.5	6	1593
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5	1594
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6	1595
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5	1597

1143 rows × 13 columns

데이터 상태 확인해보기

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

- df.info()

Quality 변수 범위 확인해보기

- df['quality'].unique()

- => array([5, 6, 7, 4, 8, 3], dtype=int64)

Quality를 다음처럼 바꾸자

데이터 범위	변환
3~4	0
5~6	1
7~8	2

```
for i,j in enumerate(range(3,8,2)):
    df.loc[df['quality'] == j,'quality'] = i
    df.loc[df['quality'] == j+1,'quality'] = i
```

변환 되었는지 확인

- df['quality'].unique()

- => array([1, 2, 0], dtype=int64)

X,Y 변수 지정

- Y = df[['quality']].to_numpy()
- X = df.drop(columns=['Id','quality']).to_numpy()

데이터 나누기

sklearn.model_selection.train_test_split

sklearn.model_selection.**train_test_split**(**arrays*, *test_size=None*, *train_size=None*, *random_state=None*, *shuffle=True*, *stratify=None*) [source]

Split arrays or matrices into random train and test subsets.

Quick utility that wraps input validation, `next(ShuffleSplit().split(X, y))`, and application to input data into a single call for splitting (and optionally subsampling) data into a one-liner.

Read more in the User Guide.

Parameters:

arrays : *sequence of indexables with same length / shape[0]
Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

test_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. If `train_size` is also None, it will be set to 0.25.

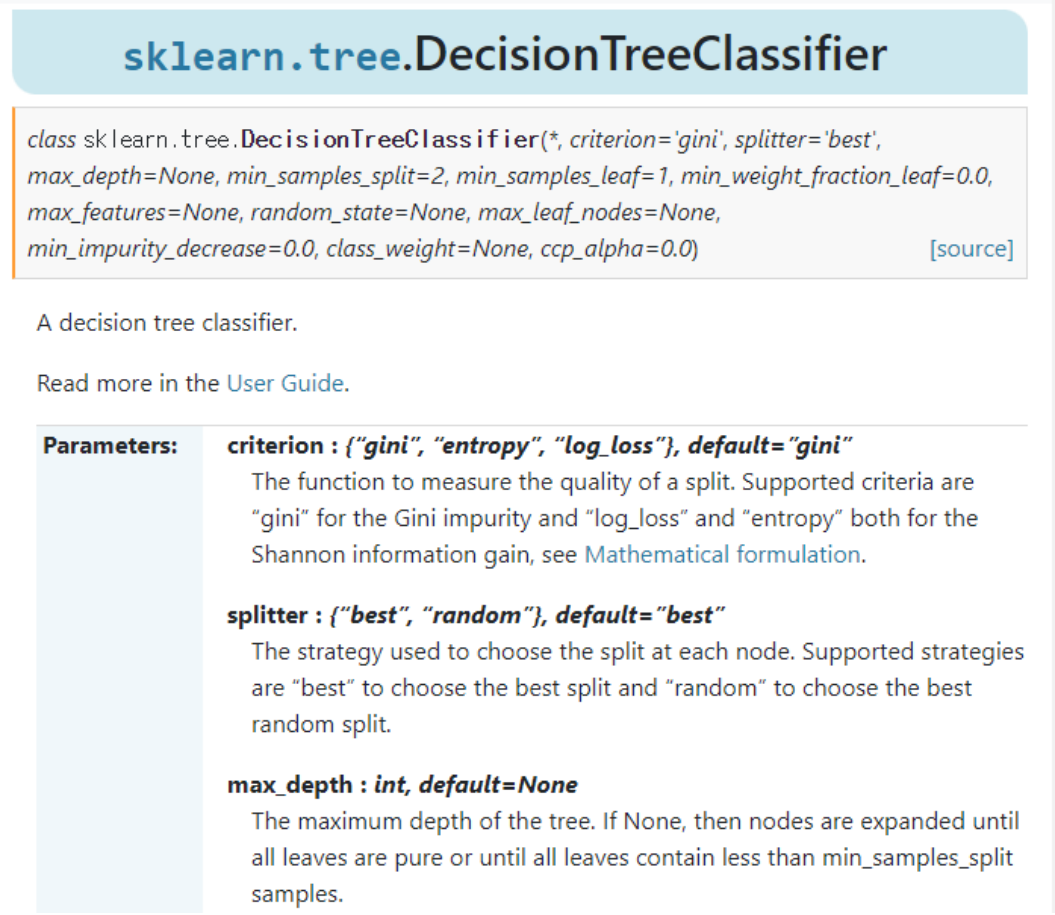
train_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

