머신러닝 - 분류

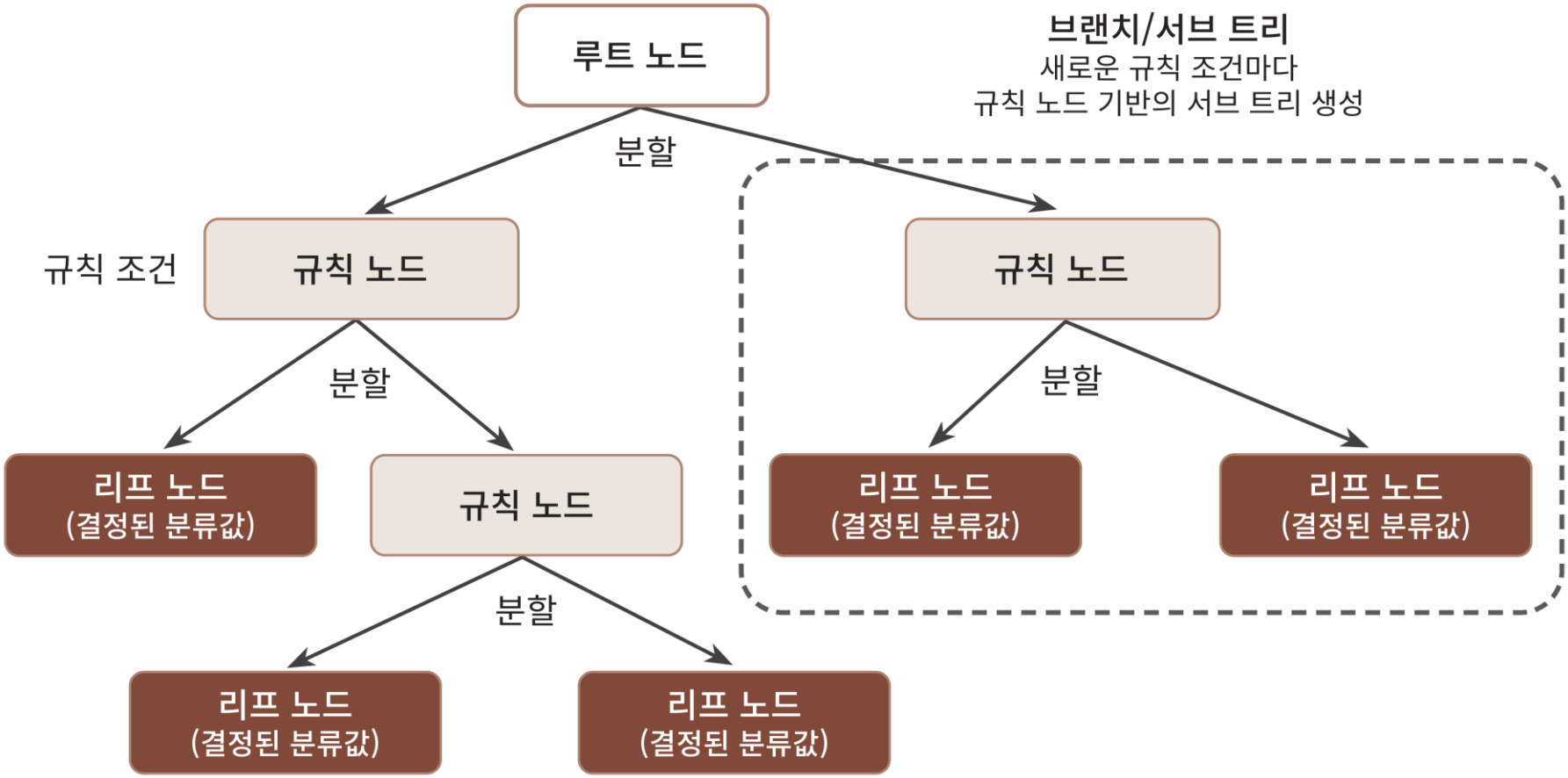
2021

1. 개요

❖ 분류 알고리즘의 종류

- 베이즈(Bayes) 통계의 생성 모델에 기반한 나이브 베이즈(Naïve Bayes)
- 독립 변수와 종속 변수의 선형 관계성에 기반한 로지스틱 회귀
- 데이터 균일도에 따른 규칙 기반의 결정 트리(Decision Tree)
- 개별 클래스간의 최대 분류 마진을 효과적으로 찾아주는 서포트 벡터 머신
- 근접 거리를 기준으로 하는 최소 근접(Nearest Neighbor) 알고리즘
- 심층 연결 기반의 신경망(Neural Network)
- 서로 다른(또는 같은) 머신 러닝 알고리즘을 결합한 앙상블(Ensemble)

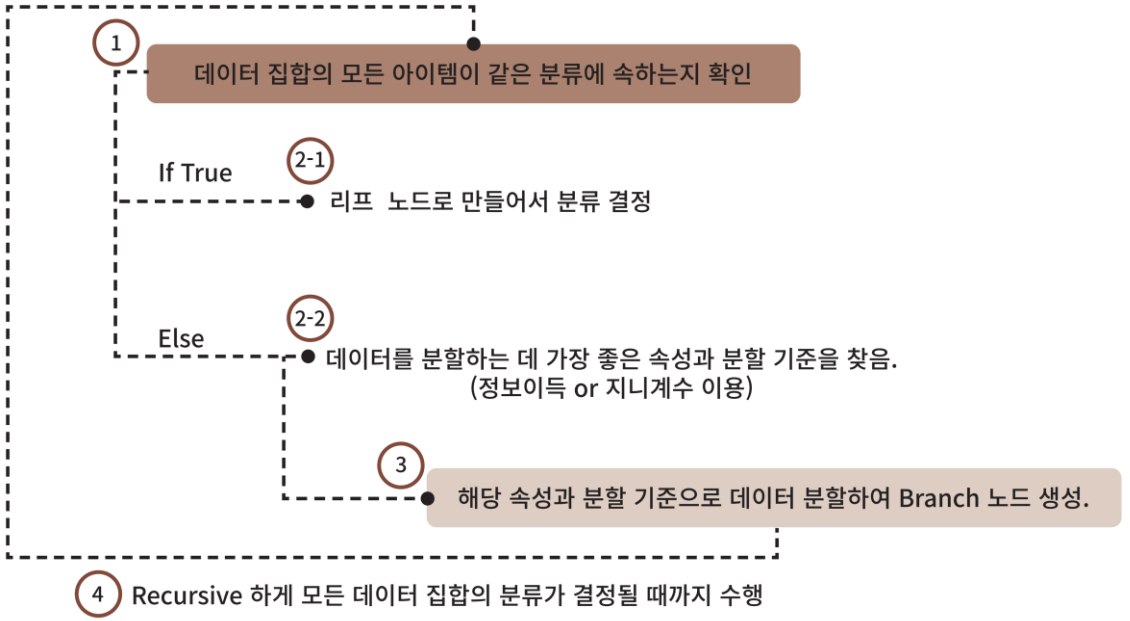
2. 결정 트리(Decision Tree)



2. 결정 트리(Decision Tree)

❖ 판단 기준

- 정보 균일도가 높은 데이터 세트를 먼저 선택하도록 규칙 조건을 생성
- 정보 균일도를 측정하는 방법
  - 엔트로피(혼잡도)를 이용한 정보 이득 지수, 즉 1 – 엔트로피 지수
    정보 이득이 높은 속성을 기준으로 분할
  - 지니 계수(0이 평등, 1이 불평등)
    지니 계수가 낮은 속성을
    기준으로 분할



2. 결정 트리(Decision Tree)

❖ 특징

결정 트리 장점	결정 트리 단점
• 쉽다. 직관적이다 • 피처의 스케일링이나 정규화 등의 사전 가공 영향도가 크지 않음.	• 과적합으로 알고리즘 성능이 떨어진다. 이를 극복하기 위해 트리의 크기를 사전에 제한하는 튜닝 필요.

