Undergraduate Research Internship in Affective AI LAB.

# 혼자 공부하는 머신러닝 & 딥러닝

4주차 : ch05. 트리 알고리즘


컴퓨터공학과 12201682 공지윤

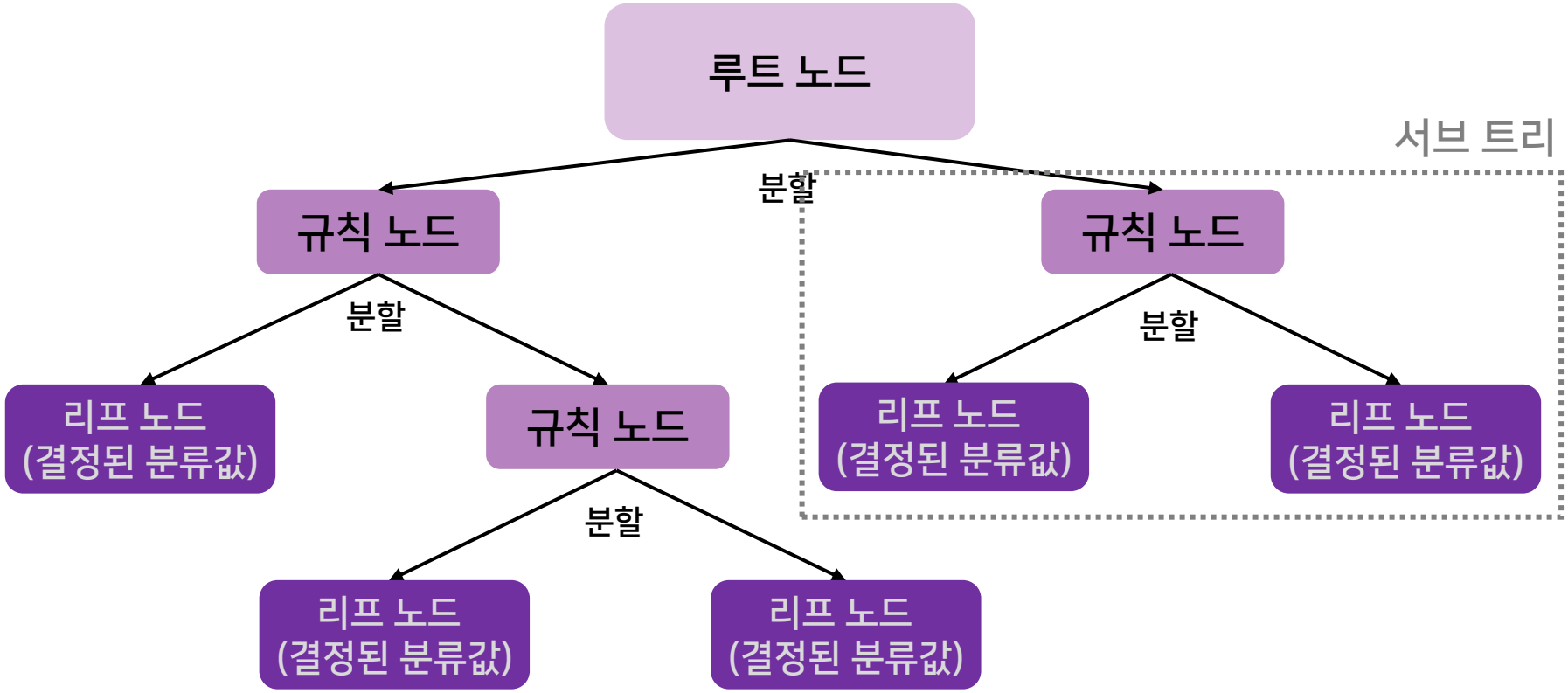
# 목차



# 1. Decision Tree

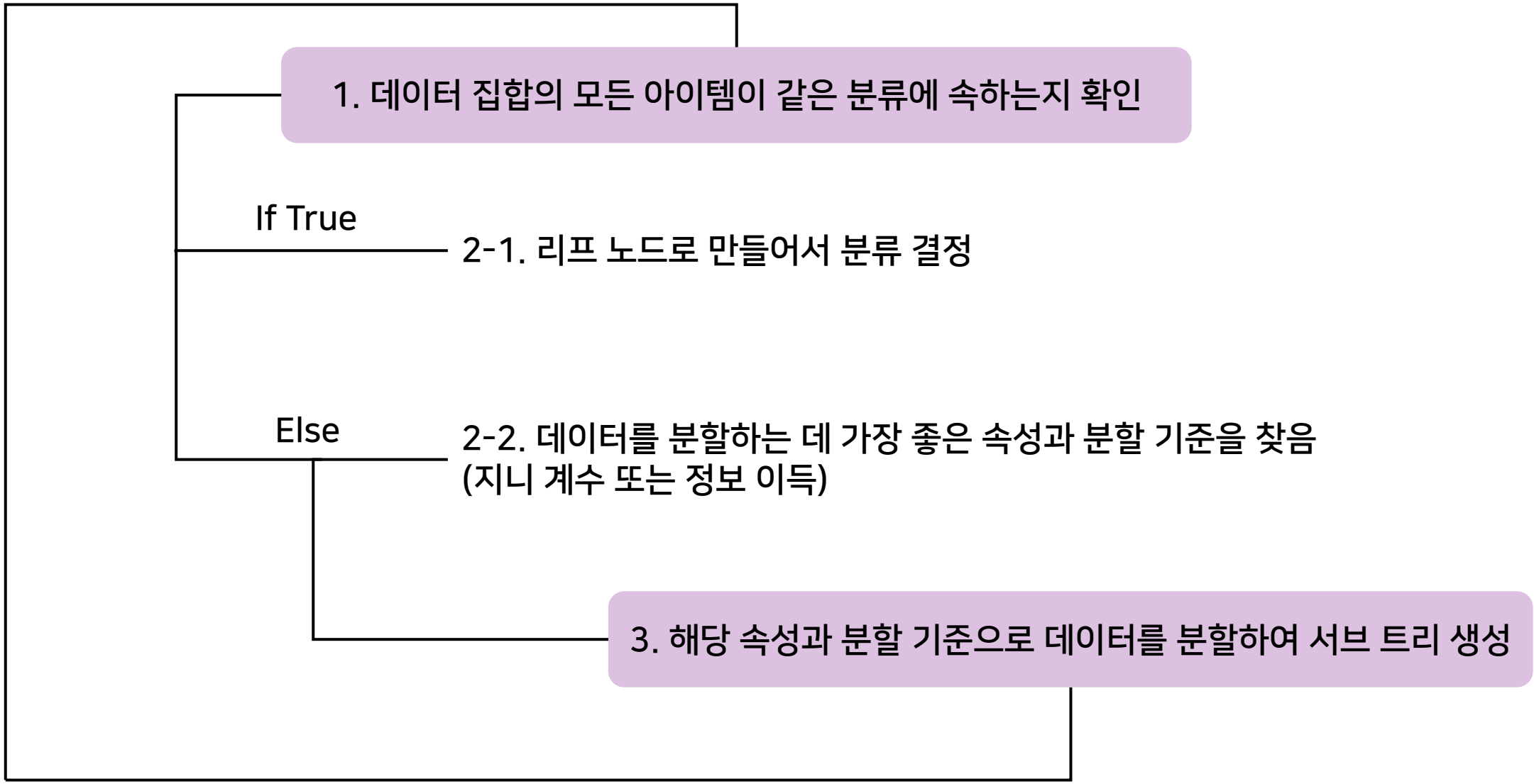
[ 데이터 균일도에 따른 규칙 기반 분류 알고리즘 ]

- 매우 쉽고 유연하게 적용될 수 있음
- 데이터의 스케일링이나 정규화 등의 사전 가공의 영향이 매우 적음
- 어떠한 피처가 가장 중요한 영향을 주는지 확인할 수 있으며, 피처 별로 어떤 척도에 따라 분류했는지 알 수 있음



# 1. Decision Tree

[ 규칙 노드 생성 프로세스 ]



# 1. Decision Tree

[ 데이터의 균일도 ]

A 집단

B 집단

< A 집단 >

데이터가 다양하다.
균일하지 못하다.
어떤 동물이 나올지 예측하기 어렵다.

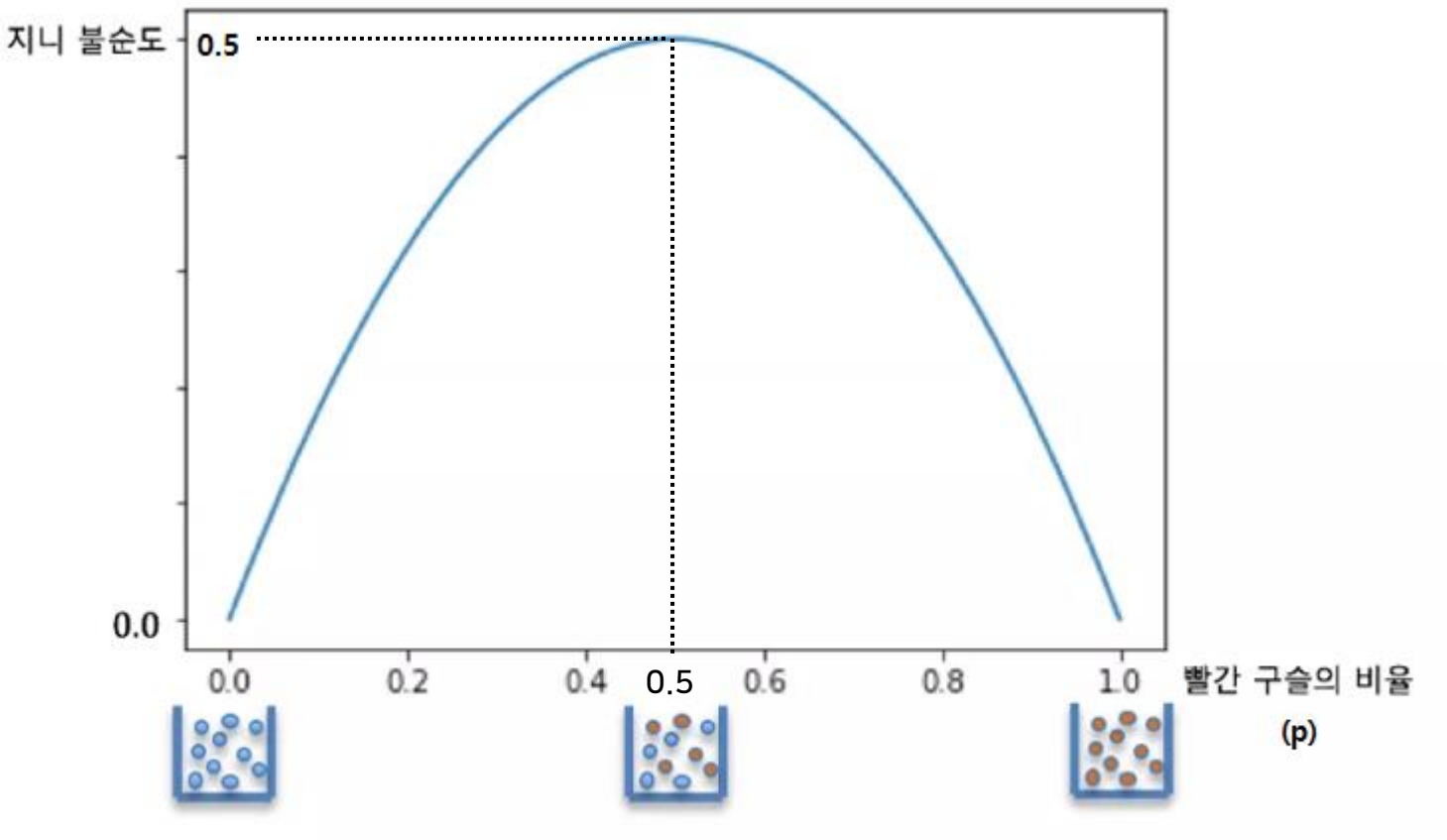
< B 집단 >

데이터가 다양하지 않다.
균일하다.
너구리로 예측할 확률이 높다.

