2주차 예습과제

범위	파머완 4.1-4.4장

1. 분류(Classification)의 개요

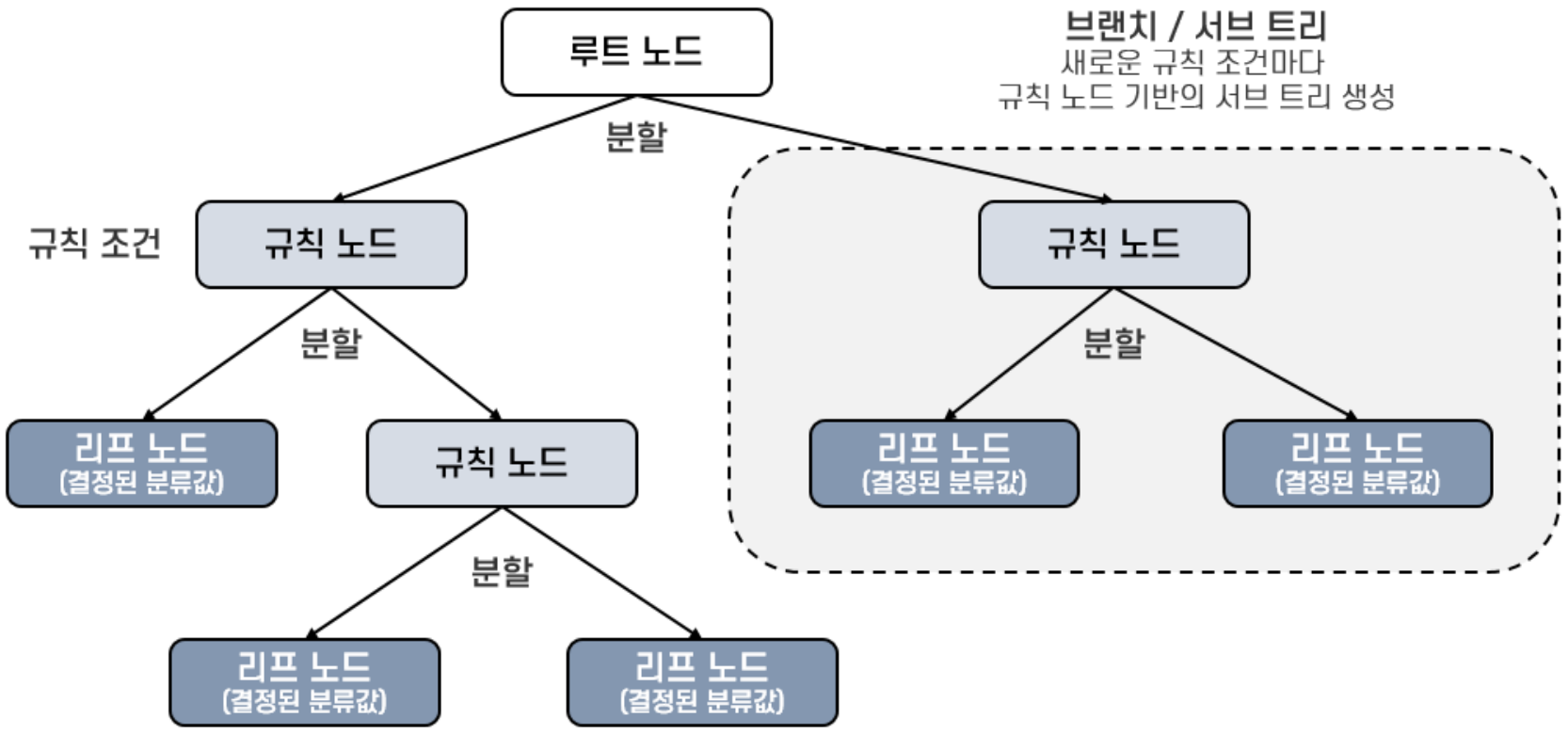
- **지도 학습** : 레이블(Label), 즉 명시적인 정답이 있는 데이터가 주어진 상태에서 학습하는 머신러닝 방식.
- **분류** : 지도 학습의 대표적인 유형. 학습 데이터로 주어진 데이터의 피처와 레이블 값(결정 값, 클래스 값)을 머신러닝 알고리즘으로 학습해 모델을 생성하고, 이렇게 생성된 모델에 새로운 데이터 값이 주어졌을 때 미지의 레이블 값을 예측하는 것.