random_state : *int, RandomState instance or None, default=None*
Controls the shuffling applied to the data before applying the split. Pass an int for reproducible output across multiple function calls. See Glossary.

- from sklearn.model_selection import train_test_split

- X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,random_state=2023)

결정 트리 불러오기 & 생성



sklearn.tree.DecisionTreeClassifier

class sklearn.tree.**DecisionTreeClassifier**(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

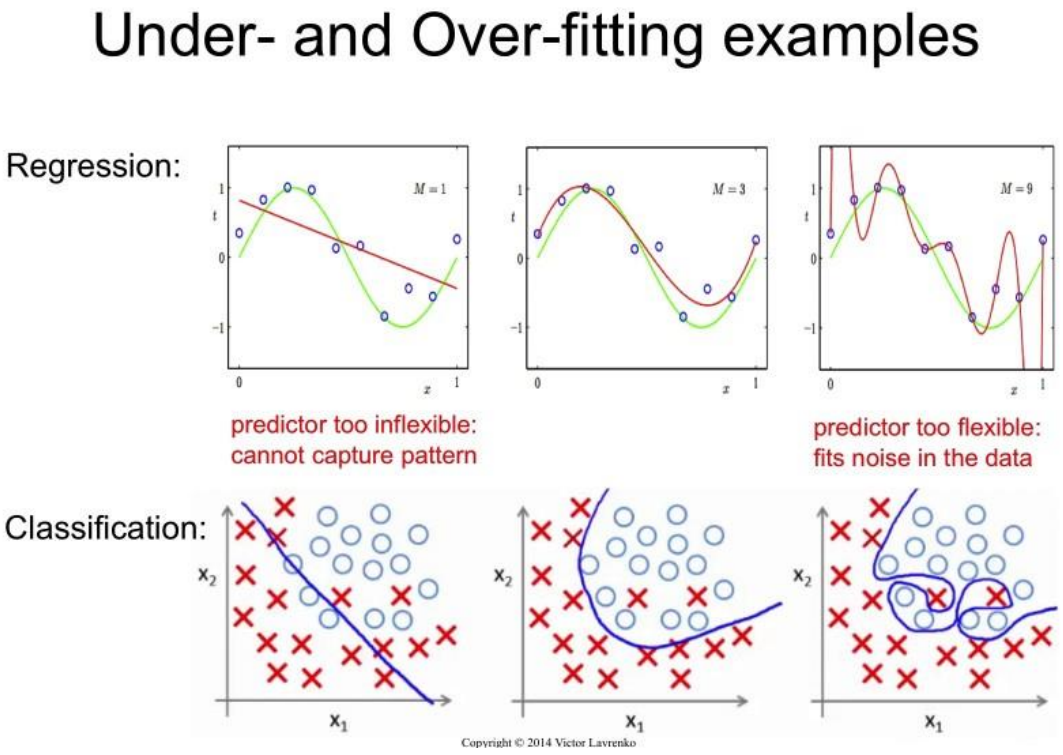
- from sklearn.tree import DecisionTreeClassifier
- model1 = DecisionTreeClassifier()