# 1. Decision Tree

[ gini 지수 ]

불순도 측정 지수. 0.5에 가까울수록 불순도가 높으며, 0에 가까울수록 불순도가 낮다.




$$G(\text{집단}) = 1 - \sum_{k=0}^{n} {p_k}^2$$

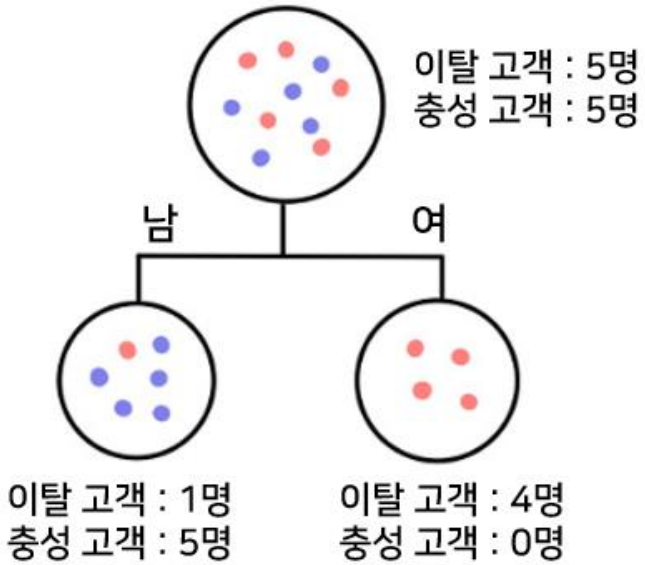
# 1. Decision Tree

[ gini 지수 ]

예) 충성 고객과 이탈 고객 구분하는 규칙 만들기

- 총 고객 : 10명 / - 후보 조건 : 성별, 결혼 여부

## (1) 성별 기준



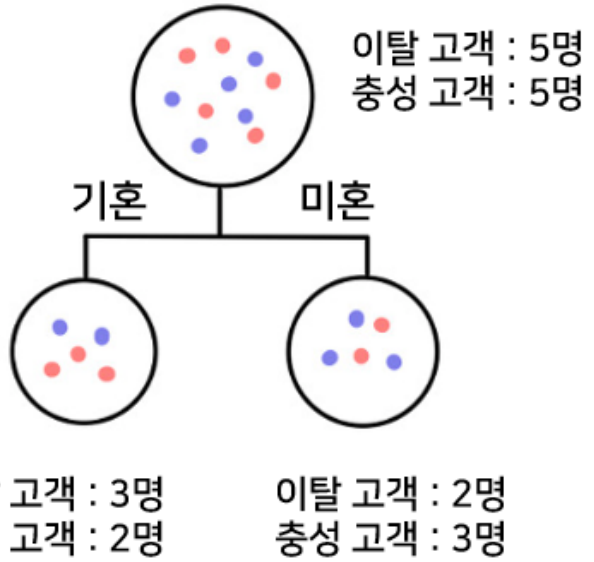

$$G(\text{상위}) = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2 = 0.5$$

$$G(\text{남}) = 1 - \left(\frac{5}{6}\right)^2 - \left(\frac{1}{6}\right)^2 = 0.278$$

$$G(\text{여}) = 1 - \left(\frac{0}{4}\right)^2 - \left(\frac{4}{4}\right)^2 = 0$$

$$G(\text{성별}) = \left(\frac{6}{10}\right) \text{X } (0.278) + \left(\frac{4}{10}\right) \text{X } (0) = 0.167$$

## (2) 결혼 여부 기준




$$G(\text{상위}) = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2 = 0.5$$

$$G(\text{기혼}) = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$$

$$G(\text{미혼}) = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 0.48$$

$$G(\text{결혼}) = \left(\frac{5}{10}\right) \text{X } (0.48) + \left(\frac{5}{10}\right) \text{X } (0.48) = 0.48$$

# 1. Decision Tree

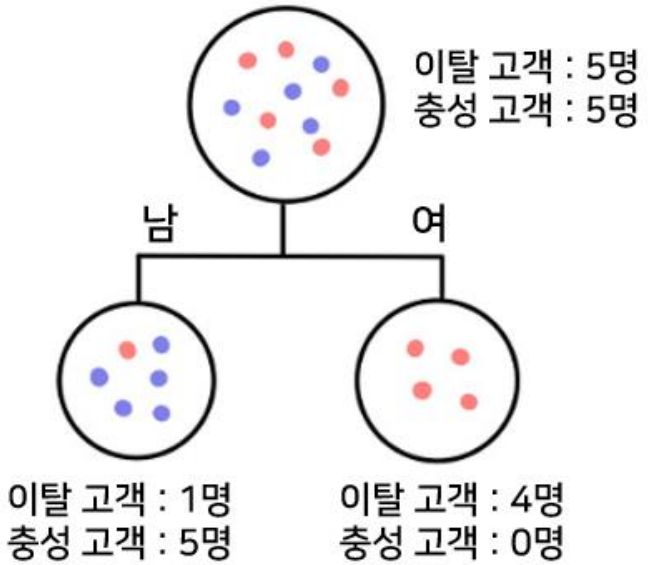
[ gini 지수 ]

< 정보 이득 >

(1) 성별 기준
$0.5 - 0.167 = 0.333$

(2) 결혼 여부 기준
$0.5 - 0.48 = 0.02$

## (1) 성별 기준



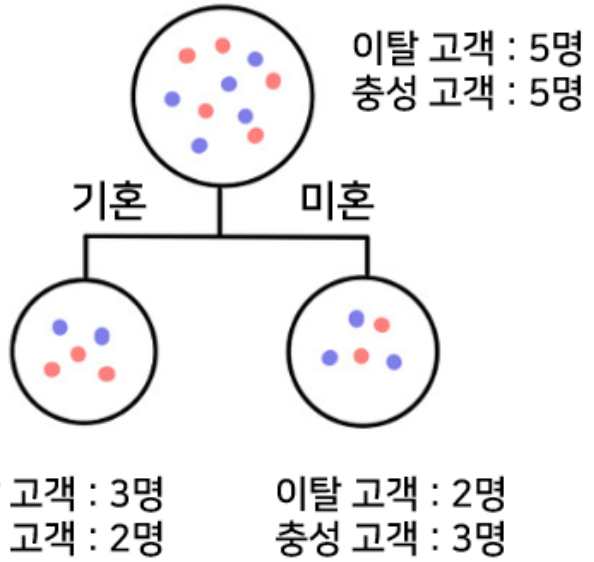

$G(\text{상위}) = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2 = 0.5$

$G(\text{남}) = 1 - \left(\frac{5}{6}\right)^2 - \left(\frac{1}{6}\right)^2 = 0.278$

$G(\text{여}) = 1 - \left(\frac{0}{4}\right)^2 - \left(\frac{4}{4}\right)^2 = 0$

$G(\text{성별}) = \left(\frac{6}{10}\right) \text{X } (0.278) + \left(\frac{4}{10}\right) \text{X } (0) = 0.167$

## (2) 결혼 여부 기준




$G(\text{상위}) = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2 = 0.5$

$G(\text{기혼}) = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$

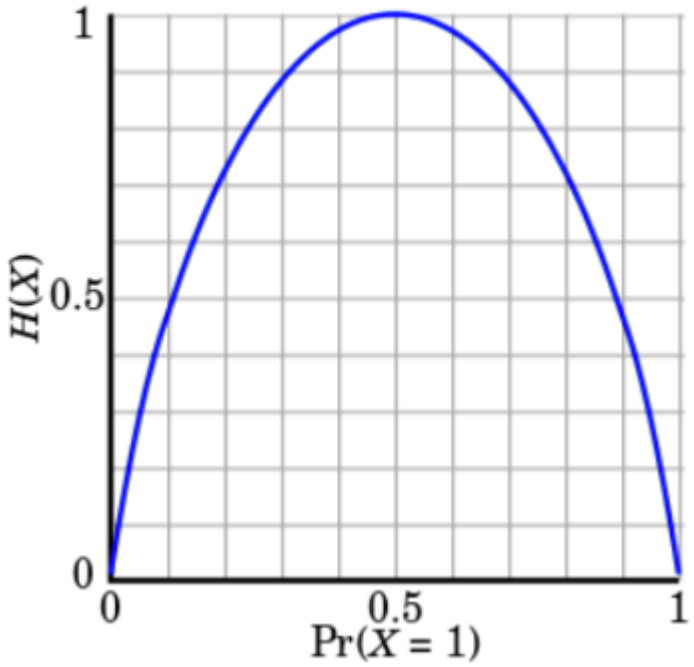
$G(\text{미혼}) = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 0.48$

$G(\text{결혼}) = \left(\frac{5}{10}\right) \text{X } (0.48) + \left(\frac{5}{10}\right) \text{X } (0.48) = 0.48$

# 1. Decision Tree

[ entropy 지수 ]

노드의 무작위성 지수. 1에 가까울수록 무작위성이 높고, 0에 가까울수록 무작위성이 낮다.




$$Entropy = -\sum_{k=0}^{n} p_k \log_2(p_k)$$

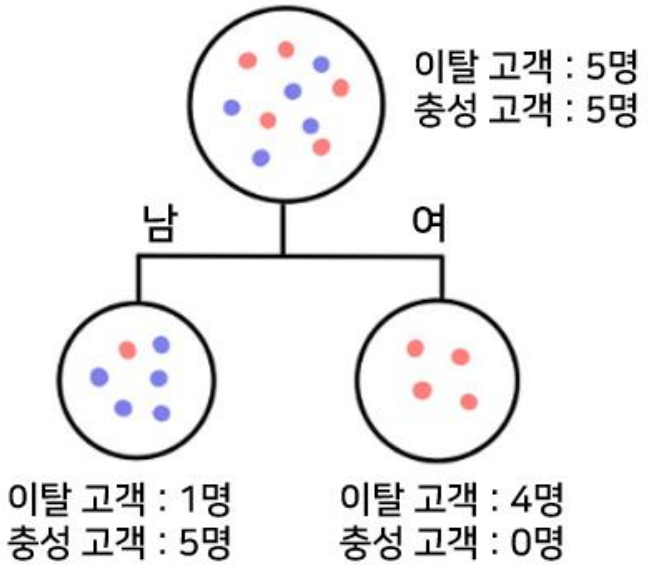
# 1. Decision Tree

[ entropy 지수 ]

예) 충성 고객과 이탈 고객 구분하는 규칙 만들기

- 총 고객 : 10명 / - 후보 조건 : 성별, 결혼 여부

## (1) 성별 기준



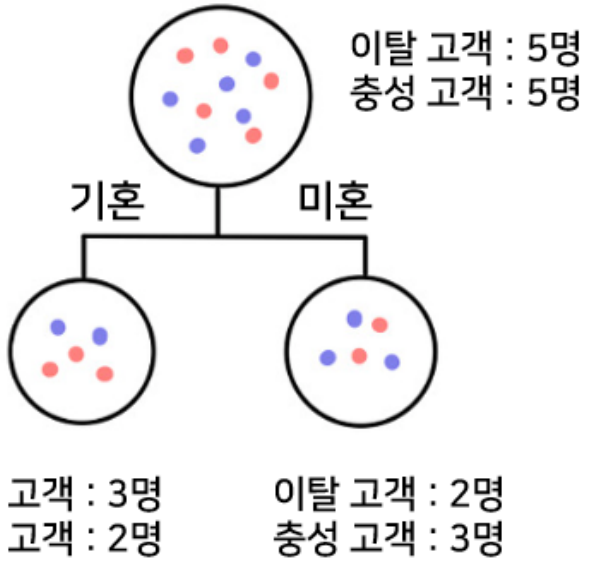

$$E(\text{상위}) = -\left(\frac{5}{10}\right)\log_2\left(\frac{5}{10}\right) - \left(\frac{5}{10}\right)\log_2\left(\frac{5}{10}\right) = 1$$

$$E(\text{남}) = -\left(\frac{5}{6}\right)\log_2\left(\frac{5}{6}\right) - \left(\frac{1}{6}\right)\log_2\left(\frac{1}{6}\right) = 0.65$$

$$E(\text{여}) = -\left(\frac{0}{4}\right)\log_2\left(\frac{0}{4}\right) - \left(\frac{4}{4}\right)\log_2\left(\frac{4}{4}\right) = 0$$

$$E(\text{성별}) = \left(\frac{6}{10}\right) X\ (0.65) + \left(\frac{4}{10}\right) X\ (0) = 0.39$$

## (2) 결혼 여부 기준




$$E(\text{상위}) = -\left(\frac{5}{10}\right)\log_2\left(\frac{5}{10}\right) - \left(\frac{5}{10}\right)\log_2\left(\frac{5}{10}\right) = 1$$

$$E(\text{기혼}) = -\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) = 0.971$$

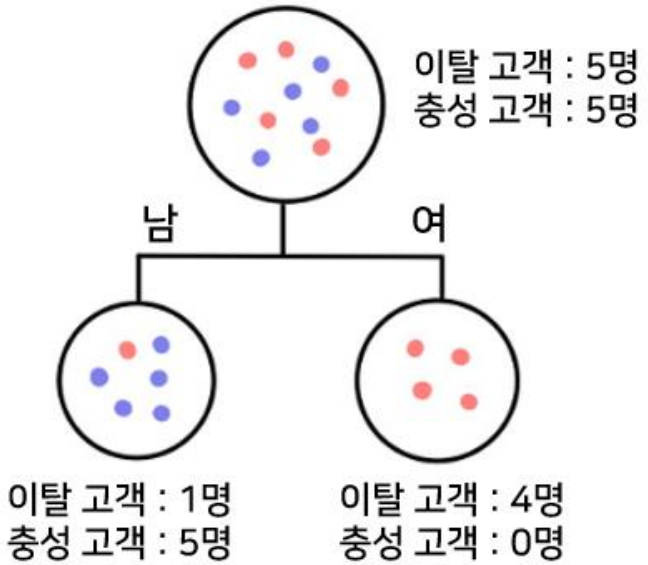
$$E(\text{미혼}) = -\left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) = 0.971$$

$$E(\text{결혼}) = \left(\frac{5}{10}\right) X\ (0.971) + \left(\frac{5}{10}\right) X(0.971) = 0.971$$

# 1. Decision Tree

[ entropy 지수 ]