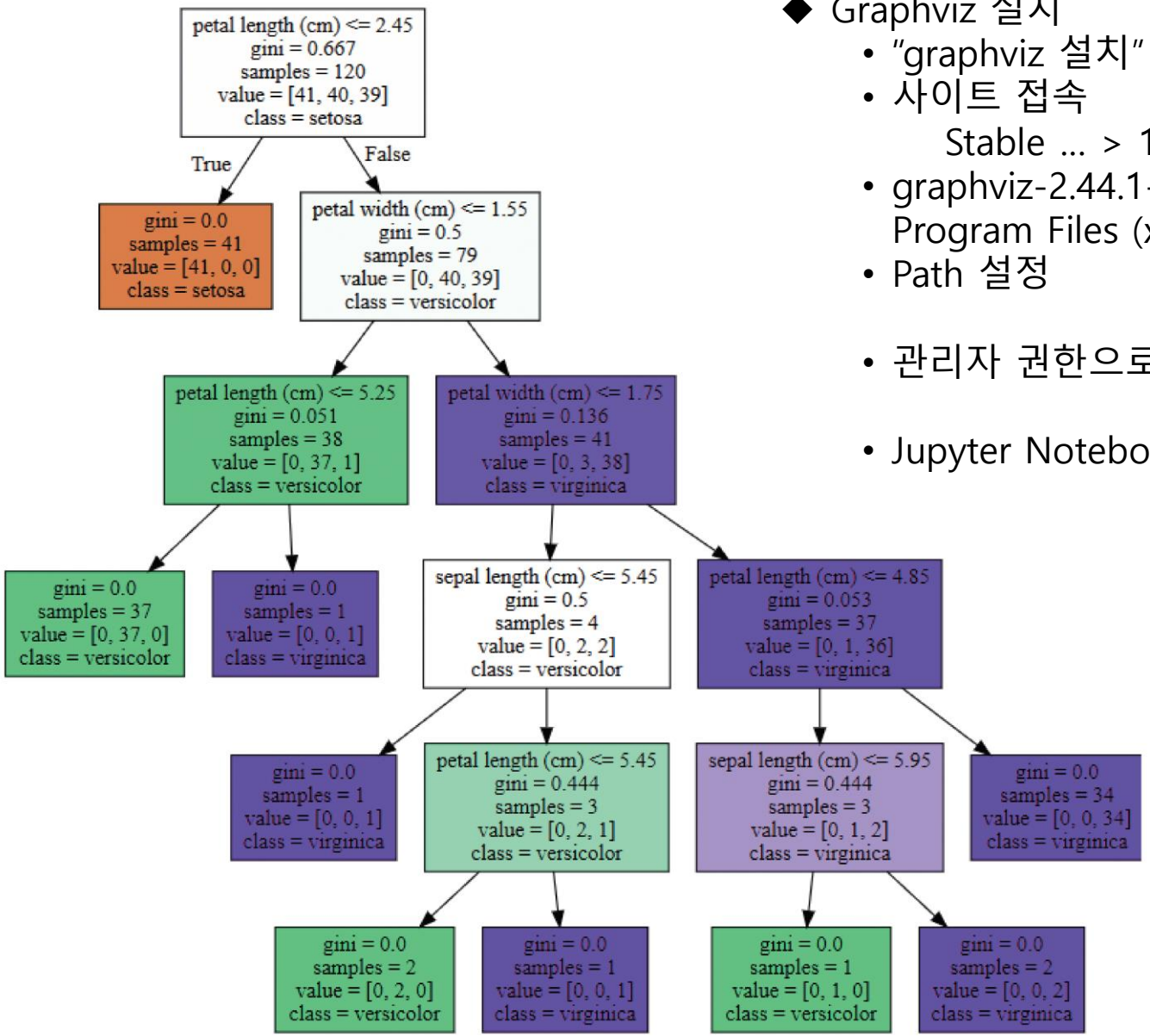
❖ 파라미터

파라미터 명	설명
min_samples_split	• 노드를 분할하기 위한 최소한의 샘플 데이터 수로 과적합을 제어하는 데 사용됨. • 디폴트는 2이고 작게 설정할수록 분할되는 노드가 많아져서 과적합 가능성 증가 • 과적합을 제어. 1로 설정할 경우 분할되는 노드가 많아져서 과적합 가능성 증가
min_samples_leaf	• 말단 노드(Leaf)가 되기 위한 최소한의 샘플 데이터 수 • Min_samples_split와 유사하게 과적합 제어 용도. 그러나 비대칭적(imbalanced) 데이터의 경우 특정 클래스의 데이터가 극도로 작을 수 있으므로 이 경우는 작게 설정 필요.

2. 결정 트리(Decision Tree)

max_features	• 최적의 분할을 위해 고려할 최대 피처 개수. 디폴트는 None으로 데이터 세트의 모든 피처를 사용해 분할 수행. • int 형으로 지정하면 대상 피처의 개수, float 형으로 지정하면 전체 피처 중 대상 피처의 퍼센트임 • 'sqrt'는 전체 피처 중 sqrt(전체 피처 개수), 즉 $\sqrt{\text{전체 피처 개수}}$만큼 선정 • 'auto'로 지정하면 sqrt와 동일 • 'log'는 전체 피처 중 log2(전체 피처 개수) 선정 • 'None'은 전체 피처 선정
max_depth	• 트리의 최대 깊이를 규정. • 디폴트는 None. None으로 설정하면 완벽하게 클래스 결정 값이 될 때까지 깊이를 계속 키우며 분할하거나 노드가 가지는 데이터 개수가 min_samples_split보다 작아질 때까지 계속 깊이를 증가시킴. • 깊이가 깊어지면 min_samples_split 설정대로 최대 분할하여 과적합할 수 있으므로 적절한 값으로 제어 필요.
max_leaf_nodes	• 말단 노드(Leaf)의 최대 개수

2. 결정 트리(Decision Tree)



◆ Graphviz 설치

- "graphviz 설치" 검색
- 사이트 접속

  Stable … > 10 > msbuild > Release > Win32
- graphviz-2.44.1-win32.zip 다운로드, 압축해제 후 Program Files (x86)에 위치시킴
- Path 설정

