# 파머완 04 분류 part 1

# 01 분류(lassification)의 개요

## 지도학습 - 분류

: 레이블이 있는 데이터가 주어진 상태에서 학습하는 머신러닝 방식

학습 데이터로 주어진 데이터의 피처와 레이블값을 머신러닝 알고리즘으로 학습해 모델 생성

→ 새로운 데이터 값이 주어졌을 때 레이블을 예측

| 분류 방법 | 특징 |
|---|---|
| Naive Bayes | 베이즈 통계와 생성 모델 기반 |
| Logistic Regression | 독립변수와 종속변수의 선형 관계성에 기반 |
| Decision Tree | 데이터 균일도에 따른 규칙 기반 |
| Support Vector Machine | 개별 클래스 간의 최대 분류 마진 |
| Nearest Neighbor | 근접 거리 기준 |
| Neural Network | 심층 연결 기반 |
| Ensemble | 서로 다른 ML 결합 |

## Ensemble

: 여러개의 약한 학습기들을 결합해 가중치를 업데이트하며 성능 향상

기본 알고리즘: 결정트리

- 데이터의 스케일링이나 정규화 등의 사전 가공 영향 적음
- 성능 향상을 위해 복잡한 규칙 구조 ⇒ 과적합 발생

| **앙상블 기법** | **세부 알고리즘** |
|---|---|
| Bagging | Random Forest |

| | |
|---|---|
| Boosting | - Gradient Boosting<br>- XgBoost<br>- LightGBM |

높은 예측 성능 ⇒ 정형 데이터의 예측 분석 영역에서 많이 사용됨

---

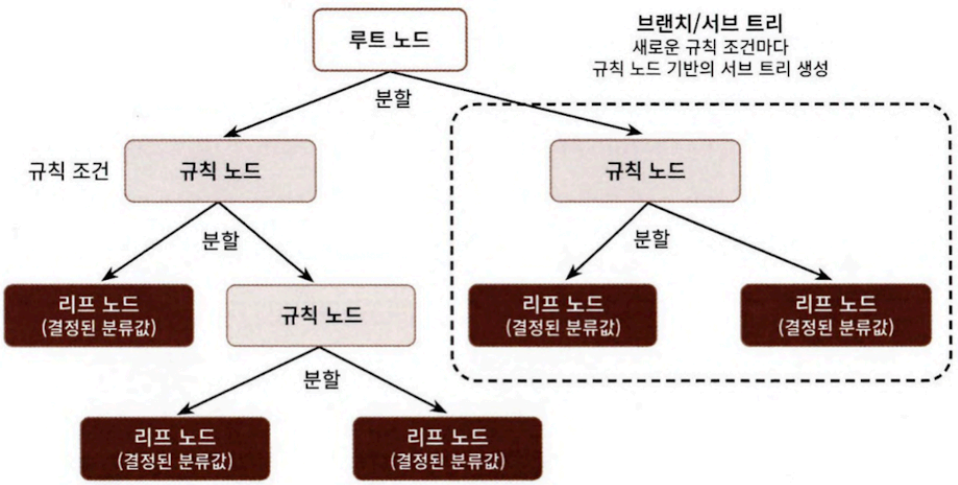
# 02 결정 트리

데이터에 있는 규칙을 학습을 통해 자동으로 찾아내 Tree 기반의 분류 규칙 만듦

어떤 기준으로 규칙을 만들어야 효율적인지가 성능을 좌우함

## 트리 구조

- 루트 노드(Root Node)
- 규칙 노드(Decision Node)
- 리프 노드(Leaf Node): 결정된 클래스 값
- 서브 트리(Sub Tree): 새로운 조건마다 생성
- 깊이(depth): 깊이가 깊어질수록 성능이 저하될 가능성이 높음