### (1) 성별 기준




$$E(\text{상위}) = -\left(\frac{5}{10}\right)\log_2\left(\frac{5}{10}\right) - \left(\frac{5}{10}\right)\log_2\left(\frac{5}{10}\right) = 1$$

$$E(\text{남}) = -\left(\frac{5}{6}\right)\log_2\left(\frac{5}{6}\right) - \left(\frac{1}{6}\right)\log_2\left(\frac{1}{6}\right) = 0.65$$

$$E(\text{여}) = -\left(\frac{0}{4}\right)\log_2\left(\frac{0}{4}\right) - \left(\frac{4}{4}\right)\log_2\left(\frac{4}{4}\right) = 0$$

$$E(\text{성별}) = \left(\frac{6}{10}\right) X\ (0.65) + \left(\frac{4}{10}\right) X\ (0) = 0.39$$

**< 정보 이득 >**

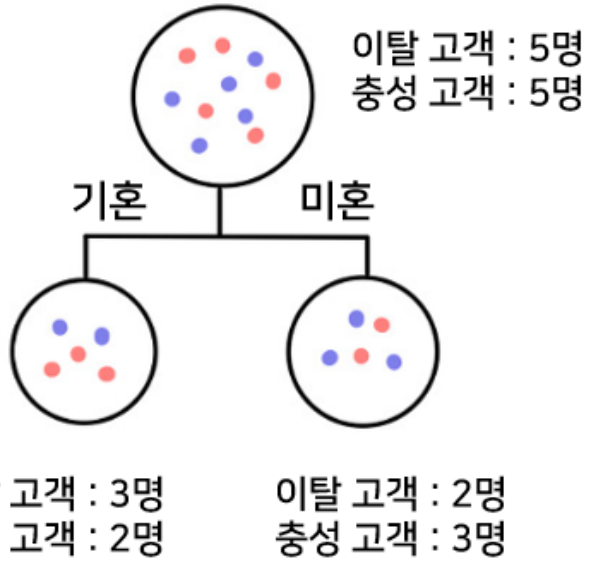
(1) 성별 기준

$1 - 0.39 = 0.61$

(2) 결혼 여부 기준

$1 - 0.971 = 0.029$

### (2) 결혼 여부 기준




$$E(\text{상위}) = -\left(\frac{5}{10}\right)\log_2\left(\frac{5}{10}\right) - \left(\frac{5}{10}\right)\log_2\left(\frac{5}{10}\right) = 1$$

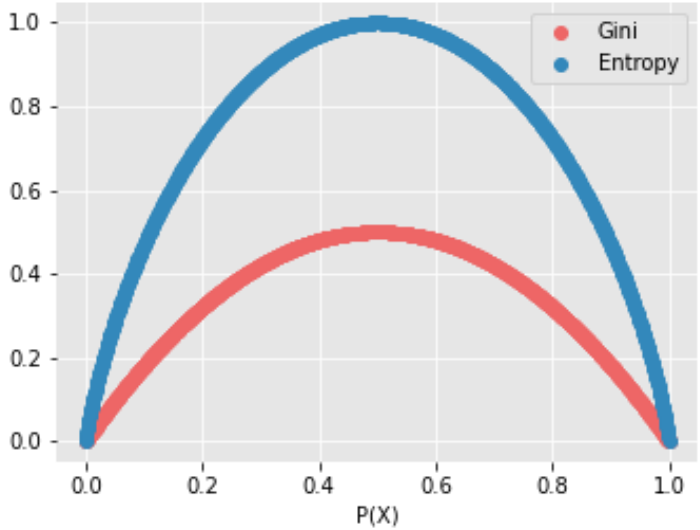
$$E(\text{기혼}) = -\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) = 0.971$$

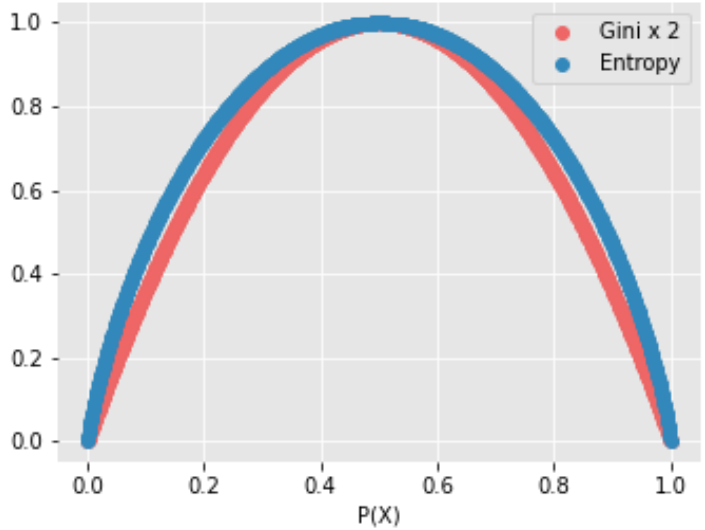
$$E(\text{미혼}) = -\left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) = 0.971$$

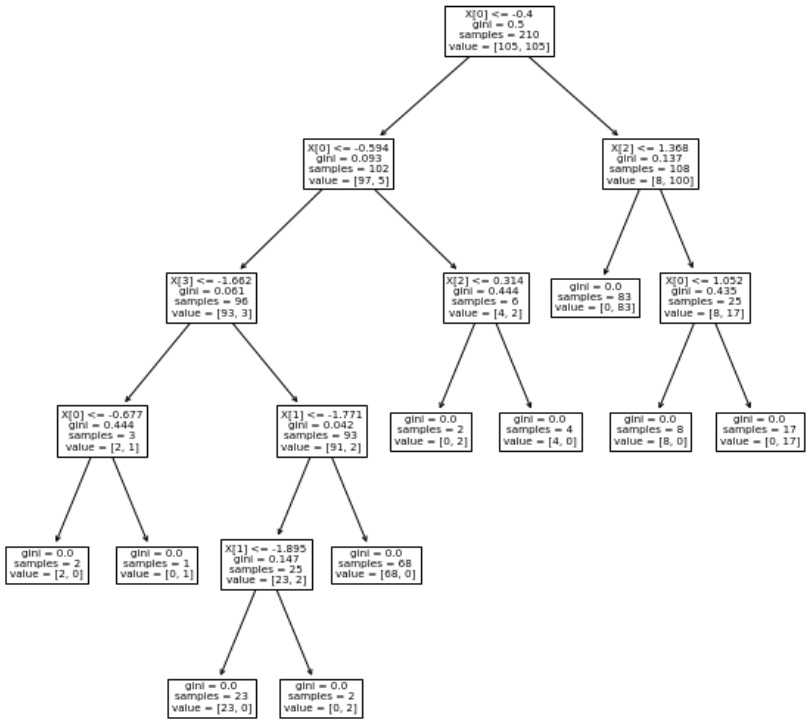
$$E(\text{결혼}) = \left(\frac{5}{10}\right) X\ (0.971) + \left(\frac{5}{10}\right) X(0.971) = 0.971$$

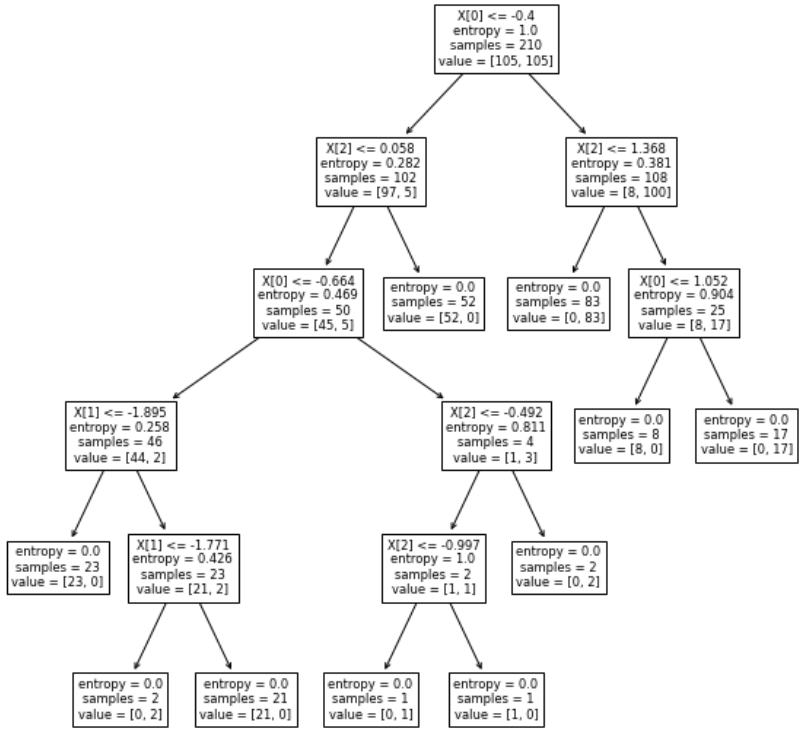
# 1. Decision Tree

[ gini vs. entropy ]



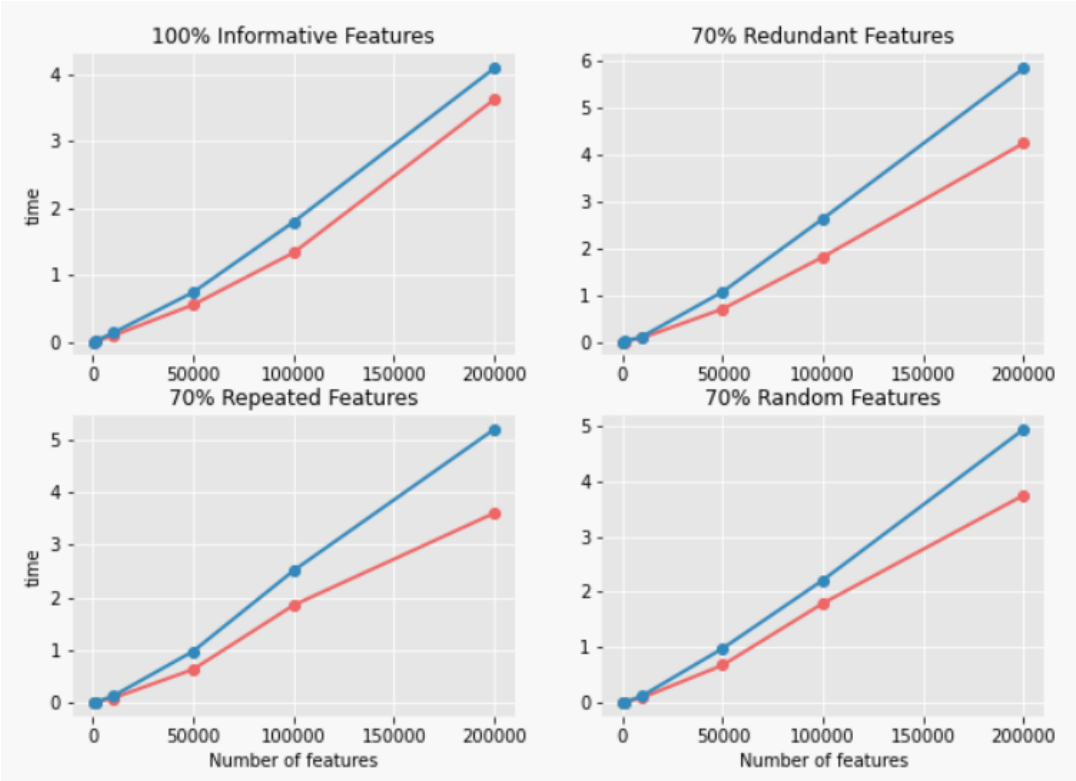




크기가 400이고 4개의 피처를 가진 데이터

: 모양 다름, 결과 비슷

# 1. Decision Tree

[ gini vs. entropy ]




- gini 계산량 < entropy 계산량
- 크기가 큰 데이터셋에는 gini를 사용하는 편이 낫다.