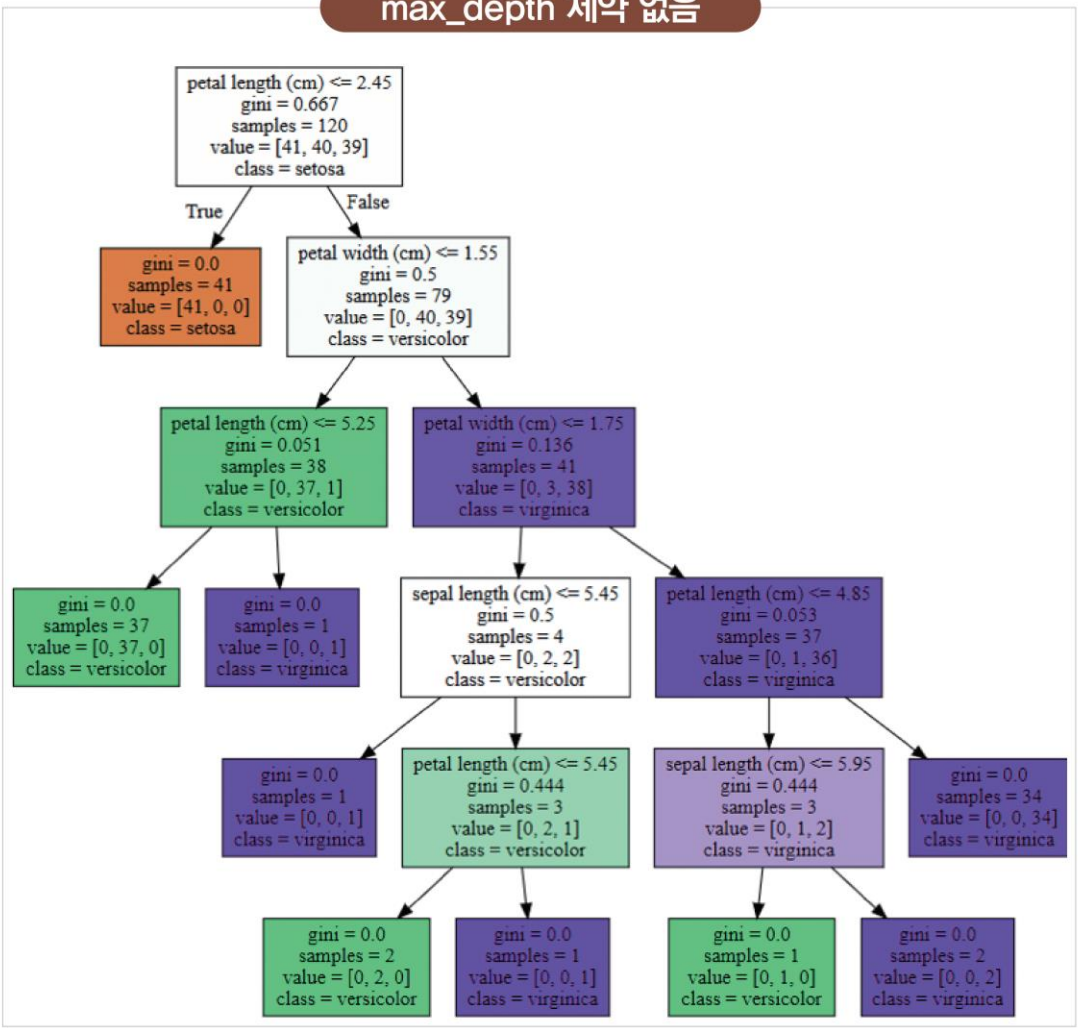
- 관리자 권한으로 pip install graphviz 실행

- Jupyter Notebook 다시 실행

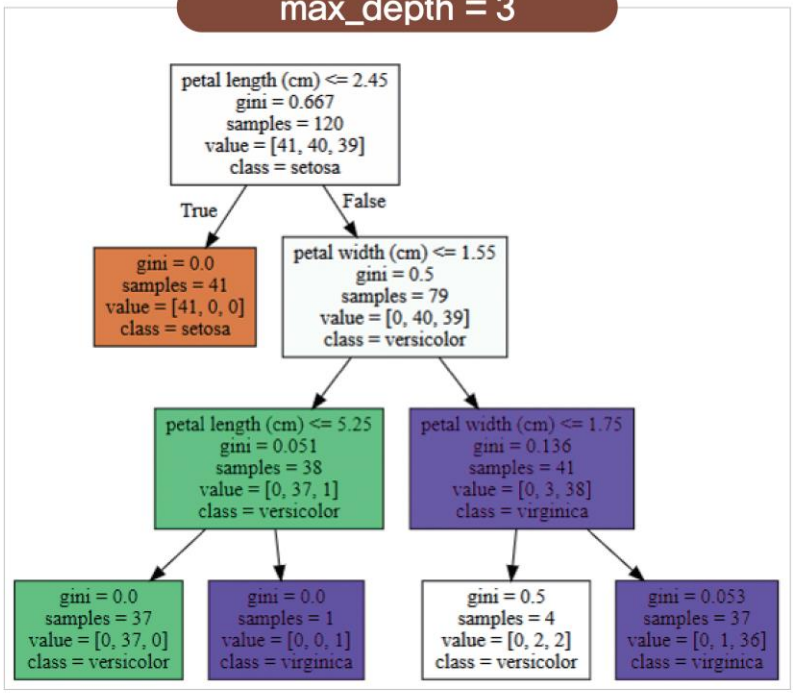
2. 결정 트리(Decision Tree)