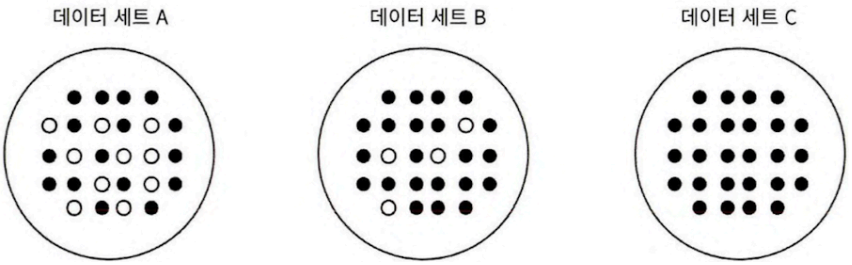



가능한 적은 결정 노드로 예측 정확도를 높이려면?

⇒ 분류할 때 최대한 많은 데이터가 해당 분류에 속하도록 규칙을 정해야함

Split: 최대한 균일한 데이터 세트를 구성하도록 분할하는 것이 필요

## 균일도




균일한 순서: C > B > A

- A: 혼잡도가 높고 균일도가 낮음 ⇒ 같은 조건에서 데이터를 판단하기 힘듦

결정 노드: 정보 균일도가 높은 데이터 세트를 먼저 선택하도록 규칙 조건 생성

1. 정보 균일도가 데이터 세트로 쪼개지도록 하는 조건을 찾아 서브 데이터 세트 생성
2. 서브 데이터 세트에서 균일도가 높은 자식 데이터 세트를 쪼개는 방식을 자식 트리로 내려가면서 반복

### 정보 균일도 측정 방법

1. Information Gain: 엔트로피를 기반으로 측정
    - 엔트로피: 데이터 집합의 혼잡도 - 서로 다른 값이 섞이면 높음
    - 정보 이득 지수: 1 - 엔트로피 지수
    - 정보 이득이 높은 속성을 기준으로 분할 = 엔트로피가 낮은 속성
2. 지니 계수: 지니 계수가 낮은 속성을 기준으로 분할 = 낮을수록 균일함

- 균일한 데이터 세트를 분할함

### 데이터 세트 분할 방식

1. 데이터 집합의 모든 아이템이 같은 분류에 속하는지 확인

2. If
    a. Ture → 리프 노드로 만들어서 분류 결정
    b. Else → 데이터를 분할하는 데 가장 좋은 속성과 기준을 찾음 ( Information Gain or 지니 계수 이용)
3. 해당 속성과 분할 기준으로 데이터를 분할하여 Branch 노드 생성
4. Recursive하게 모든 데이터 집합의 분류가 결정될 때까지 수행

# 결정 트리 모델의 특징

| 장점 | 단점 |
| --- | --- |
| 균일도라는 룰 기반 → 쉽고 직관적 | 과적합으로 정확도가 떨어짐 |
| 룰이 명확하고 노드가 생성되는 과정을 시각화로 표현 가능 | **서브 트리를 계속 생성한다면?** 피처가 많고 균일도가 다양할수록 트리의 깊이가 커짐 |
| 전처리 작업 필요 x | 크기를 사전에 제한하는 튜닝이 필요함 |

# 결정 트리 파라미터

- DecisionTreeClassifier
- DecisionTreeRegressor
- CART(Classfication And Regression Trees)