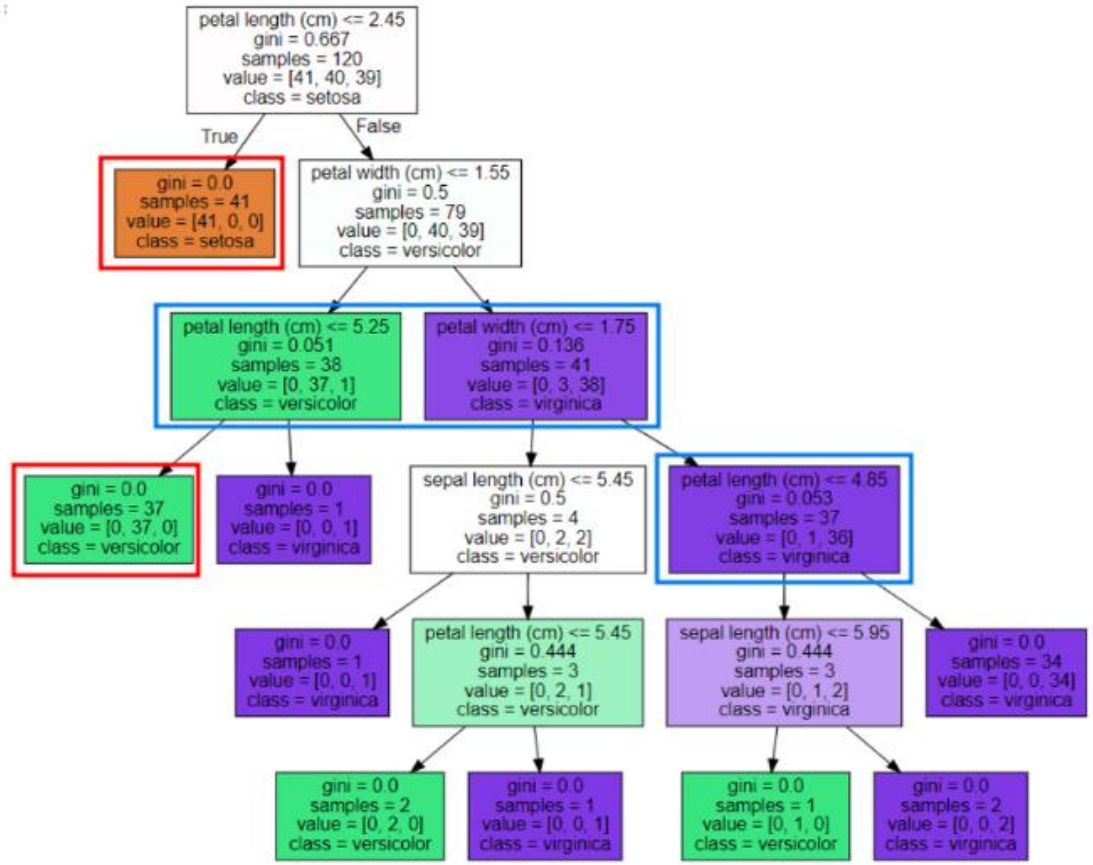
출처 : https://quantdare.com/decision-trees-gini-vs-entropy/

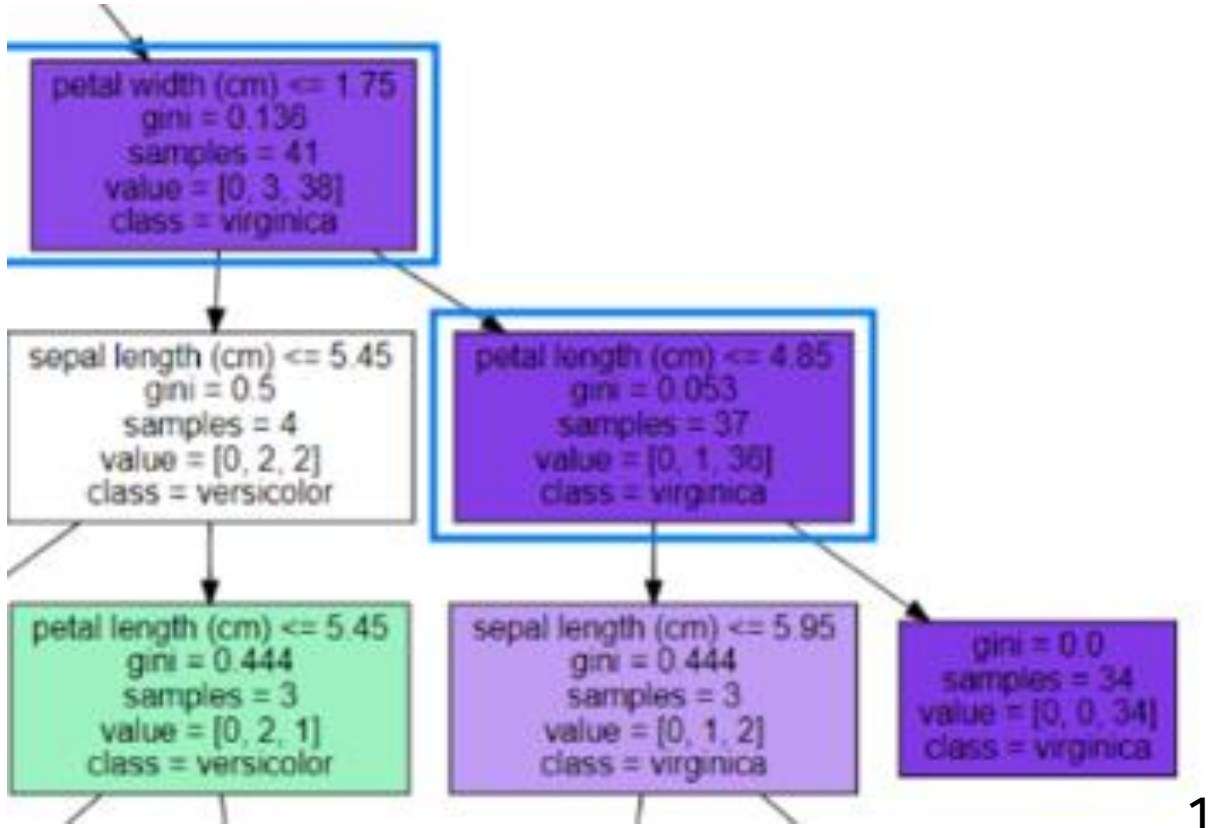
# 1. Decision Tree

[ Decision Tree의 단점 ]

- 과적합으로 알고리즘 성능이 떨어짐.
- 단일 분류 값으로 만들기 위해서 계속해서 규칙을 생성.
- 학습 데이터 셋에서는 좋은 성능을 보일 수 있지만, 테스트 데이터 셋은 오히려 성능이 떨어짐.

Tree의 크기를 사전에 제한하는 튜닝이 필요!



# 1. Decision Tree

[ Decision Tree Parameter ]

(1) max_depth : 트리의 최대 깊이를 규정
- 기본값 : None
- 노드가 가지는 데이터 개수가 min_samples_split보다 작아질 때까지 트리의 깊이 계속 증가

(2) max_features : 최적의 분할을 위해 고려할 최대 피처 개수
- 기본값 : None
- Int, float, sqrt, log 등

(3) min_samples_split : 노드를 분할하기 위한 최소한의 sample 데이터 수
- 기본값 : 2

(4) min_samples_leaf : 분할이 될 경우 리프 노드들이 가져야 할 최소한의 sample 데이터 수
- 기본값 : 1

(5) max_leaf_nodes : 리프 노드의 최대 개수
- 기본값 : None

# 2. Cross Validation

[ Cross Validation의 필요성 ]


고정된 test 셋을 통해 모델의 성능 검증, 수정 반복?

↓

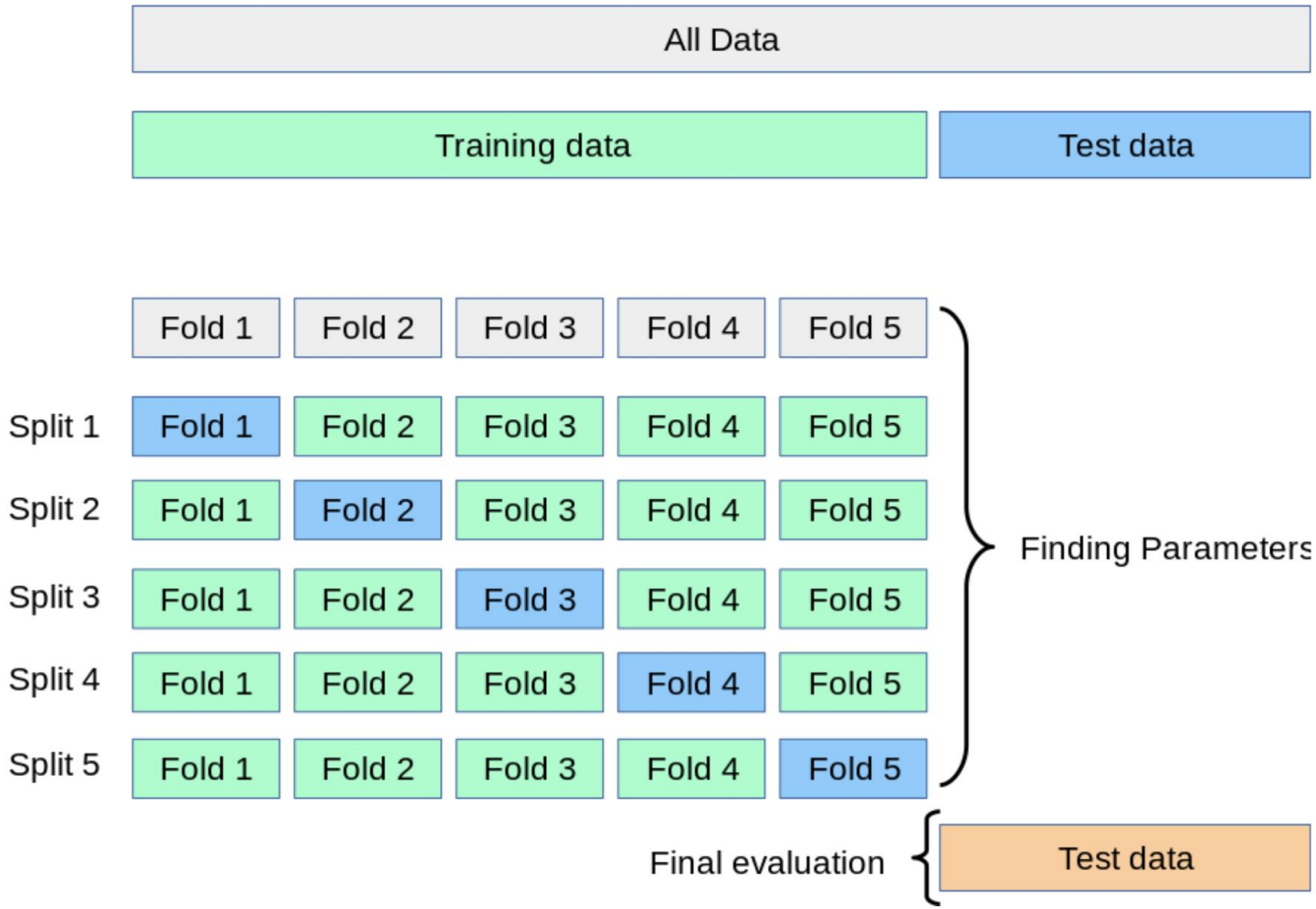
모델이 test 데이터에 과적합!


모의고사 수능

# 2. Cross Validation

[ K-fold CV ]

일반적인 K-Fold CV
K = 5



# 2. Cross Validation

[ Stratified K-fold CV ]

- 불균형한 (imbalanced) 분포도를 가진 라벨 데이터 집합을 위한 K-fold 방식

예) 신용 카드 사기 거래 : 2만 건의 거래 중 100건의 사기 거래 존재 (0.5%)

- 학습 데이터와 검증 데이터 세트가 가지는 레이블 분포도가 유사하도록 검증 데이터 추출
- 분류는 대부분 Stratified K-fold CV 사용
- 회귀에서는 Stratified K-fold CV 지원되지 않음

# 2. Cross Validation

[ Stratified K-fold CV ]

K-fold CV의 단점을 보여주는 예시 : iris 데이터 (총 3개의 label)

```
3  irisDF['target'].value_counts()
```

```
0    50
1    50
2    50
Name: target, dtype: int64
```

## (1) K-fold CV

```
## 교차 검증 : 1 ##
Train 레이블 데이터 분포 :
1    50
2    50
Name: label, dtype: int64

Valid 레이블 데이터 분포 :
0    50
Name: label, dtype: int64


## 교차 검증 : 2 ##
Train 레이블 데이터 분포 :
0    50
2    50
Name: label, dtype: int64

Valid 레이블 데이터 분포 :
1    50
Name: label, dtype: int64
```

## (2) Kfold (shuffle = True)

```
## 교차 검증 : 1 ##
Train 레이블 데이터 분포 :
0    39
2    32
1    29
Name: label, dtype: int64

Valid 레이블 데이터 분포 :
1    21
2    18
0    11
Name: label, dtype: int64


## 교차 검증 : 2 ##
Train 레이블 데이터 분포 :
1    35
2    35
0    30
Name: label, dtype: int64

Valid 레이블 데이터 분포 :
0    20
1    15
2    15
Name: label, dtype: int64
```

## (3) Stratified K-fold CV

```
## 교차 검증 : 1 ##
Train 레이블 데이터 분포 :
2    34
0    33
1    33
Name: label, dtype: int64

Valid 레이블 데이터 분포 :
0    17
1    17
2    16
Name: label, dtype: int64


## 교차 검증 : 2 ##
Train 레이블 데이터 분포 :
1    34
0    33
2    33
Name: label, dtype: int64

Valid 레이블 데이터 분포 :
0    17
2    17
1    16
Name: label, dtype: int64
```

# 2. Cross Validation

[ 참고 : 시계열에서의 cross validation ]

Independent and Identically Distribution (독립항등분포)

### 3.1.2.1. Cross-validation iterators for i.i.d. data

Assuming that some data is Independent and Identically Distributed (i.i.d.) is making the assumption that all samples stem from the same generative process and that the generative process is assumed to have no memory of past generated samples.

The following cross-validators can be used in such cases.

**Note:** While i.i.d. data is a common assumption in machine learning theory, it rarely holds in practice. If one knows that the samples have been generated using a time-dependent process, it is safer to use a time-series aware cross-validation scheme. Similarly, if we know that the generative process has a group structure (samples collected from different subjects, experiments, measurement devices), it is safer to use group-wise cross-validation.