❖ max_depth를 제한 없음에서 3으로 설정한 경우

max_depth 제약 없음



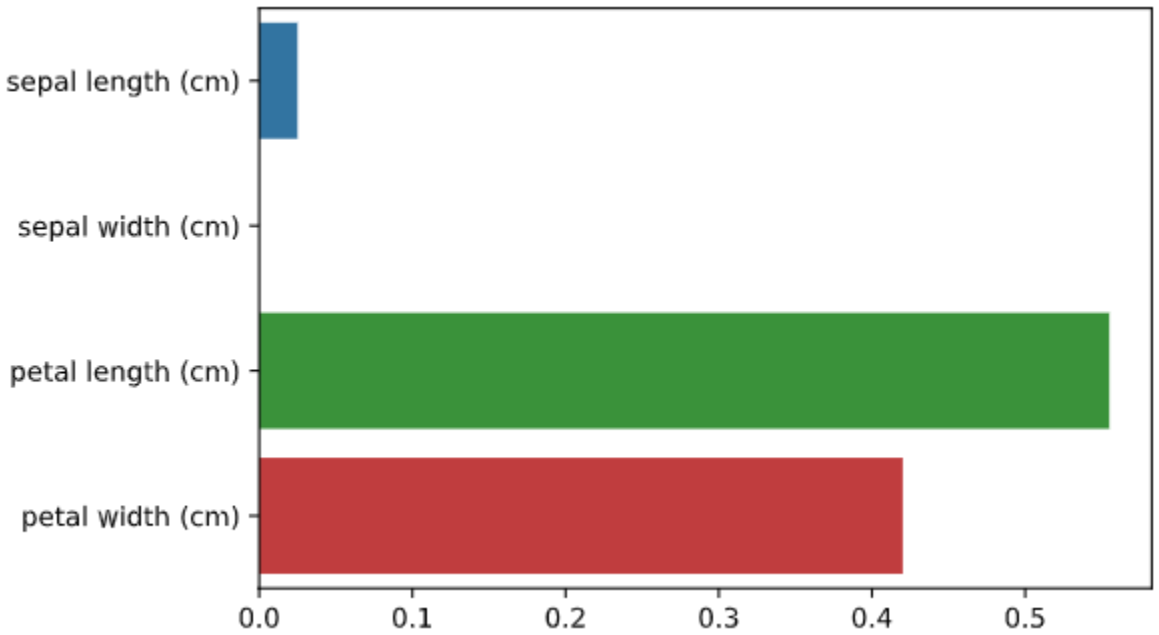
max_depth = 3



❖ 모델이 제공하는 정보

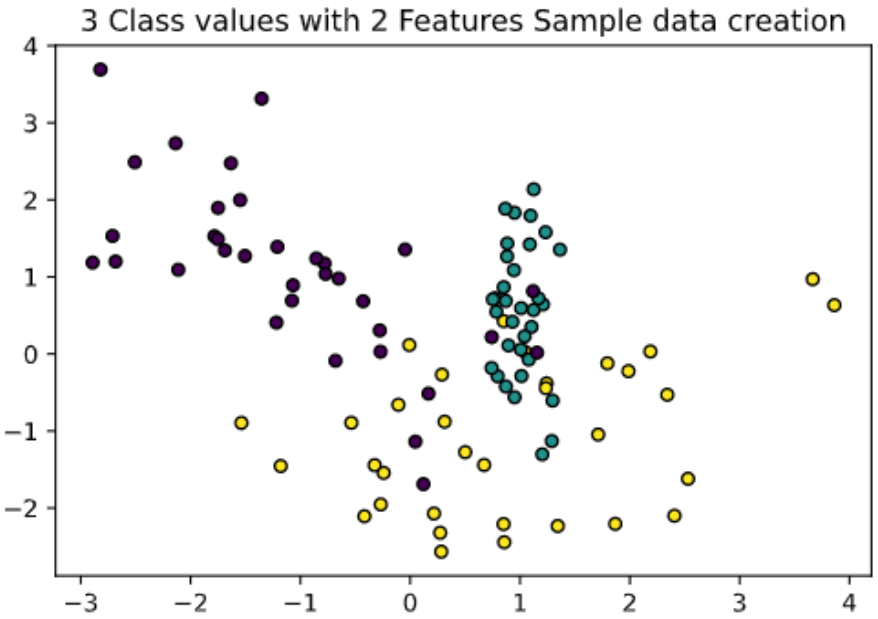
- `dt_clf.feature_importances_`

끝에 _ 가 있음

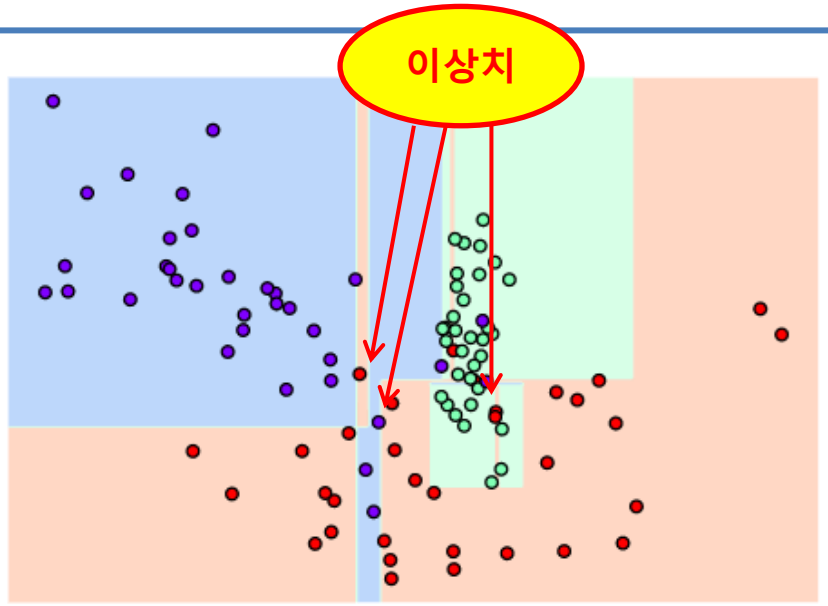


2. 결정 트리(Decision Tree)

❖ 과적합

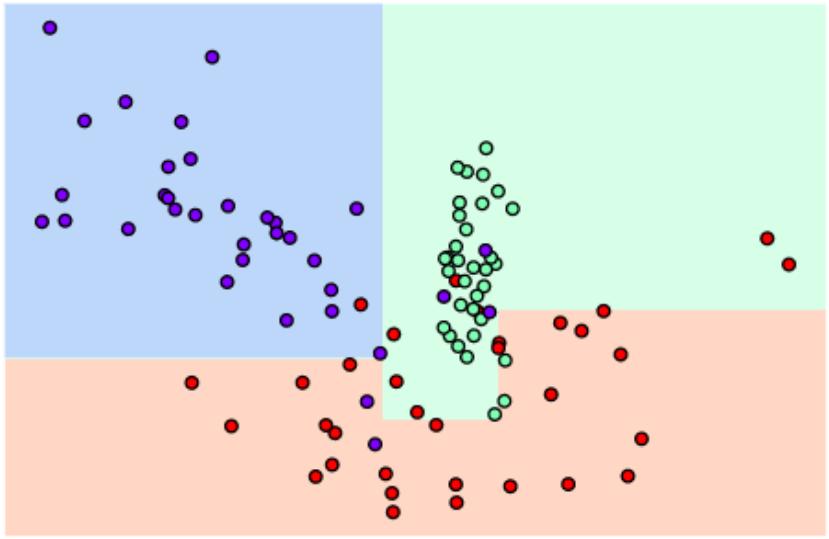


과적합



제약조건 없음

일반화

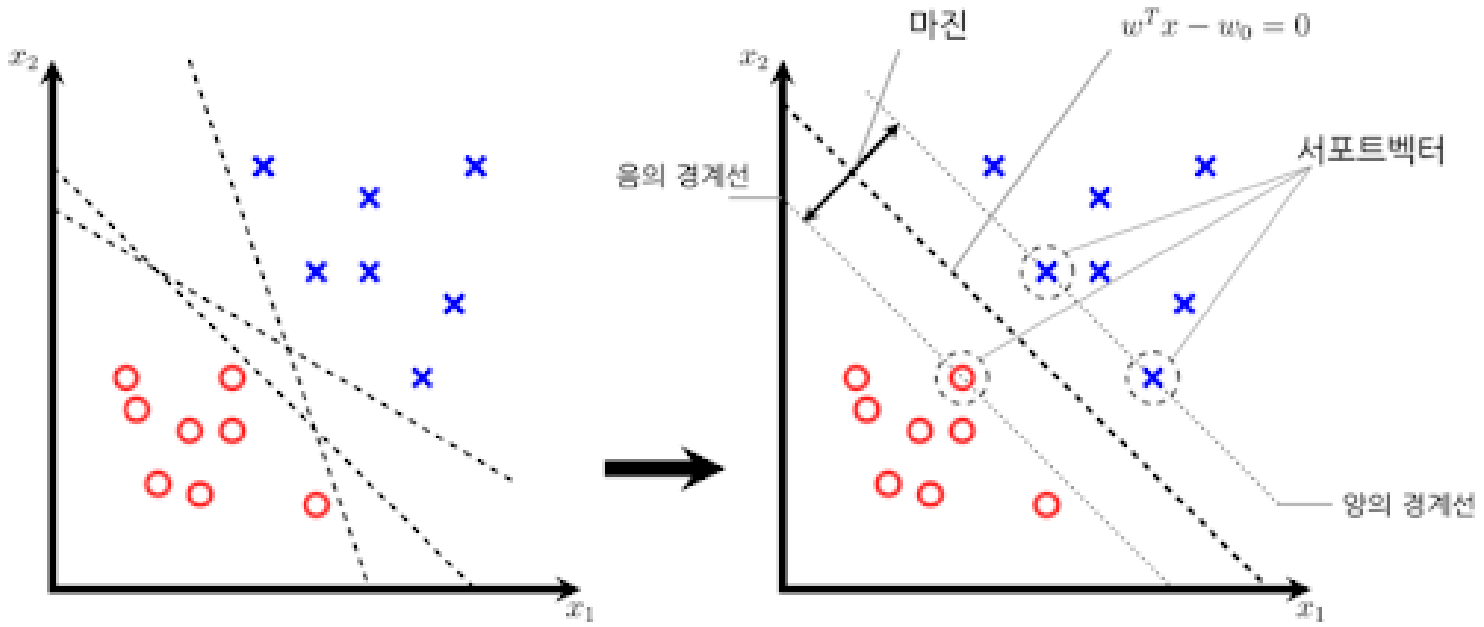


min_samples_leaf=6 제약조건

3. 서포트 벡터 머신(Support Vector Machine)