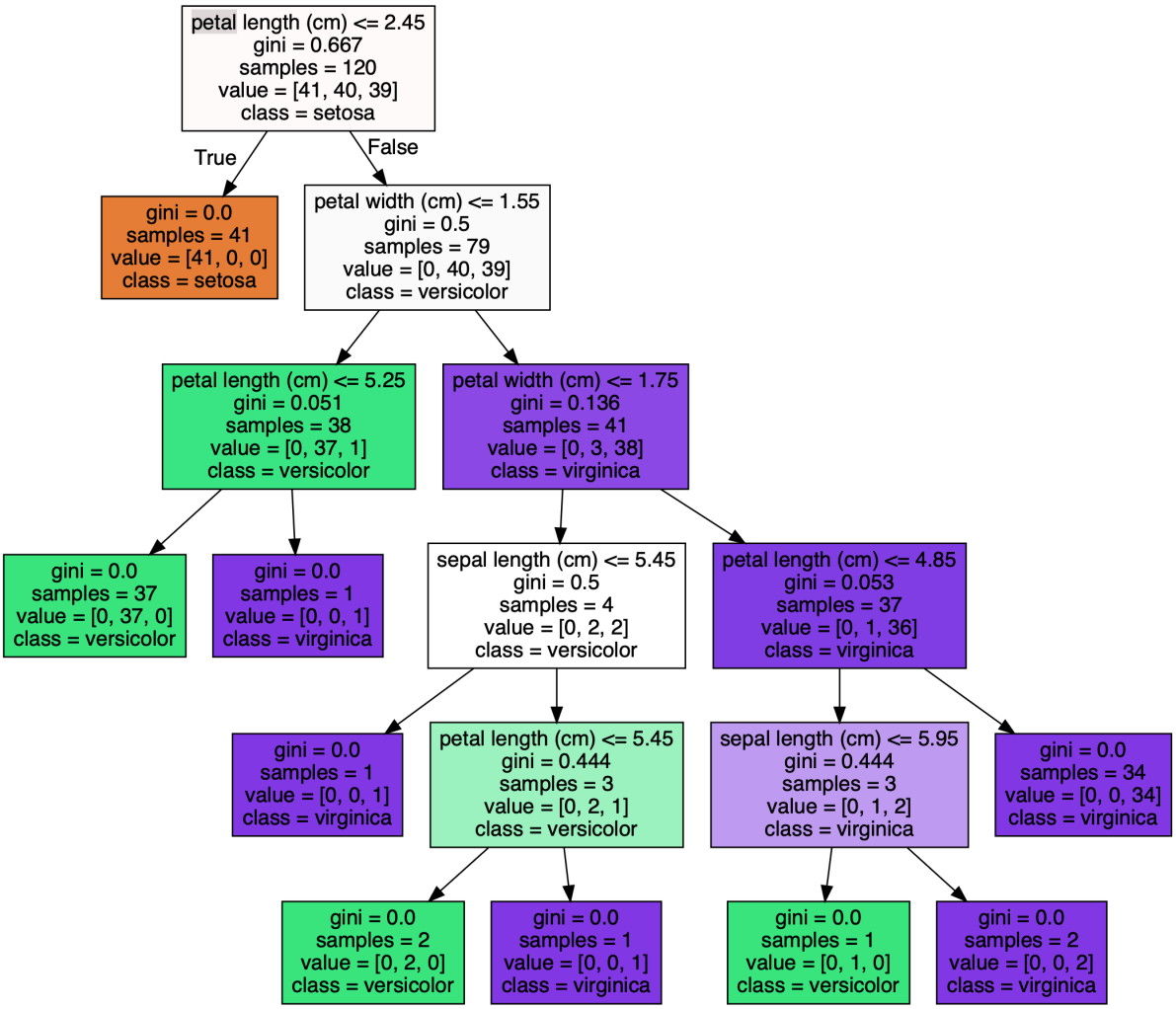
| 파라미터 | 설명 | 특징 |
| --- | --- | --- |
| min_samples_split | 노드 분할을 위한 최소 샘플 수 (default=2) | - 과적합 제어에 사용<br>- 작게 설정할수록 분할 노드가 많아짐 |
| min_samples_leaf | 왼쪽과 오른쪽 브랜치 노드에서의 최소 샘플 수 | - 크게 설정할수록 노드 분할을 덜 수행<br>- 과적합 제어<br>- 비대칭적 데이터는 작게 설정 필요 |
| max_features | 최대 피처 개수 (default=None, 모든 피처 사용) | - int: 대상 피처 개수<br>- float: 대상 피처의 퍼센트<br>- sqrt: root(전체피처개수)<br>- auto: sqrt와 동일<br>- log: log2(전체피처개수)<br>- None: 전체 피처 선정 |
| max_leaf_nodes | 말단 노드(leaf)의 최대 개수 | |