# 2. Cross Validation

[ 참고 : 시계열에서의 cross validation ]

Independent and Identically Distribution (독립항등분포)

### 3.1.2.1. Cross-validation iterators for i.i.d. data

Assuming that some data is Independent and Identically Distributed (i.i.d.) is making the assumption that all samples stem from the same generative process and that the generative process is assumed to have no memory of past generated samples.

The following cross-validators can be used in such cases.

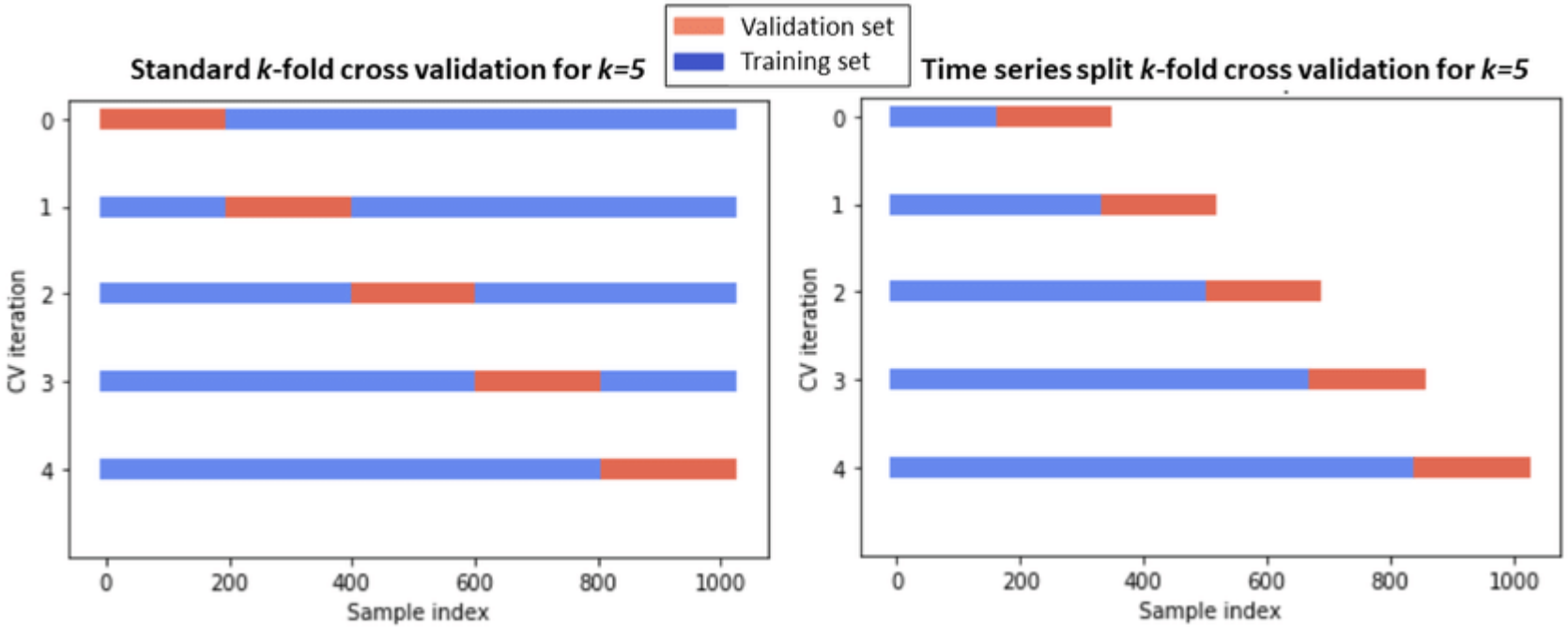
데이터는 기계 학습 이론에서 일반적인 가정이지만 실제로는 거의 유지되지 않습니다.
샘플이 시간 의존적인 프로세스를 사용하여 생성되었음을 알고 있는 경우 시계열 인식 교차 검증 체계를 사용하는 것이 더 안전합니다.
마찬가지로, 생성 과정이 그룹 구조(다른 주제, 실험, 측정 장치에서 수집된 샘플)를 가지고 있다는 것을 알고 있다면, 그룹별 교차 검증을 사용하는 것이 더 안전합니다.

# 2. Cross Validation

[ 참고 : 시계열에서의 cross validation ]

## 1. Walk forward Validation

- 시계열 데이터를 일련의 연속된 구간으로 분할하여 교차 검증을 수행하는 방법.
- 기본적으로 데이터는 순서대로 분할되며, 각 분할에서는 과거 데이터를 훈련 세트로 사용하고 미래 데이터를 테스트 세트로 사용
- Sklearn의 TimeSeriesSplit으로 구현 가능

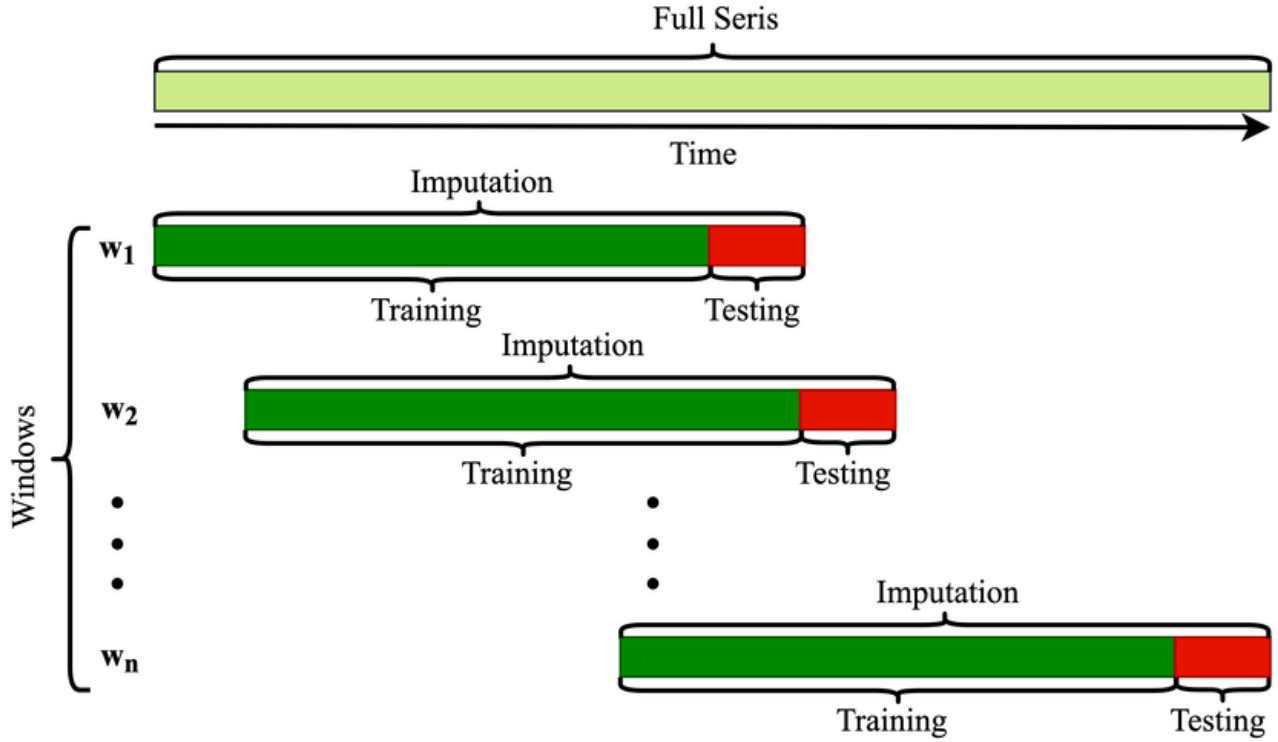


# 2. Cross Validation

[ 참고 : 시계열에서의 cross validation ]

## 2. 롤링 윈도우(Rolling Window)

- 훈련과 테스트 데이터의 크기를 고정한 상태로 윈도우를 이동시켜 교차 검증을 수행하는 방법
- 예) 윈도우 크기를 12개월로 설정하고 한 달 씩 윈도우를 이동시키면서 훈련과 테스트를 반복한다.

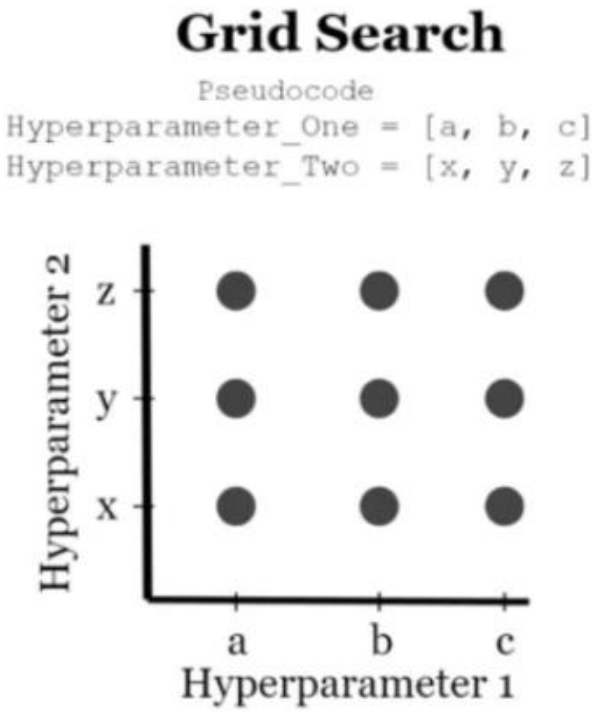


# 2. Cross Validation

[ 하이퍼파라미터 튜닝 ]

## 1. Grid Search CV

- 가능한 모든 조합의 하이퍼파라미터로 훈련시켜서 최적의 조합을 찾는 방법.
- 단점 : 모든 가능성을 살펴보기에 하이퍼파라미터나 데이터가 많은 경우, 시간이 매우 오래 걸림




```
from sklearn.model_selection import GridSearchCV
params = {
    'min_impurity_decrease' : np.arange(0.0001, 0.001, 0.0001),
    'max_depth' : range(5, 20, 1),
    'min_samples_split' : range(2, 100, 10)
}
gs = GridSearchCV(estimator = DecisionTreeClassifier(), param_grid = params, n_jobs=-1)
gs.fit(train_input, train_target)
```

# 2. Cross Validation

[ 하이퍼파라미터 튜닝 ]

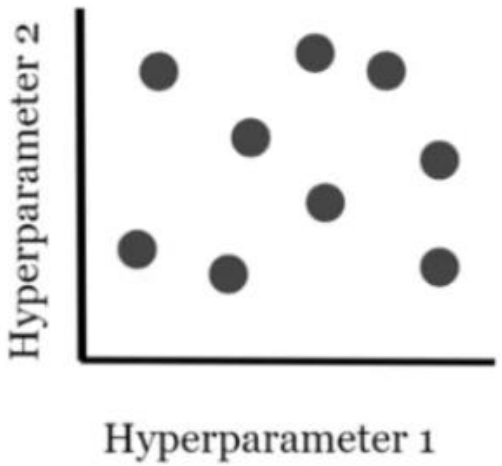
## 2. Random Search CV

- 주어진 범위 내에서 임의의 조합을 추출하여 최적의 조합을 탐색하는 방법
- 하이퍼파라미터의 개수가 많을 때 또는 해당 목록을 전달하기 어려울 때 유용한 방법

**Random Search**

Pseudocode

```
Hyperparameter_One = random.num(range)
Hyperparameter_Two = random.num(range)
```



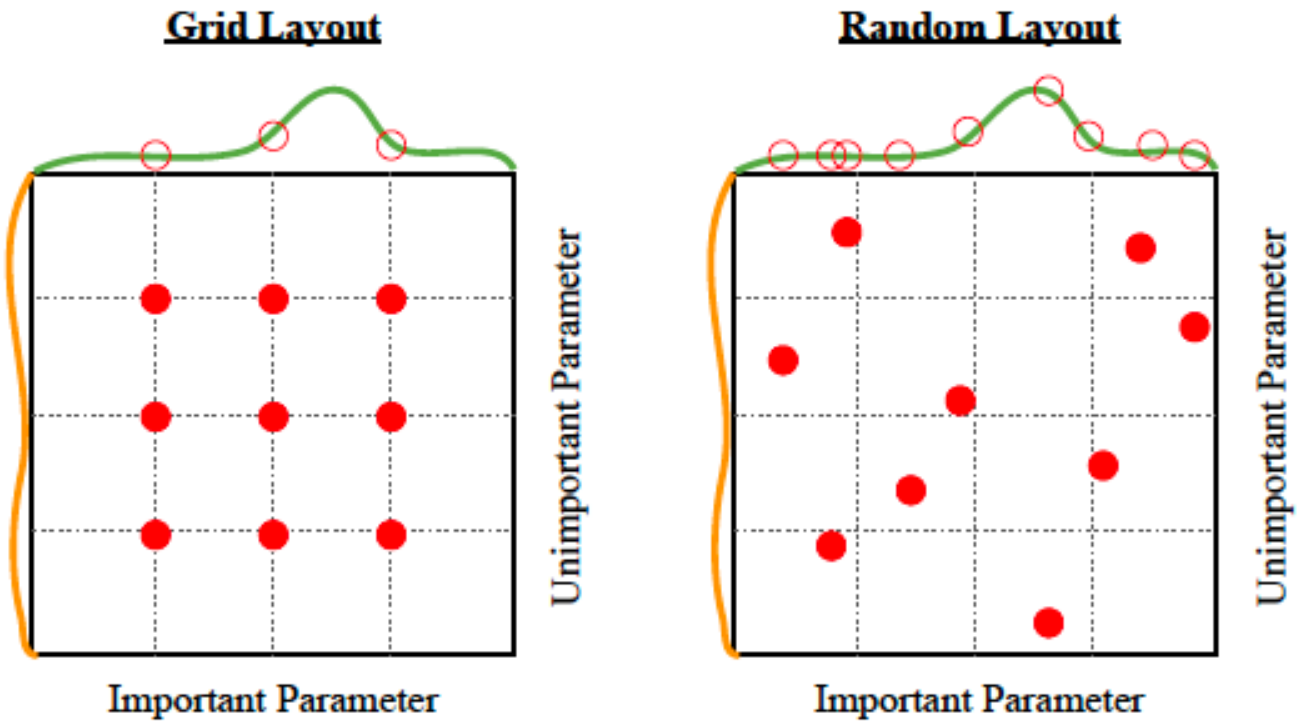

```python
from sklearn.model_selection import RandomizedSearchCV
gs = RandomizedSearchCV(estimator = DecisionTreeClassifier(), param_distributions = params, n_jobs=-1)
gs.fit(train_input, train_target)
```