❖ 개요

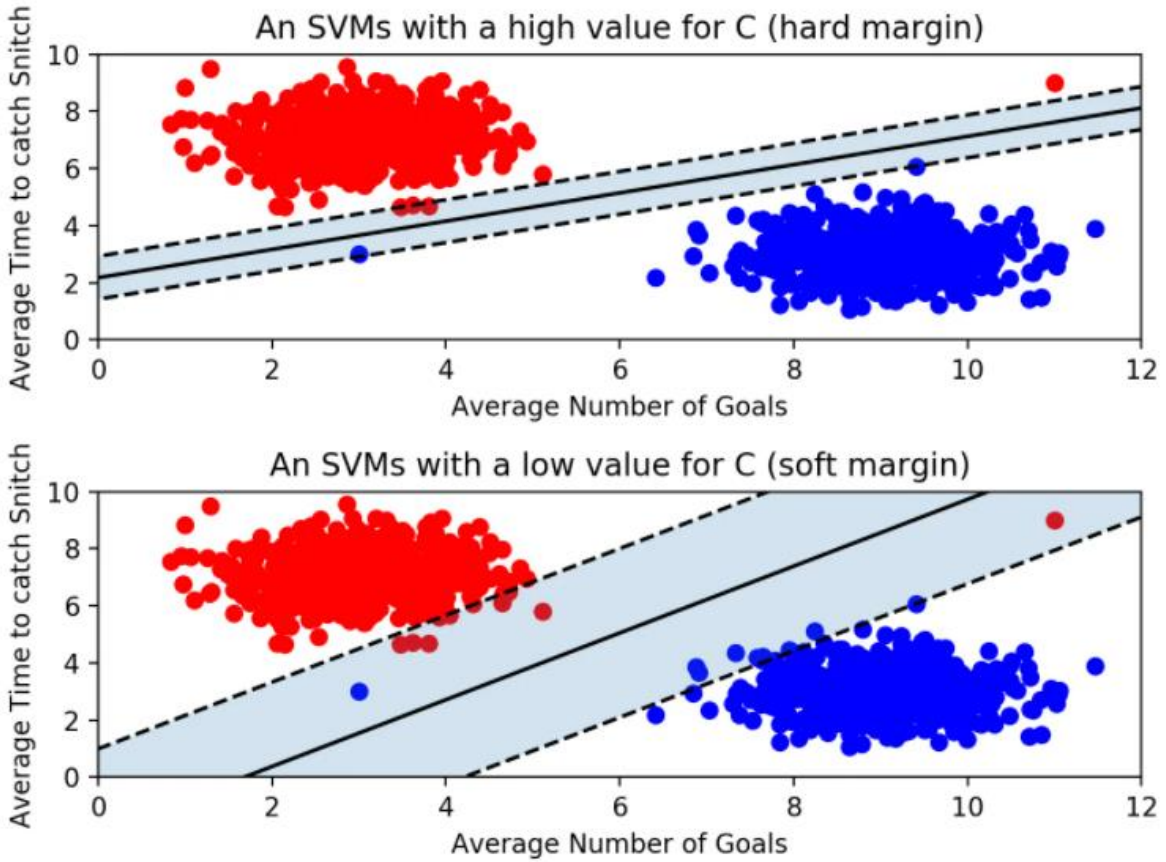
- 결정 경계(Decision Boundary), 즉 분류를 위한 기준 선을 정의하는 모델
- 마진을 최대화하는 분류선을 찾는 기법
- `from sklearn.svm import SVC`



3. 서포트 벡터 머신(Support Vector Machine)

❖ C 파라미터 - 이상치(Outlier) 허용 여부

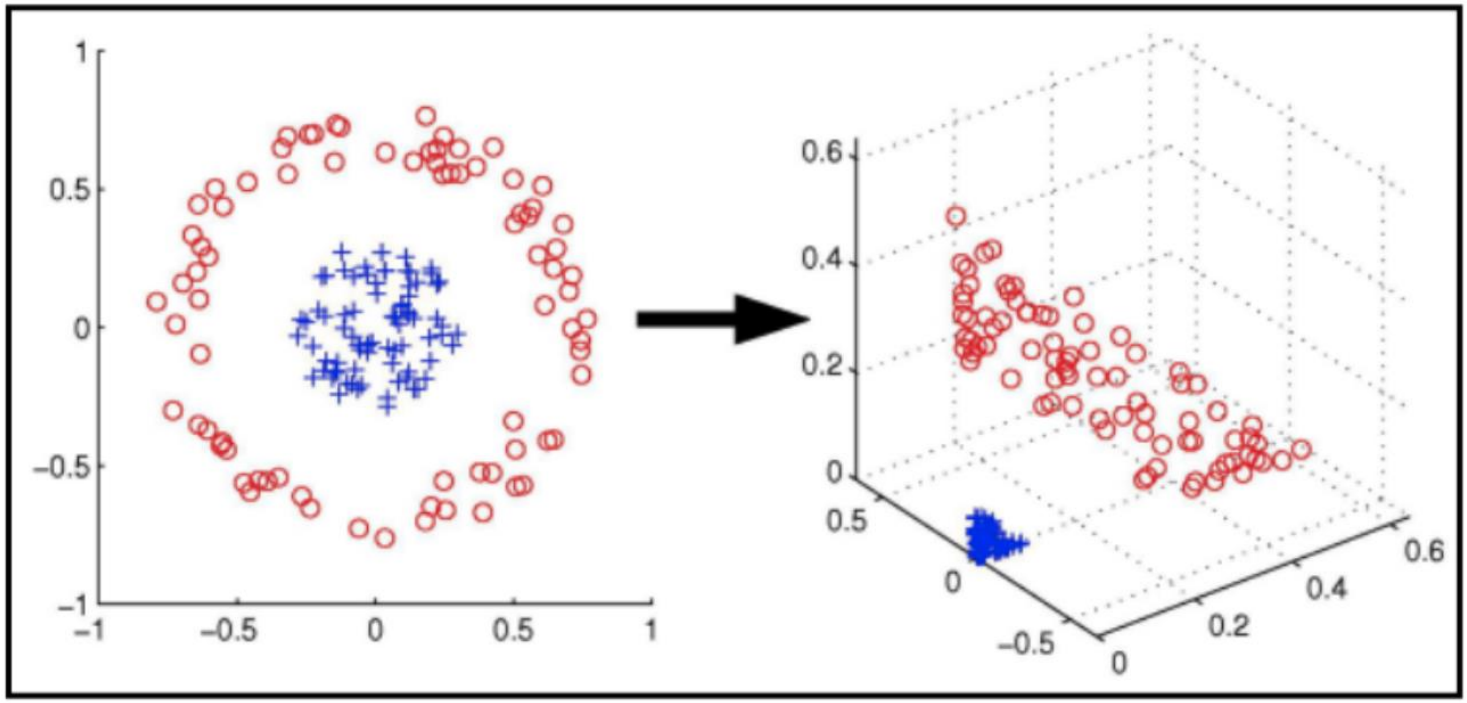
- 하드 마진 – 아웃라이어를 허용하지 않고 기준을 까다롭게 세운 모델
- 소프트 마진 – 아웃라이어들이 마진안에 어느정도 포함되도록 기준을 너그롭게 잡은 모델
- 파라미터 C로 조절
  - 클수록 하드 마진
  - 작으면 소프트 마진