# 결정 트리 모델의 시각화

Graphviz 패키지 사용: 그래프 기반의 dot 파일로 기술된 이미지를 시각화

- export_graphviz(Estimator,피처 이름 리스트, 레이블 이름 리스트): 학습된 결정 트리 규칙을 실제 트리 형태로 시각화

## 결정 트리 생성




노드 색깔: 붓꽃 데이터의 레이블 값

- 주황색 0:setosa, 초록색 1:versicolor, 보라색 2:virginica
- 색깔이 짙어질수록 지니 계수가 낮고 레이블에 속하는 데이터가 많음

  ⇒ 균일도가 높음

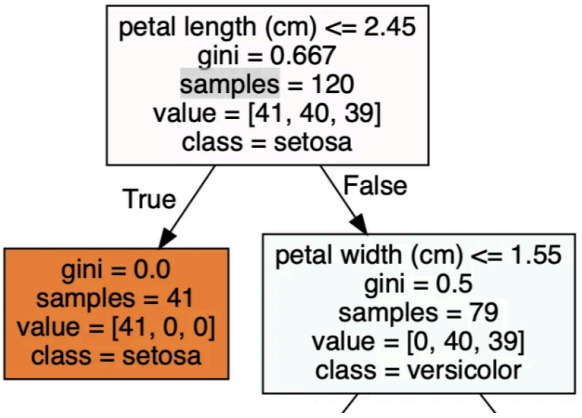
리프 노드: 최종 클래스 값이 결정되는 노드

- 하나의 클래스 값으로 최종 데이터가 구성되거나
  리프 노드가 될 수 있는 하이퍼 파라미터 조건을 충족하면 됨

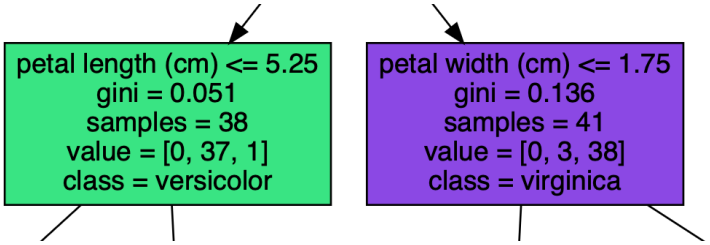
## 결정 트리 요소

자식 노드를 만들기 위한 규칙 조건 ⇒ 이 조건이 없으면 리프 노드

- gini: 다음의 value=[]로 주어진 데이터 분포에서의 지니 계수
- samples: 현 규칙에 해당하는 데이터 수
- value=[]: 클래스 값 기반의 데이터 건수




| 노드 | 조건 | 데이터 분포 | 지니 계수 |
|---|---|---|---|
| 1번 노드 | petal length ≤ 2.45 | 전체 120개<br>(Setosa 41, Versicolor 40, Virginica 39) | 0.667 |
| 2번 노드 | petal length ≤ 2.45<br>(True) | 전체 41개<br>(모두 Setosa) | 0 |
| 3번 노드 | petal length ≤ 2.45<br>(False)<br>→ petal width ≤ 1.55 | 전체 79개<br>(Versicolor 40, Virginica 39) | 0.5 |




| **노드** | 조건 | 데이터 분포 | 지니 계수 |
|---|---|---|---|
| 4번 노드 | petal width ≤ 1.55 (True)<br>→ petal length ≤ 5.25 | 전체 38개 (Versicolor: 37, Virginica: 1) | 지니 계수 낮음 = 클래스 혼재 |
| 5번 노드() | petal width ≤ 1.55 (False)<br>→ petal width ≤ 1.75 | 전체 41개(versicolor 3, Virginica 28) | 지니 계수 낮음 = 클래스 혼재 |