# 2. Cross Validation

[ 하이퍼파라미터 튜닝 ]

## 2. Random Search CV

- 주어진 범위 내에서 임의의 조합을 추출하여 최적의 조합을 탐색하는 방법
- 하이퍼파라미터의 개수가 많을 때 또는 해당 목록을 전달하기 어려울 때 유용한 방법



# 3. Ensemble

## (1) 앙상블 학습

: 여러 개의 약한 분류기를 생성하여 그 예측을 결합함으로써 보다 정확한 최종 예측을 도출하는 기법

## (2) 앙상블 학습의 목표

- 다양한 분류기의 예측 결과를 결합함으로써 단일 분류기 보다 신뢰성이 높은 예측 값을 얻는 것

## (3) 앙상블의 특징

- 단일 모델의 약점을 다수의 모델들을 결합하여 보완
- 뛰어난 성능을 가진 모델로만 구성하는 것보다, 성능이 떨어지는 서로 다른 유형의 모델을 섞는 것이 전체 성능에 도움이 될 수 있음
- 예) Random Forest의 기반 알고리즘인 Decision Tree -> '과적합'이라는 단점 -> 수 천개의 분류기 결합을 통해 성능 보완

## (4) 앙상블의 유형

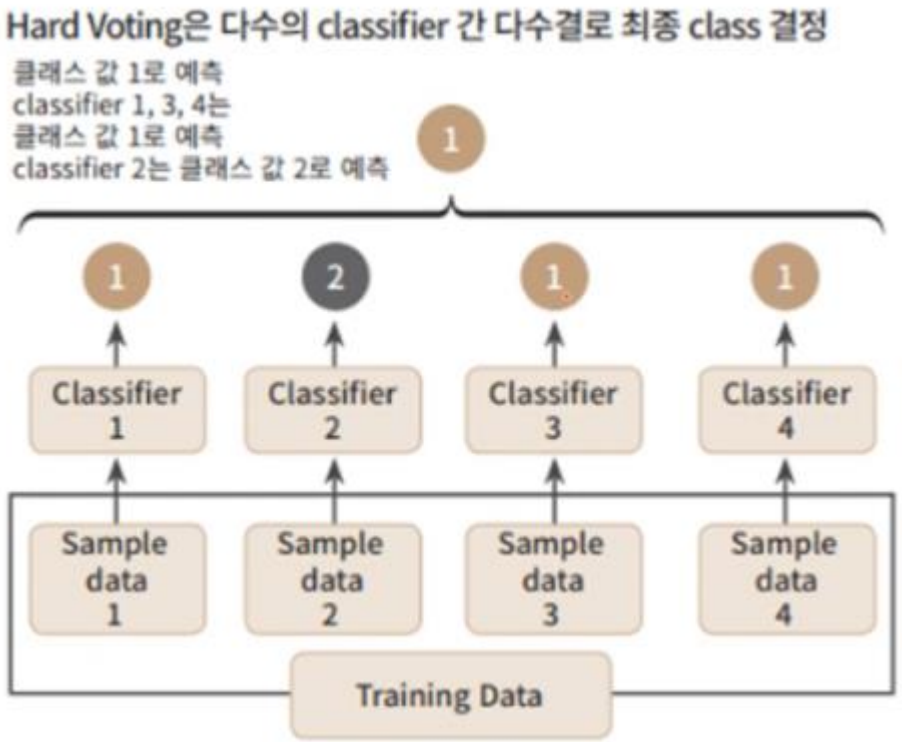
- 보팅 (Voting), 배깅 (Bagging), 부스팅 (Boosting)

# 3. Ensemble

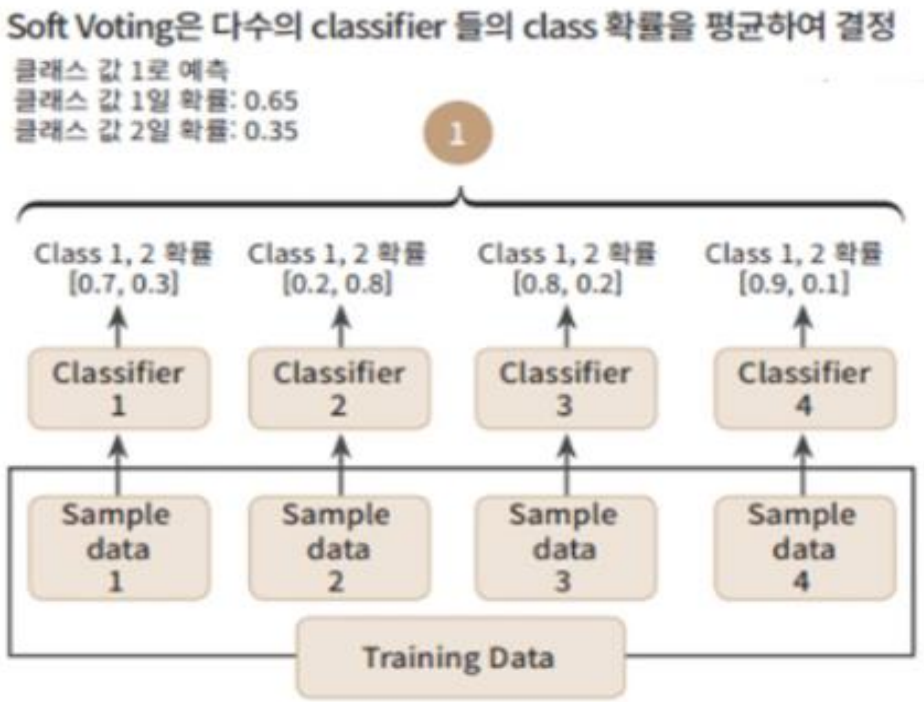
[ 보팅 (Voting) ]

같은 데이터에 대해 서로 다른 알고리즘을 가진 여러 개의 분류기가 투표를 통해 최종 예측 결과를 결정하는 방식

< 하드 보팅 >




< 소프트 보팅 >



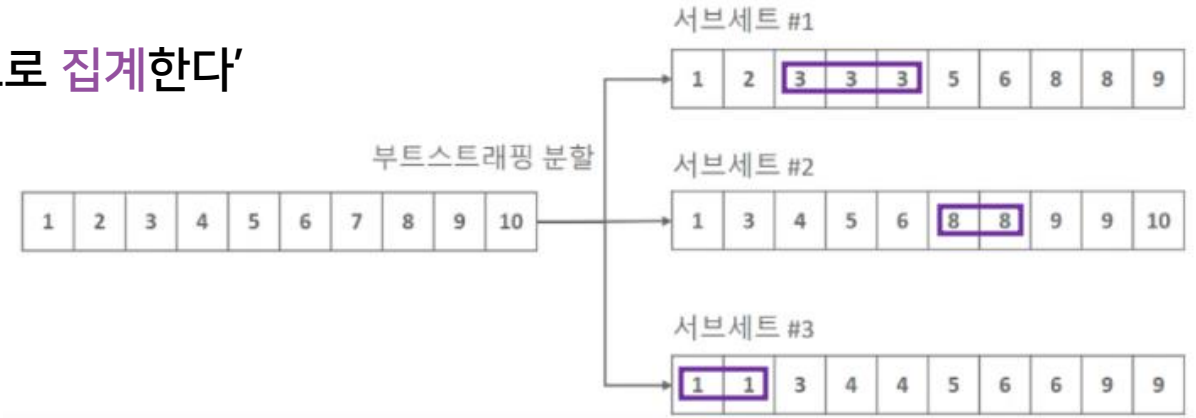

[0.7, 0.3], [0.2, 0.8], [0.8, 0.2], [0.9, 0.1]

- 클래스 1일 확률 : $(0.7 + 0.2 + 0.8 + 0.9) / 4 = 0.65$
- 클래스 2일 확률 : $(0.3 + 0.8 + 0.2 + 0.1) / 4 = 0.35$

# 3. Ensemble

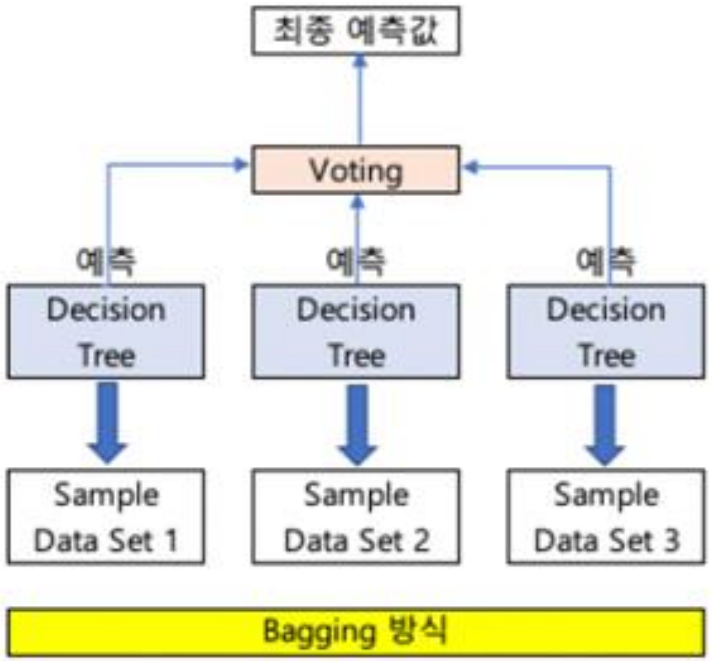
[ 배깅 (Bagging) ]

Bagging = Bootstrap + Aggregating → '부트스트랩 방식으로 집계한다'