모델 학습 후 학습 & 평가 데이터 정확도 확인

- model1.fit(X_train,Y_train)
- model1.score(X_train,Y_train)
- model1.score(X_test,Y_test)

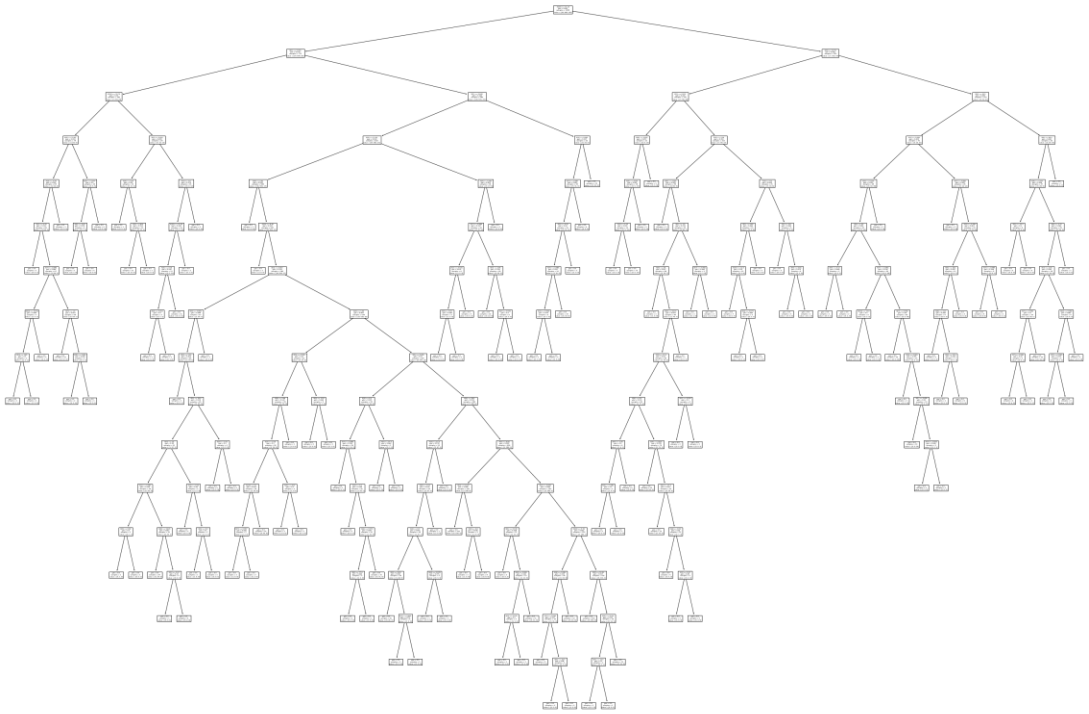
- 실행하면 정확도가 100% 나온다.
- Q. 결정 트리는 왜 실전 문제에서 약한 것일까?

오버 & 언더 피팅



- 오버 피팅
  - 학습 데이터 정확도 ↑
  - 테스트 데이터 < 학습 데이터 정확도
  - 학습을 너무 많이 한 것.
- 언더 피팅
  - 학습 데이터 정확도 ↓
  - 테스트 데이터 & 학습 데이터 정확도 ↓
  - 모델이 단순 or 학습을 덜 한 것.
- 목표 : 학습 & 실전 문제 **정확도 거의 동일**하게

결정 트리 상태 확인



저렇게 복잡한 것은, 많이 학습한 것과 동일.