- 문제점: 규칙 생성 로직을 미리 제어하지 않으면, 트리 노드를 계속해서 생성 → 과적합 가능성이 높음

  ⇒ 복잡한 트리가 생성되는 것을 막기 위한 하이퍼 파라미터 필요

## 하이퍼 파라미터

- **max_depth**: 결정 트리의 최대 트리 깊이 제어
  - 제한 없음 → 3개 : 더 간단한 결정 트리
- **min_samples_splits**: 자식 규칙 노드를 분할하기 위한 최소한의 샘플 수
  - 3으로 설정 시: 샘플 수가 4개보다 적기 때문에 분할하지 않고 리프 노드가 됨 → 트리 깊이가 줄어 더 간단해짐
- **min_samples_leaf**: 분할될 경우, 자식 노드 각각의 최소 샘플 수
  - 값이 커지면 분할 될 조건이 어려워짐 → 브랜치 노드 감소, 간결

**feature importance**

: feature 중요도

- 명확한 규칙 트리를 만드는 데 기여
- 간결하고 Outlier에 강한 모델을 만들어줌

**feature_importances_** : DecisionTreeClassifier 제공

- 반환: ndarray 형태, 피처 순서대로 값 할당

- 값이 높을수록 중요도가 높음

## 결정 트리 과적합(Overfitting)

**make_classification()**

: 분류를 위한 테스트용 데이터를 쉽게 만들도록 하는 함수

반환: 피처 데이터 세트 & 클래스 레이블 데이터 세트

**visualize_boundary()**

: 머신러닝 모델이 클래스 값을 예측하는 결정 기준을 색상과 경계로 나타내어 모델이 어떻게 분류하는지 시각화 해주는 함수

## 결정 트리 실습 - 사용자 행동 인식 데이터 세트

중복된 피처명을 가질 경우, DataFrame에 로드시 오류 발생

깊이가 깊어질수록 테스트 데이터 세트의 정확도가 떨어짐

⇒ 하이퍼 파라미터를 이용하여 과적합을 방지해야함

---

# 03 앙상블 학습

## 앙상블 학습 개요

앙상블 학습(Ensemble Learning): 여러개의 Classifier를 생성하고 그 예측을 결합하여 정확한 최종 예측을 도출하는 기법

- 데이터 타입마다 다른 성능 차이
  - 비정형 데이터 분류(이미지, 영상, 음성): 딥러닝
  - 정형 데이터 분류: 앙상블

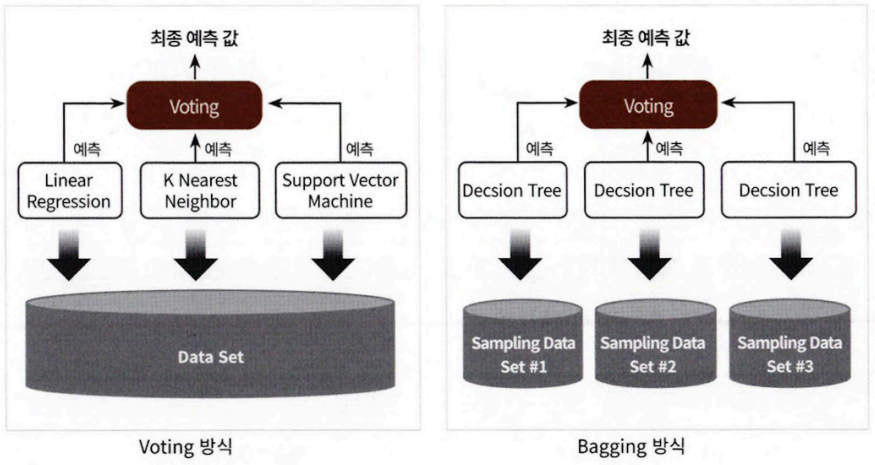
- 부스팅 계열의 앙상블 알고리즘 사용도 증가 → 기존의 그래디언트 부스팅을 뛰어남는 새로운 알고리즘 개발 가속화