Categorical Data – Voting 방식으로 결과 집계

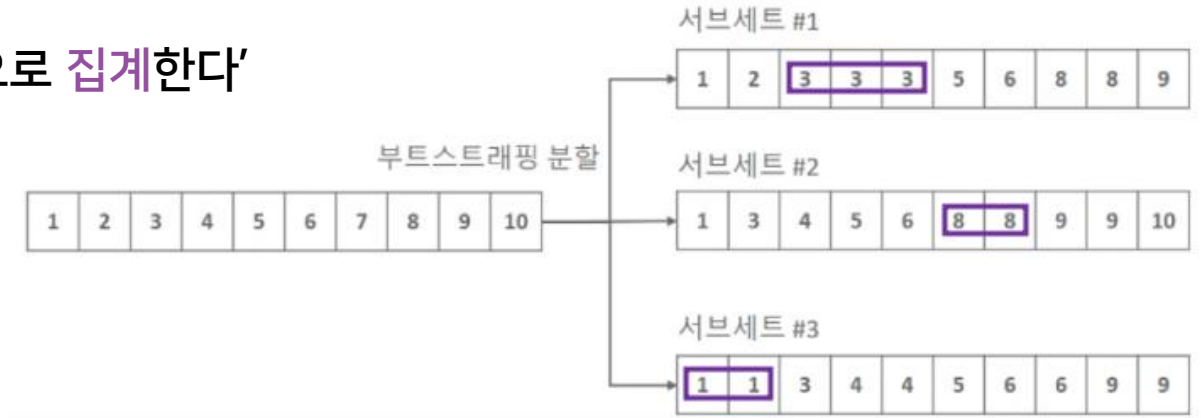
Continuous Data – 평균으로 결과 집계

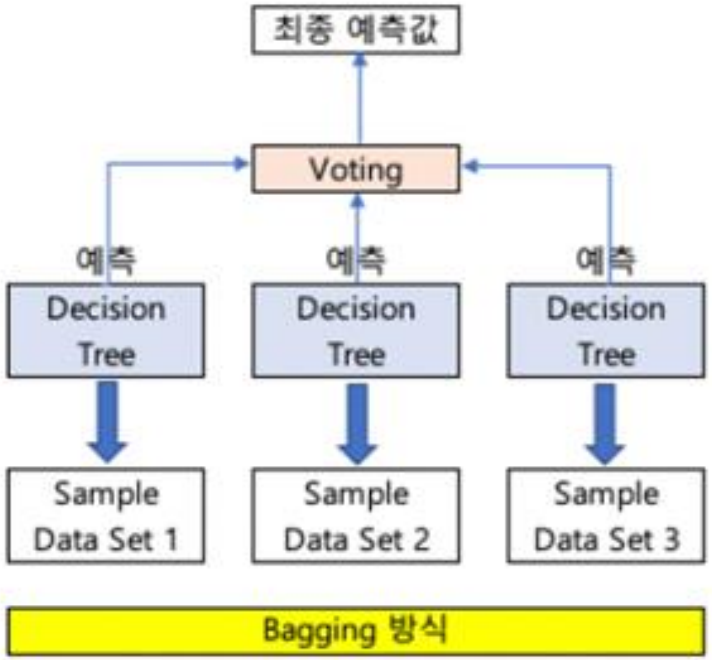


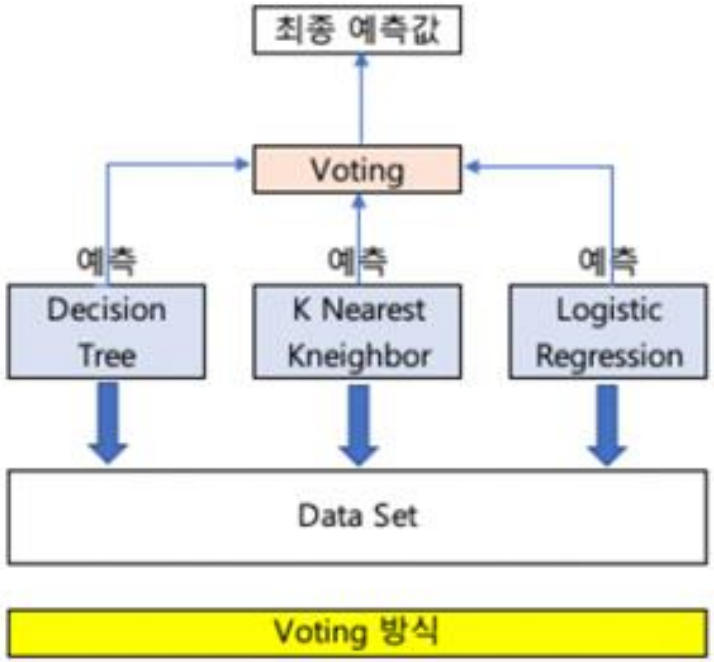
# 3. Ensemble

[ 배깅 (Bagging) ]

Bagging = Bootstrap + Aggregating → '부트스트랩 방식으로 집계한다'



# 3. Ensemble

[ 배깅 (Bagging) ]

집계 방식의 다양함

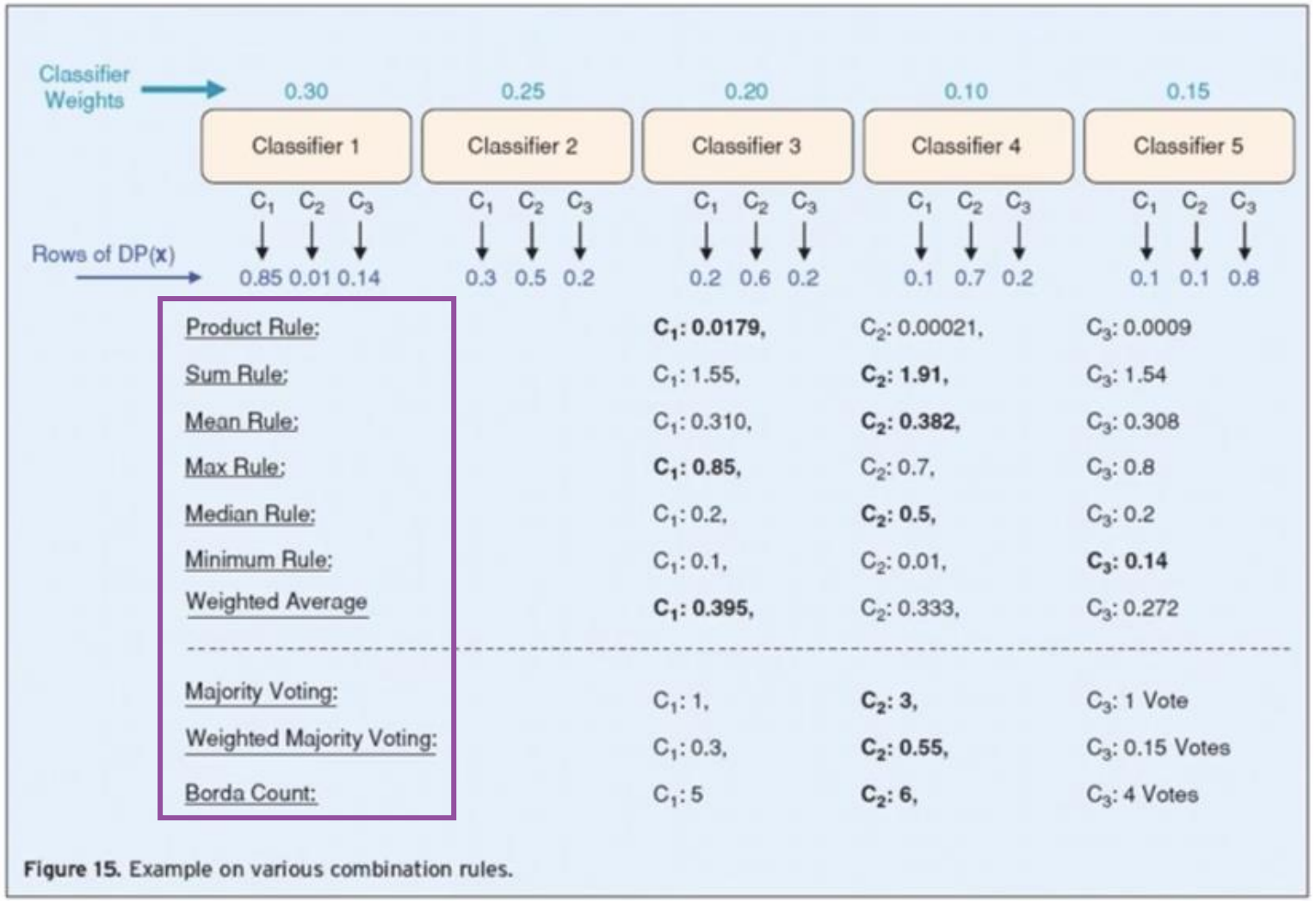
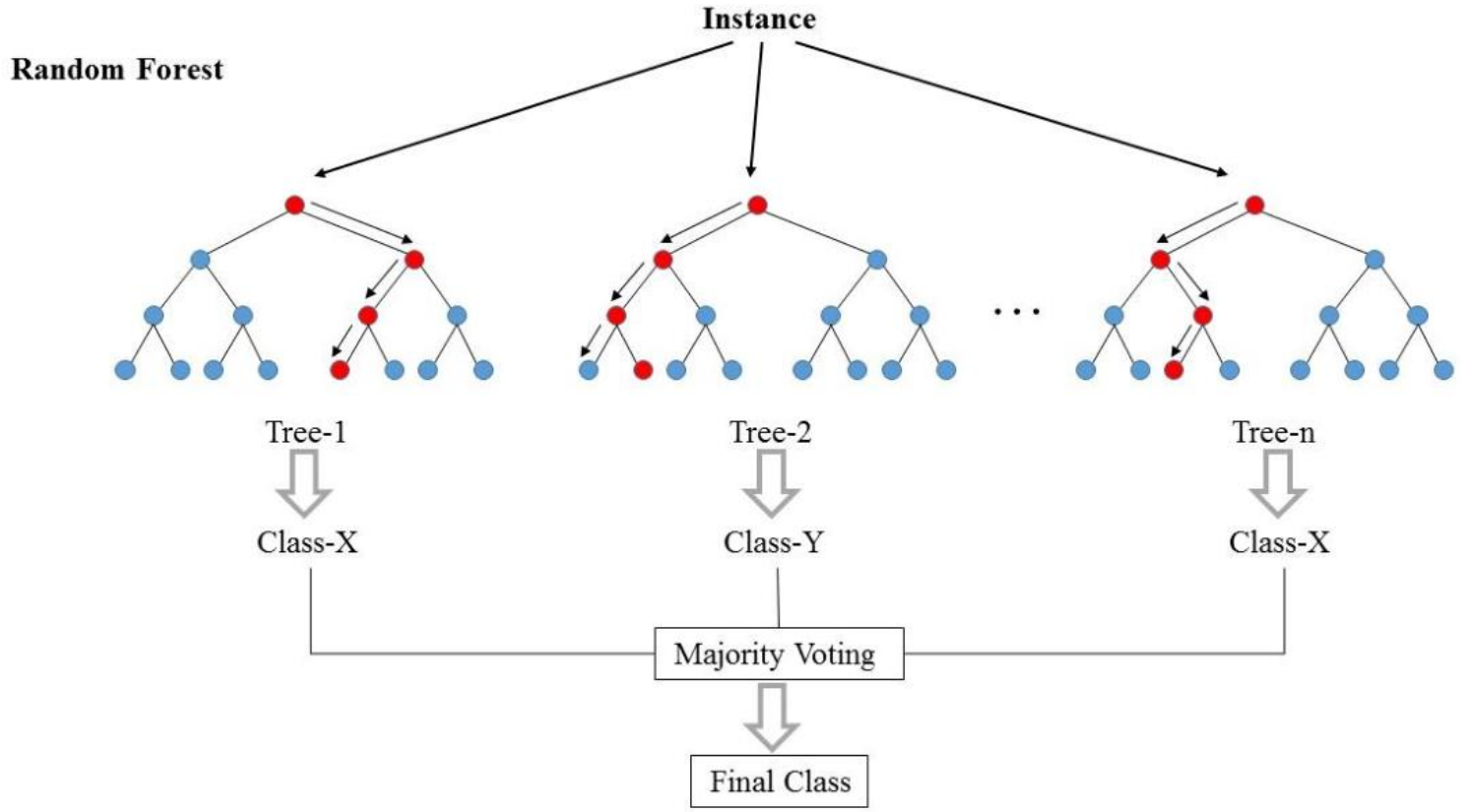



**Figure 15.** Example on various combination rules.

# 3. Ensemble

[ 배깅 (Bagging) ]

랜덤 포레스트 (Random Forest)




< 핵심 parameter >

(1) n_estimators : 랜덤 포레스트에서 결정 트리의 개수 지정
- 크게 설정할수록 좋은 성능 기대할 수 있음.
- 크게 설정할수록 학습 수행 시간이 오래 걸림

(2) max_features : 학습에 사용하는 최대 피처의 개수
- 서브 세트 별로 피처의 종류가 다를 수 있음

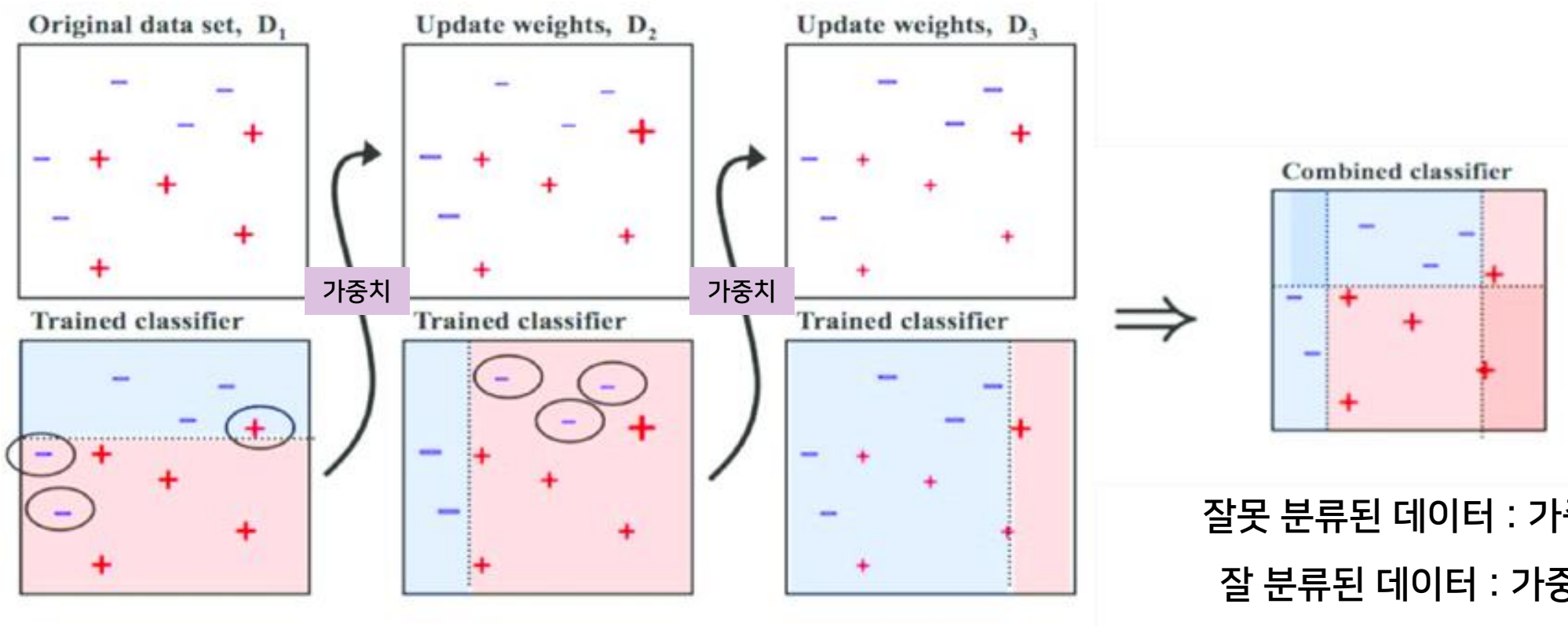
# 3. Ensemble

[ 부스팅 (Boosting) ]