❖ 커널(kernel) 파라미터 – 비선형 문제 해결

- linear – 선/면 등으로 분할
- poly – 차원을 확대하여 분할

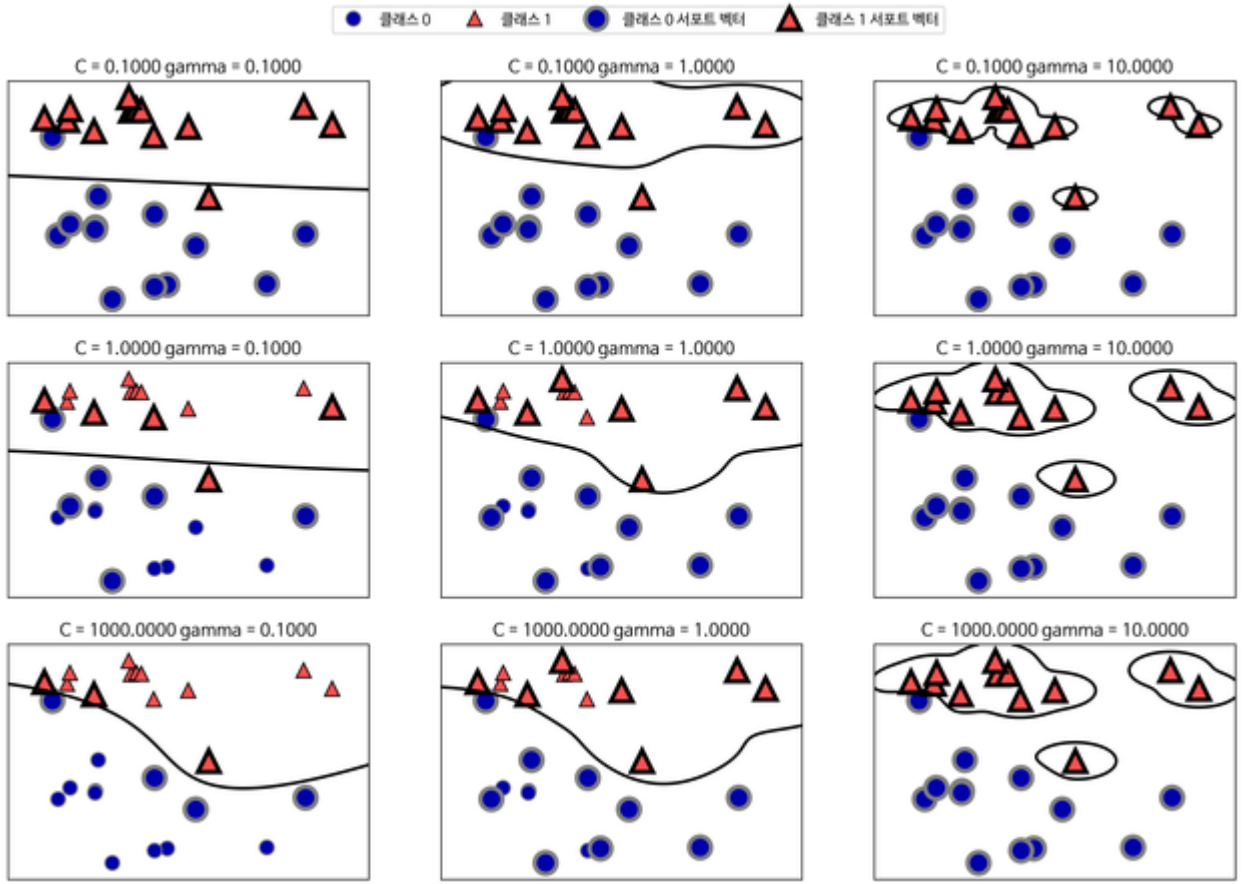
  예) $(x,\ y)$ -> $(\sqrt{2}xy,\ x^2,\ y^2)$



3. 서포트 벡터 머신(Support Vector Machine)