- from sklearn.tree import plot_tree
- import matplotlib.pyplot as plt
- plt.figure(figsize=(50, 35))
- plot_tree(model1)

가지치기

- 학습을 제한 시키기
- 최대 깊이를 지정 및 줄이기
- Sklearn 에서는 max_depth 지정 가능

Parameters:

criterion : ***{"gini", "entropy", "log_loss"}, default="gini"***

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation.

splitter : ***{"best", "random"}, default="best"***

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : ***int, default=None***

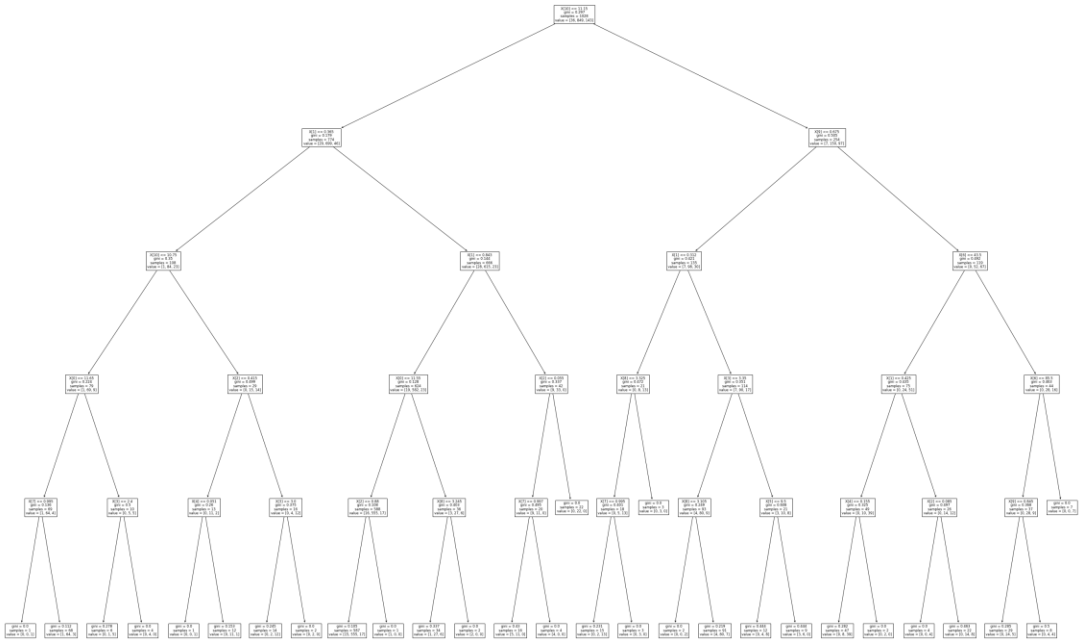
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

가지치기 실습

- model1 = DecisionTreeClassifier(max_depth=5)
- model1.fit(X_train,Y_train)
- model1.score(X_train,Y_train)
- model1.score(X_test,Y_test)

- 훈련 데이터의 정확도 90%
- 테스트 데이터의 정확도 87%
- 확실히 실전 정확도가 높아짐.

가지치기 상태 확인



- plt.figure(figsize=(50, 35))

  plot_tree(model1)