가중치를 활용하여 약한 분류기를 강한 분류기로 만드는 방법

예) AdaBoost, GBM, XGBoost, LightGBM

< AdaBoost (Adaptive Boosting) >



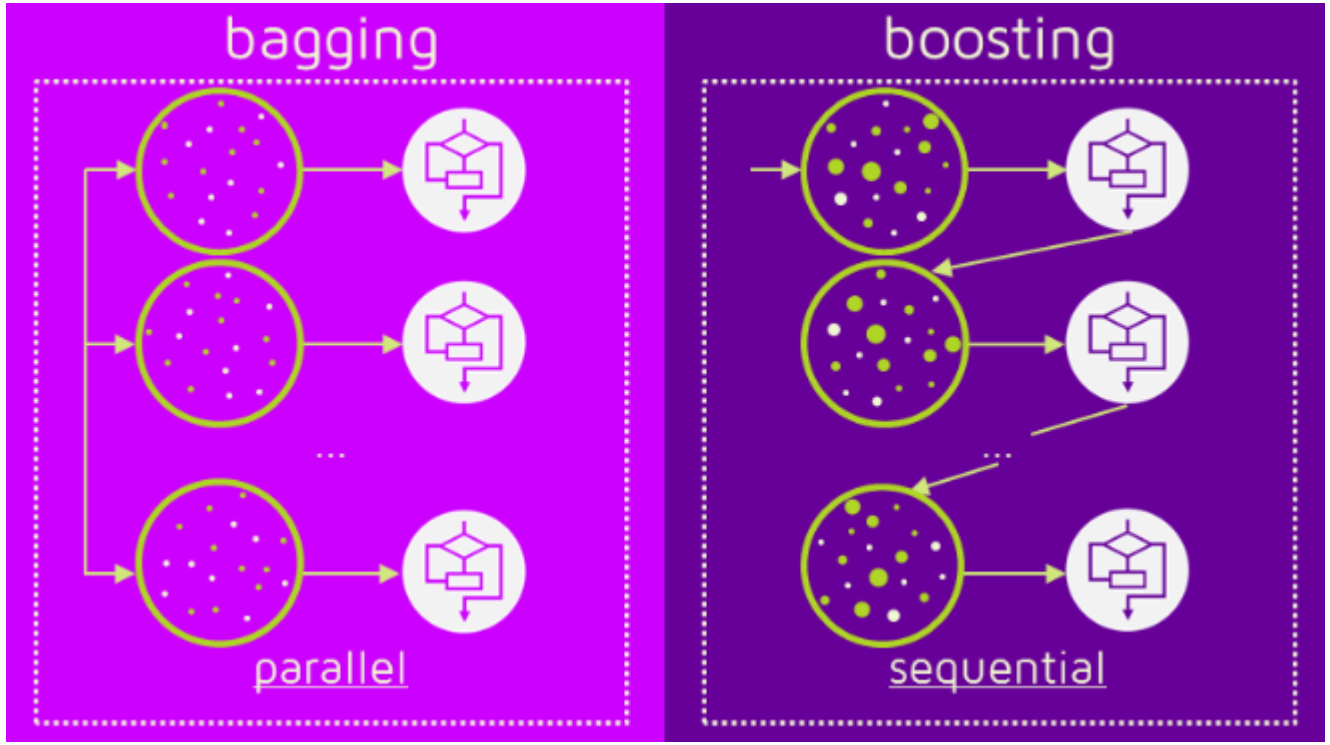

잘못 분류된 데이터 : 가중치 ↑

잘 분류된 데이터 : 가중치 ↓

# 3. Ensemble

[ 부스팅 (Boosting) ]

- 장점 : 배깅에 비해 성능이 좋다.
- 단점 : 속도가 느리다, 오버피팅 될 가능성이 있다.



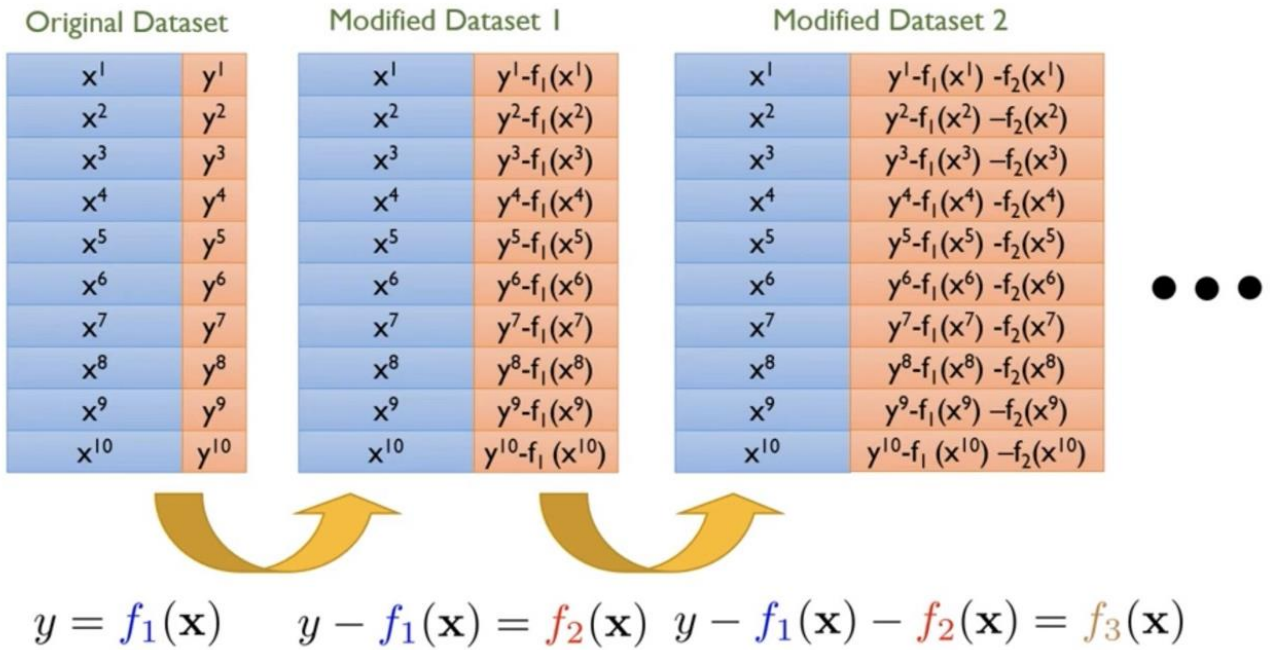

- 오버피팅이 발생할 가능성이 높은 경우
- 높은 분산을 가진 모델을 개선하려는 경우
- 정확도를 높이고자 하는 경우
- 낮은 편향을 가진 모델을 개선하려는 경우

# 3. Ensemble

[ 부스팅 (Boosting) ]

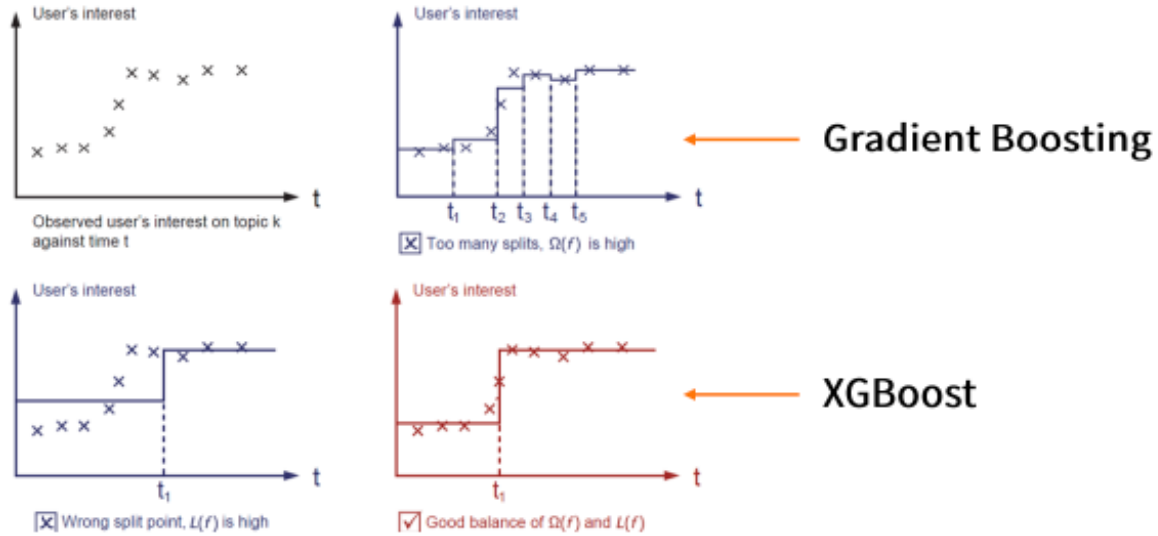
< GBM (Gradient Boosting Machine) >

- Gradient Descent + Boosting
- 손실함수 : 로지스틱 손실함수 (분류), MSE (회귀)
- 손실함수의 미분으로 gradient를 구하고, 이를 다음 모델에 전달하여 잔차 줄이는 방식



# 3. Ensemble

[ 부스팅 (Boosting) ]




< XGBoost (Extreme Gradient Boosting) >

- GBM을 병렬 처리가 가능하도록 만든 것
- 병렬 처리로 학습, 분류 속도가 빠름
- XGBoost는 자체에 과적합 규제 기능이 있음
- Early Stopping(조기 종료) 기능이 있음

1. 더 빠른 학습
2. 더 빠른 예측 수행 시간
3. 더 작은 메모리 사용량
4. 카테고리형 피처의 자동 변환과 최적 분할

< LightGBM >

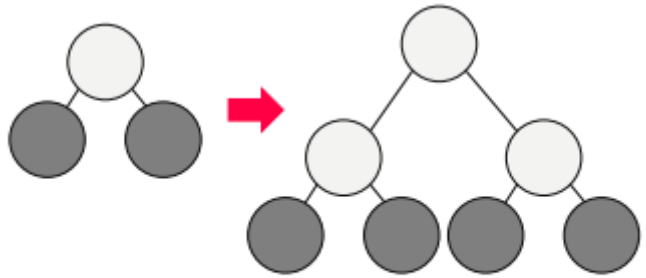
- XGBoost 이후에 Microsoft사에서 발전시킨 방식
- XGBoost의 예측 성능과 별다른 차이가 없고, 기능상의 다양성은 약간 더 많음
- 단점 : 적은 dataset 에 적용할 경우 과적합이 발생하기 쉬움

  ※ 적은 dataset : 일반적으로 10,000건 이하의 dataset 정도 (LightGBM 공식문서)

# 3. Ensemble

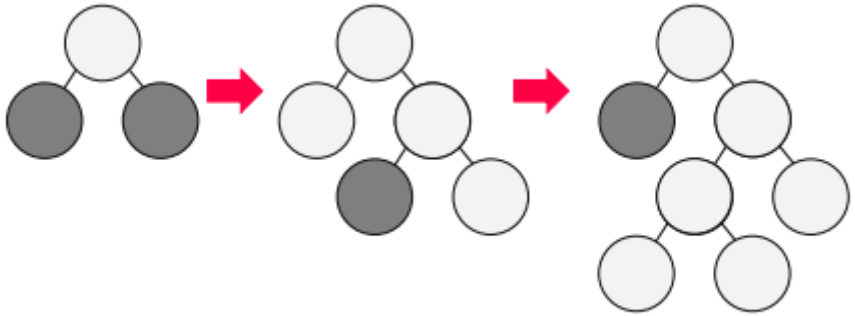
[ 부스팅 (Boosting) ]

< XGBoost >



Level-wise growth

< LightGBM >



Leaf-wise growth

- depth를 최소화 목표 : overfitting 방지

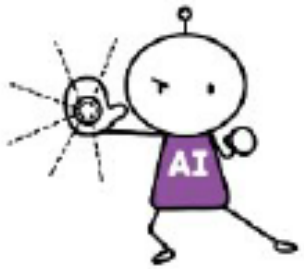
- 균형을 고려하지 않고 지속적으로 분할하여 비 대칭적인 tree 생성
- 깊이가 깊어지기는 하지만 학습을 반복하면서 level wise보다 예측 오류 손실을 최소화 할 수 있음

Undergraduate Research Internship in Affective AI LAB.

# 감사합니다 :>

[ 참고 자료 ]


https://bigdaheta.tistory.com/27
https://www.youtube.com/watch?v=d6nRgztYWQM&t=812s
https://m.blog.naver.com/winddori2002/221931868686
https://dacon.io/en/competitions/official/235946/codeshare/5623
https://quantdare.com/decision-trees-gini-vs-entropy/


컴퓨터공학과 12201682 공지윤