** 앙상블 알고리즘은 쉽고 편하면서 강력한 성능을 가짐!

## 앙상블 학습의 유형

1. **Voting**
    - 여러개의 분류기가 투표를 통해 최종 예측 결과를 결정하는 방식
    - 서로 다른 알고리즘을 가진 분류기를 결합
2. **Bagging**
    - 여러개의 분류기가 투표를 통해 최종 예측 결과를 결정하는 방식(voting과 동일)
    - 같은 유형의 알고리즘 기반인 분류기를 결합
    - 데이터 샘플링을 서로 다르게 학습

    ex) RandomForest



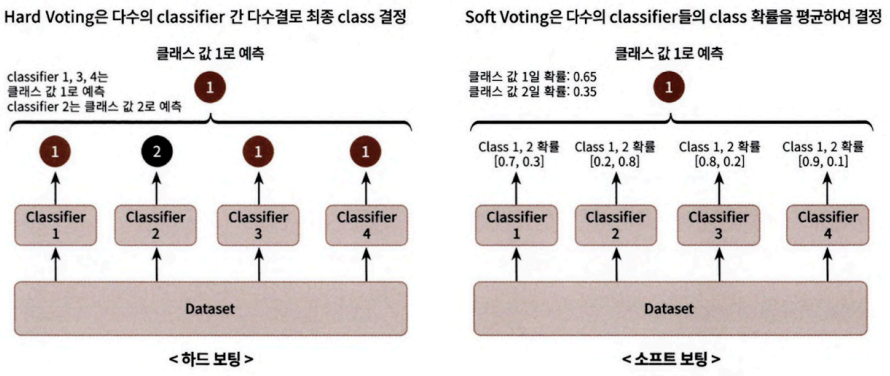

- **Voting**: 3개의 ML로 같은 데이터 세트에 대해 학습하고 예측한 결과를 Voting을 통해 최종 예측 결과 선정
- **Bagging**: 단일 ML로 개별 Classifier에게 데이터를 샘플링해서 추출하는 방식인 **Bootstrapping** 분할 방식으로 학습하고 예측한 결과를 Voting을 통해 최종 예측 결과 선정
    - 중첩 허용 → 10000개의 데이터를 10개로 나눌 경우, 1000개의 데이터에 중복된 데이터가 있음

3. **Boosting**

- 여러개의 분류기가 순차적으로 학습 → 앞에서 학습한 분류기가 **예측이 틀린 데이터**에 대해서는 다음 분류기에 **가중치(weight)**를 부여해 학습과 예측을 진행

# 보팅 유형 - 하드 보팅과 소프트 보팅

1. **Hard Voting**
    - 다수의 classifier 간 다수결로 최종 class 결정
2. **Soft Voting**
    - 다수의 classifier들의 class 확률을 평균하여 설정
    - 일반적으로 많이 사용




# 보팅 분류기(Voting Classifier)

주요 생성 인자

1. estimators: 리스트, 보팅에 사용될 여러 개의 classifier 객체들을 튜플로 입력 받음
2. voting: hard, soft 방식 (default:'hard')

## 앙상블 알고리즘 정리

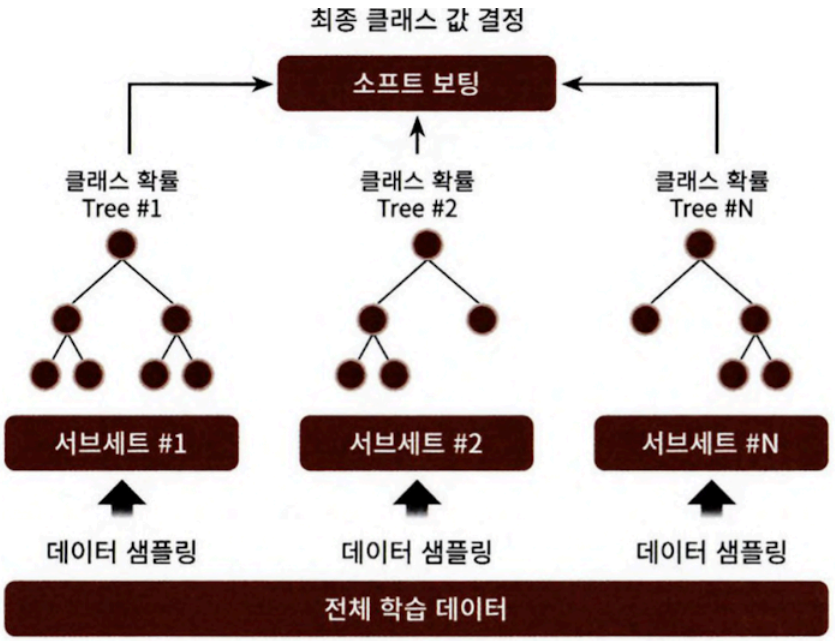
- 보팅, 스태킹 등은 서로 다른 알고리즘 기반
- 배깅, 부스팅은 결정 트리 알고리즘 기반 → 직관적인 분류 기준이 있지만 오히려 과적합이 발생하여 실제 테스트 데이터에서 성능이 떨어지기 쉬움
- 앙상블 학습: 많은 분류기들을 결합하여 다양한 상황에서 학습하게 함으로 결정 트리 알고리즘의 단점을 극복함
- 결정 트리 알고리즘의 장점 + 단점 보완 ⇒ 편향 - 분산 트레이드 오프의 효과를 극대화