앙상블

- 여러 사람을 의미

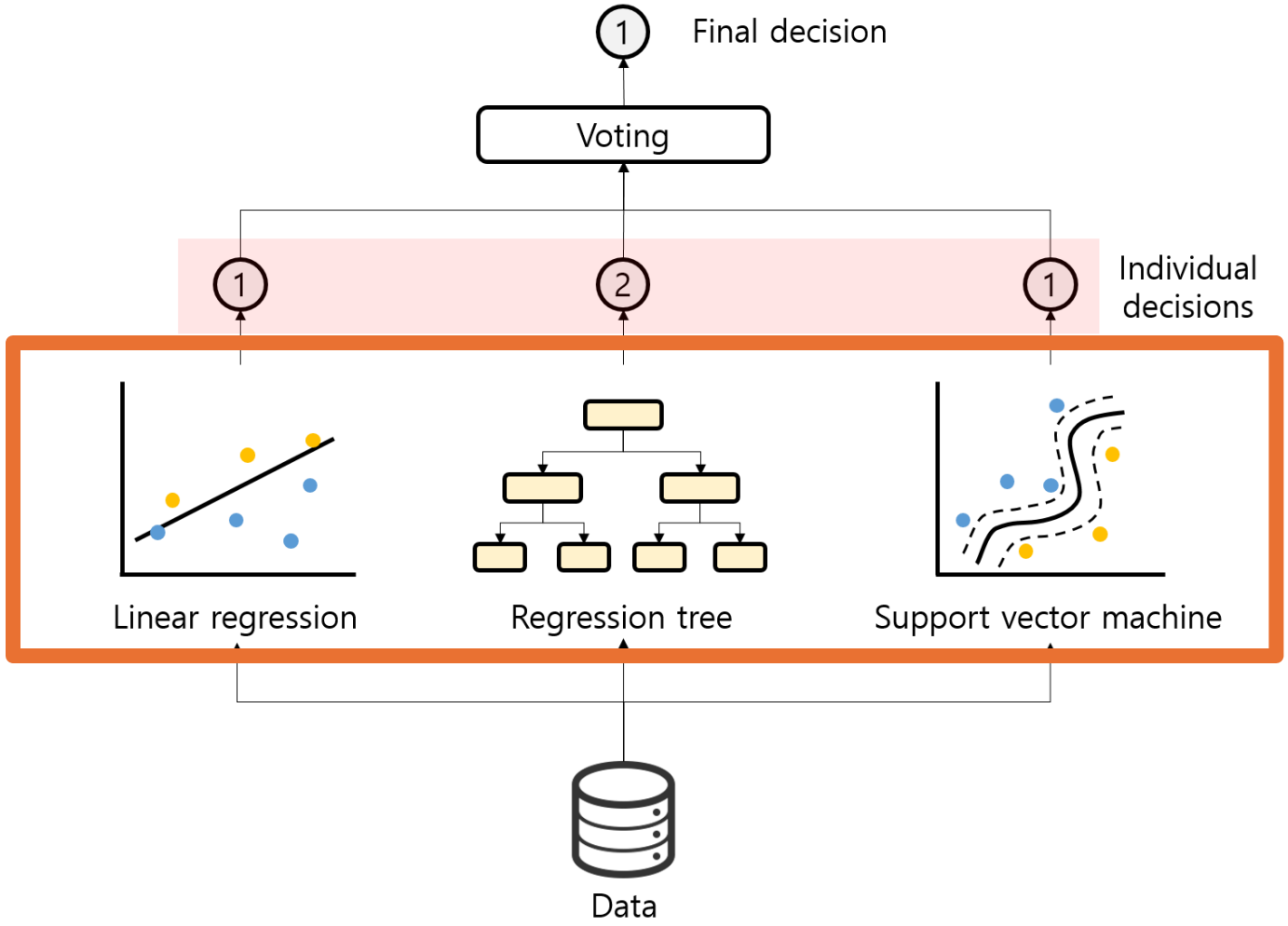
한 명의 전문가

VS

여러 명의 일반인

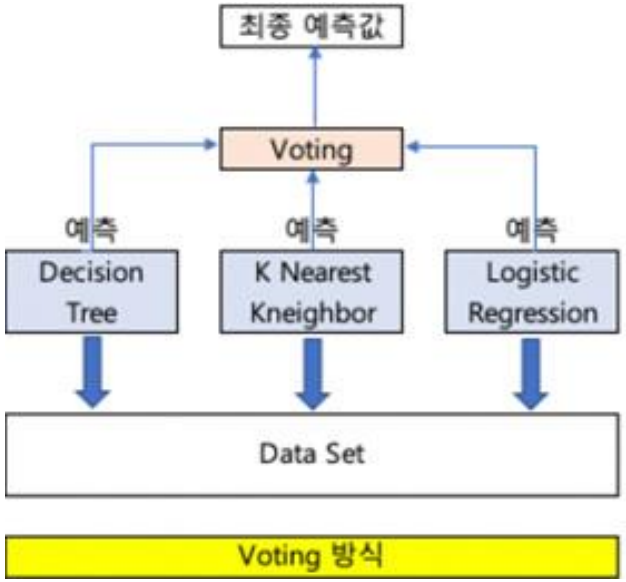
앙상블 모델의 구조

다양한 머신러닝 기법 사용

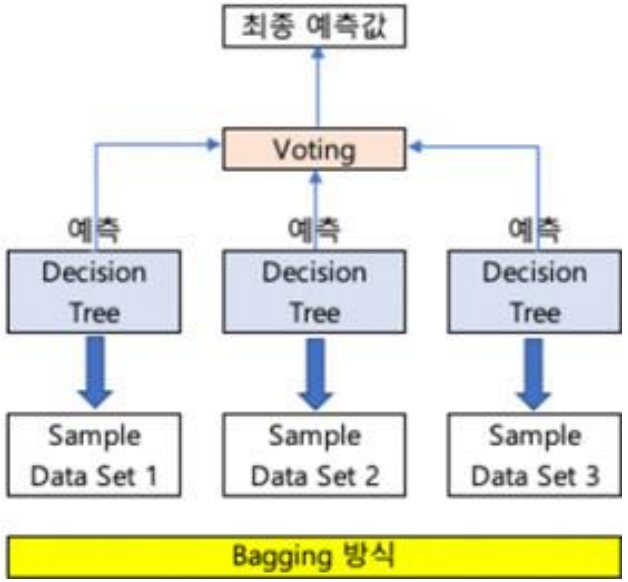


앙상블 종류

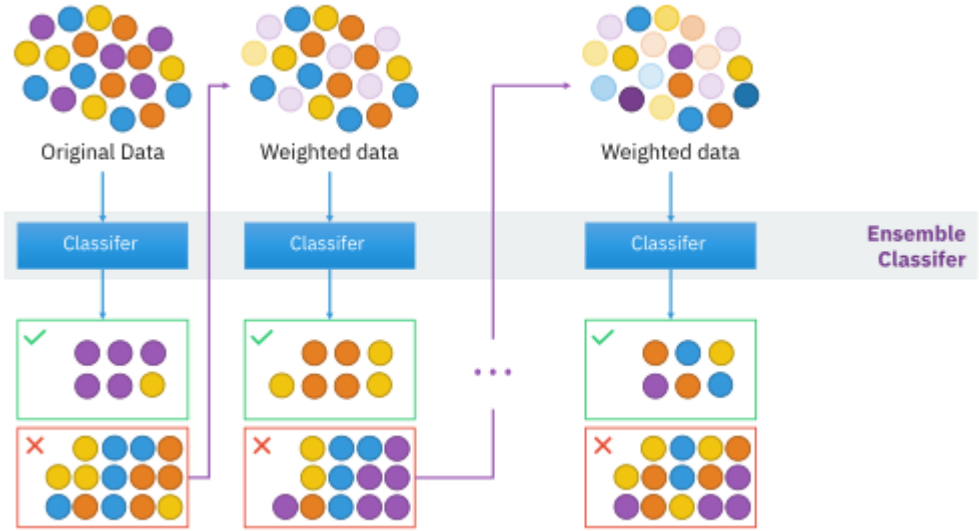
Voting



Bagging



Boosting



Ex) Voting 모델

Ex) 랜덤 포레스트

Ex) 그라디언트 부스팅

랜덤 포레스트 실습

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation. Note: This parameter is tree-specific.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

- from sklearn.ensemble import RandomForestClassifier
- model2 = RandomForestClassifier()

모델 훈련 & 정확도 확인

- model2.fit(X_train,Y_train)
- model2.score(X_train,Y_train)
- model2.score(X_test,Y_test)

- 훈련 데이터는 정확도 100%
- 테스트 데이터는 정확도 89%
  - 결정트리보다 성능 개선