❖ 감마(gamma) 파라미터

- 가까이 있는 것에 얼마나 더 가중치를 부여할 것인지
- 커질수록 경계에 가까운 데이터의 중요도가 올라감



4. 앙상블 학습

❖ 개요

- 여러 개의 분류기를 생성하고 그 예측을 결합

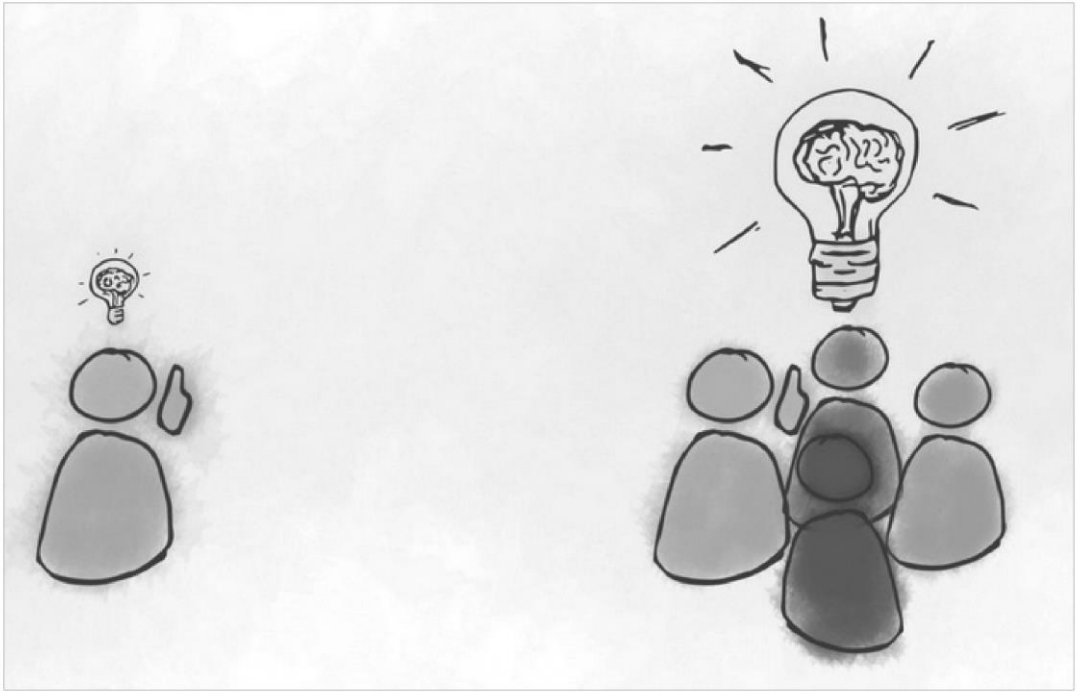
  → 정확한 최종 예측을 도출

- 이미지, 영상, 음성 분류

  → 딥러닝이 좋은 성능

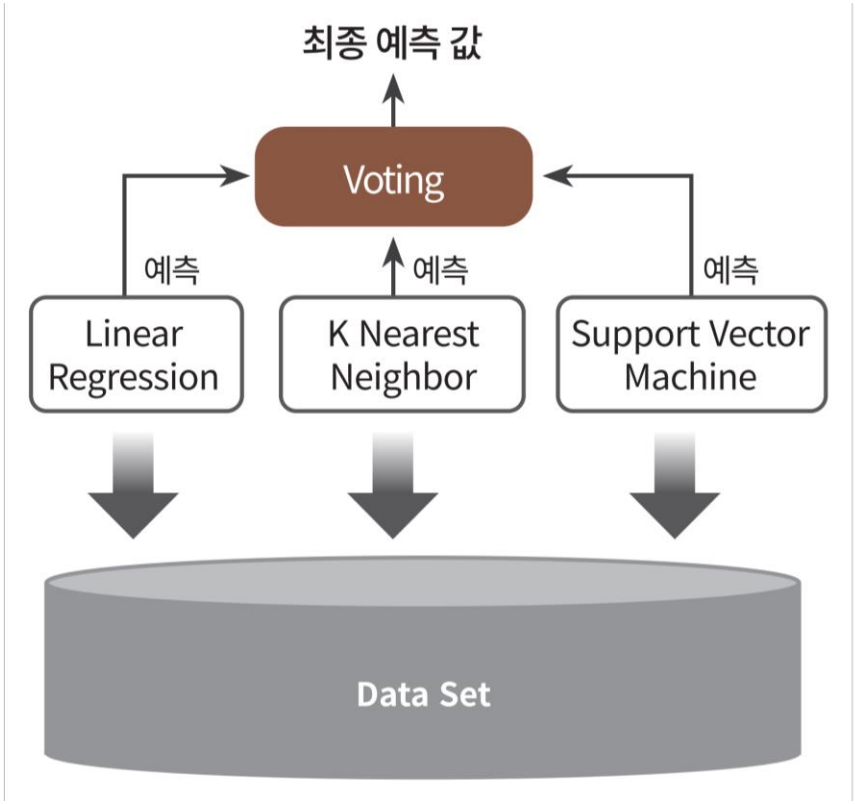
- 정형 데이터 분류

  → 앙상블이 뛰어난 성능



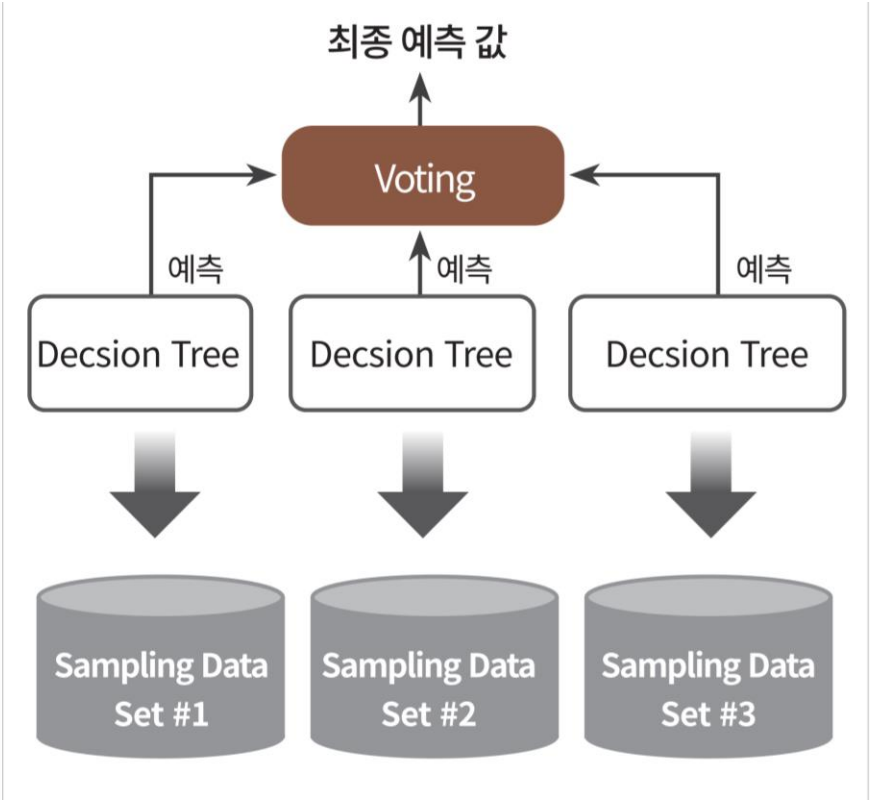
〈 집단 지성으로 어려운 문제도 쉽게 해결책을 찾을 수 있습니다 〉

❖ 유형



Voting 방식

서로 다른 알고리즘을 가진 분류기를 결합



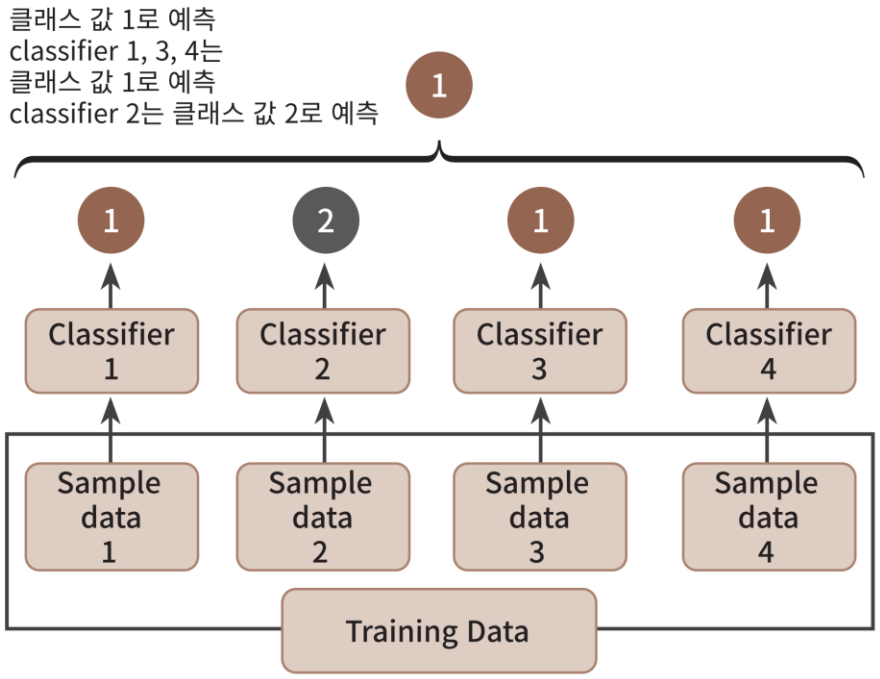
Bagging(Bootstrap Aggregating) 방식

같은 유형의 알고리즘 기반이지만, 데이터 샘플링을 서로 다르게 가져감.