# 04 랜덤 포레스트

## 랜덤 포레스트의 개요 및 실습

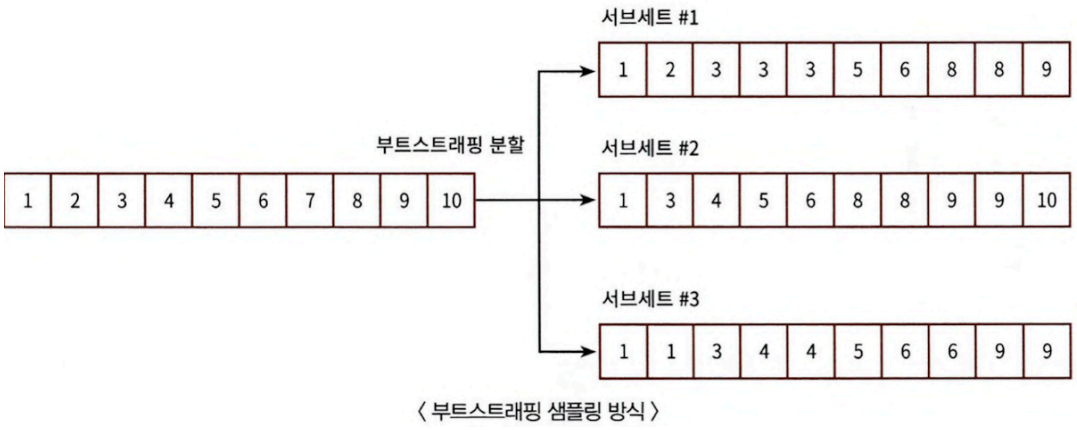
### 랜덤 포레스트

: 여러 개의 결정 트리 분류기가 전체 데이터에서 배깅 방식으로 각자의 데이터를 샘플링해 개별적으로 학습을 수행한 뒤 최종적으로 모든 분류기가 보팅을 통해 예측 결정




### 부트스트래핑(bootstrapping)

: 여러 개의 데이터 세트를 중첩되게 분리하는 것




〈부트스트래핑 샘플링 방식〉

# 랜덤 포레스트 하이퍼 파라미터 및 튜닝

## 트리 기반 앙상블 알고리즘 단점

하이퍼 파라미터가 너무 많음 → 튜닝 시간이 많이 소모됨

- 트리 기반 하이퍼 파라미터 + 배깅, 부스팅, 학습, 정규화 등의 하이퍼 파라미터까지 추가되어, 일반적인 ML 알고리즘에 비해 많음

## 랜덤 포레스트 하이퍼 파라미터

**n_estimators** : 결정 트리 개수 지정 (default=10)

**max_features** : 분할하는 피처를 참조할 때 전체가 아닌 sqrt(전체 피처 개수)만큼 참조

**max_depth, min_samples_leat, min_samples_split**