4. 앙상블 학습

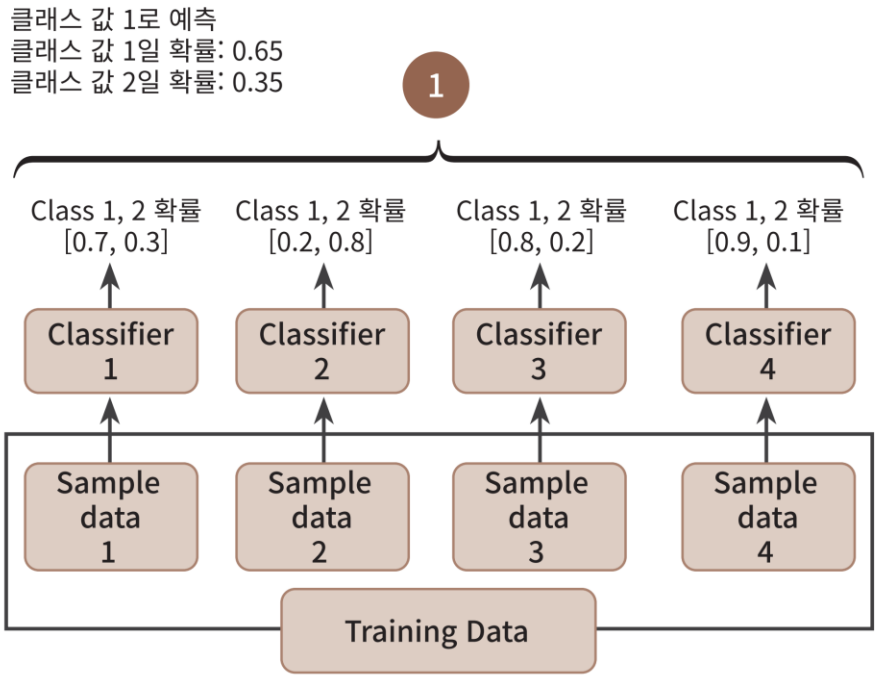
❖ 보팅 유형

Hard Voting은 다수의 classifier 간 다수결로 최종 class 결정



< 하드 보팅 >

Soft Voting은 다수의 classifier 들의 class 확률을 평균하여 결정

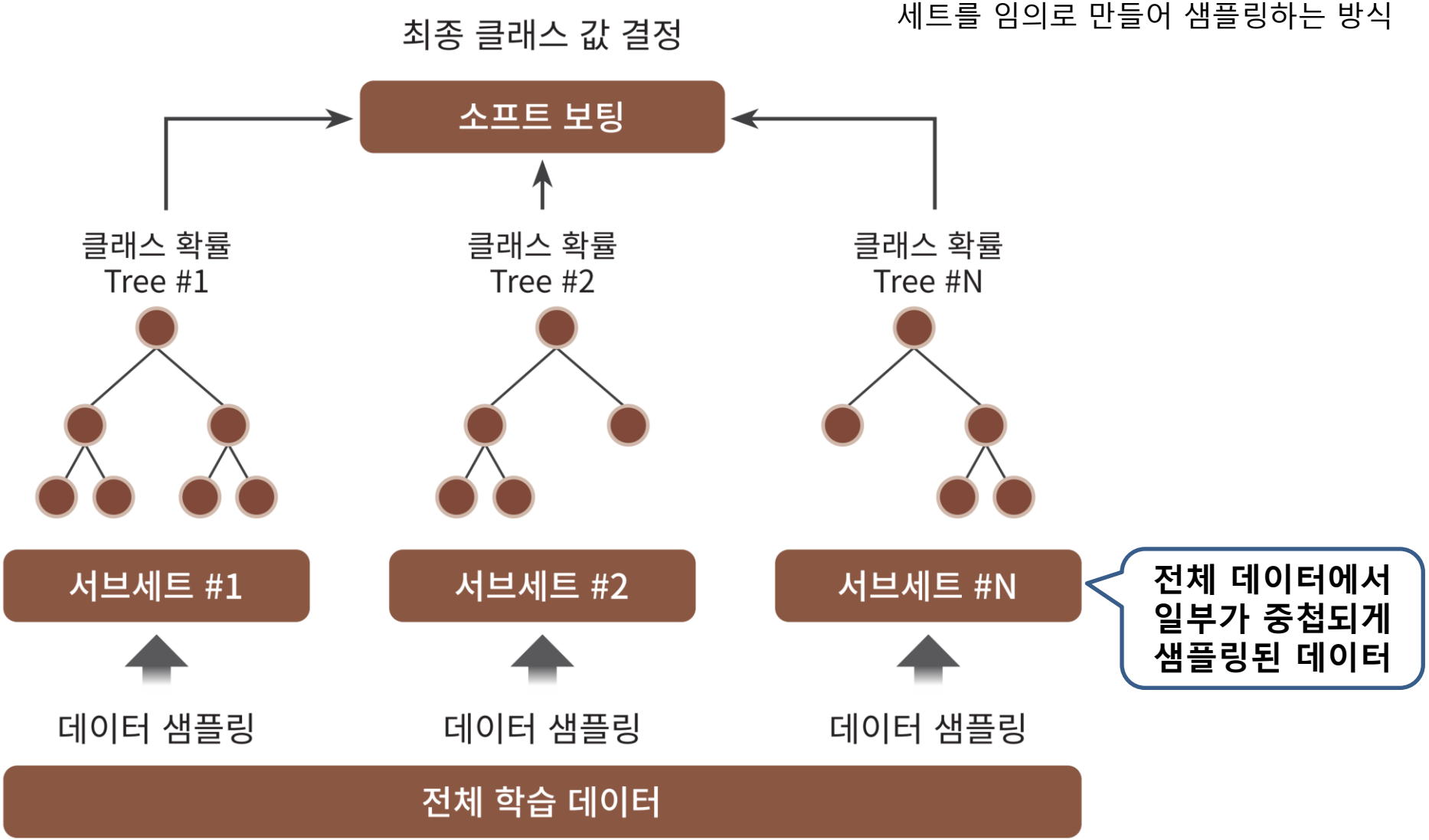


< 소프트 보팅 >

5. 랜덤 포레스트(Random Forest)

❖ 결정 트리를 이용한 배깅 방식

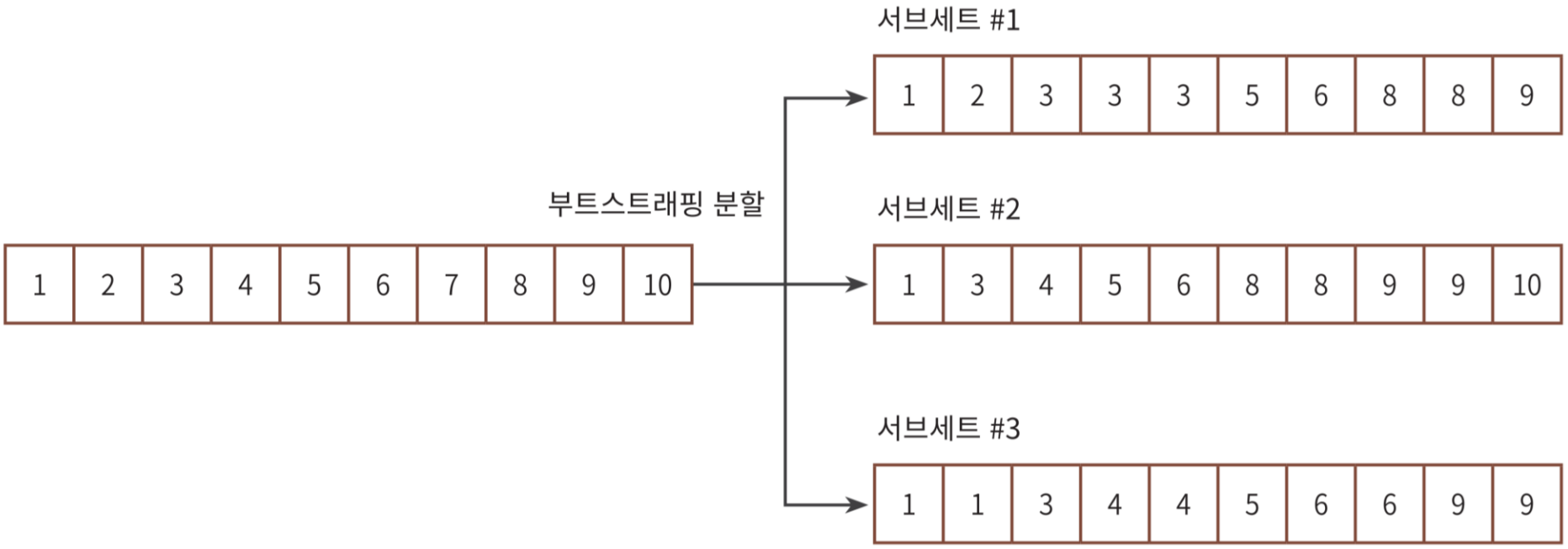
※ Bootstraping
통계학에서 여러 개의 작은 데이터 세트를 임의로 만들어 샘플링하는 방식



5. 랜덤 포레스트(Random Forest)

❖ 부트스트래핑 샘플링 방식

- 서브 세트의 데이터 건수는 전체 데이터와 동일
- 복원 방식으로 추출하여 개별 데이터는 중첩 가능



※ Bootstraping: 통계학에서 여러 개의 작은 데이터 세트를 임의로 만들어 샘플링하는 방식

5. 랜덤 포레스트(Random Forest)

❖ 하이퍼 파라미터

- n_estimators: 결정 트리의 개수 지정. 디폴트는 10
- max_features: 결정 트리에서 사용하는 max_features. 디폴트는 'auto'
- max_depth, min_samples_leaf 등 결정 트리에서 사용하는 파라미터

- `from sklearn.ensemble import RandomForestClassifier`

6. K-최근접 이웃(K-NN, K Nearest Neighbor)

- 훈련이 별도로 필요하지 않고, 훈련 데이터 저장이 전부 → Lazy Model
- `from sklearn.neighbors import KNeighborsClassifier`
- 거리 계산
  - 유클리드 거리: 수학에서 배웠던 거리
  - 맨해튼 거리: X축과 Y축을 따라간 거리