2. 결정 트리(Decision Tree)

- 결정 트리
  - 장점
    - 매우 쉽고 직관적인 알고리즘.
    - 데이터의 스케일링이나 정규화 등의 사전 가공의 영향이 매우 적음.
  - 단점
    - 예측 성능을 향상시키기 위해 복잡한 규칙 구조를 가져야 하며, 이로 인한 과적합이 발생해 반대로 예측 성능이 저하될 수 있음.
    - 이를 극복하기 위해 트리의 크기를 사전에 제한하는 튜닝 필요.

- **결정 트리** : 데이터의 규칙을 학습을 통해 자동으로 찾아내 트리 기반의 분류 규칙을 만듦.
  - 일반적으로 if/else 방식의 스무 고개를 생각하면 된다. if/else를 자동으로 찾아내 예측.
- 데이터의 어떤 기준을 바탕으로 규칙을 만들어야 가장 효율적인 분류가 될 것인가가 알고리즘의 성능을 크게 좌우하게 된다.

결정 트리의 구조



- 규칙 노드라고 쓰여진 노드는 규칙 조건이 되며, 리프 노드라고 쓰여진 노드는 조건에 의해 결정된 클래스 값이다. 그리고 새로운 규칙 조건이 생길 때마다 서브 트리가 생성된다. 데이터 세트에 피처가 있고, 이 피처가 결합되어 규칙 조건이 생성될 때마다 규칙 노드가 만들어진다.

- 하지만 많은 규칙이 있다는 것은 분류를 결정하는 방식이 더욱 복잡해진다는 뜻 ⇒ 과적합으로 이어지기 쉽다.
- **즉, 트리의 깊이(depth)가 깊어질수록 결정 트리의 예측 성능이 저하될 가능성이 높다.**

- **가능한 한 적은 결정 노드로 높은 예측 정확도를 가지려면?**
  - 데이터를 분류할 때 최대한 많은 데이터가 해당 분류에 속할 수 있도록 결정 노드의 규칙이 정해져야 한다.
  - ➡️ 어떻게 트리를 분할할 것인가가 중요하다. 최대한 **균일한 데이터셋**을 구성할 수 있도록 분할하는 것이 필요하다.
  - 균일도가 낮고 혼잡도가 높은 데이터셋에서는 같은 조건에서 데이터를 판단하는 데 있어 더 많은 정보가 필요하기 때문.

균일한 데이터셋이란 무엇인가?

- 위의 그림을 통해 균일한 데이터셋을 확인할 수 있다.
- 가장 균일도가 높은 순서대로 나열하면 C > B > A 순서.
- **데이터 세트의 균일도는 데이터를 구분하는데 필요한 정보의 양에 영향을 미친다.**
- 데이터세트 C에서는 어떤 데이터를 선택하더라도 검은색 공이 선택되기 때문에 예측이 쉬우나, A에서는 상대적으로 혼잡도가 높고 균일도가 낮기 때문에 데이터를 판단하는 데 있어 더 많은 정보가 필요하게 되는 것이다.

- 따라서 결정 노드는 정보 균일도가 높은 데이터 세트를 먼저 선택하게끔 규칙 조건을 만든다.
- 즉, 정보 균일도가 데이터셋으로 쪼개질 수 있도록 조건을 찾아 서브 데이터셋을 만들고, 다시 이 서브 데이터셋에서 균일도가 높은 자식 데이터셋을 쪼개는 방식을 자식 트리로 내려가면서 반복하는 방식으로 데이터값을 예측하게 된다.

- **정보 균일도를 측정하는 대표적인 방법**
  - **정보 이득(Information Gain) 지수**
    - 엔트로피라는 개념을 기반으로 함.

> 💡 **엔트로피** : 주어진 데이터 집합의 혼잡도. 서로 다른 값이 섞여 있으면 엔트로피가 높고 같은 값이 섞여 있으면 엔트로피가 낮음.

    - 정보 이득 지수 : 1에서 엔트로피 지수를 뺀 값 (**1 - 엔트로피 지수**)
    - 결정 트리는 정보 이득이 높은 속성을 기준으로 분할함.
      - (정보이득 ⬆️ = 엔트로피 ⬇️ = 균일도 ⬆️)
  - **지니 계수**
    - 경제학에서 불평등 지수를 나타낼 때 사용하던 계수.
    - 0이 가장 평등하고 1로 갈수록 불평등함.
    - 지니 계수가 낮을수록 데이터 균일도가 높은 것으로 해석해 지니 계수가 낮은 속성을 기준으로 분할함. (지니 계수 ⬇️ = 균일도 ⬆️)
  - ➡️ **정보 이득이 높거나 지니 계수가 낮은 조건을 찾아서 분할.**

결정 트리 파라미터

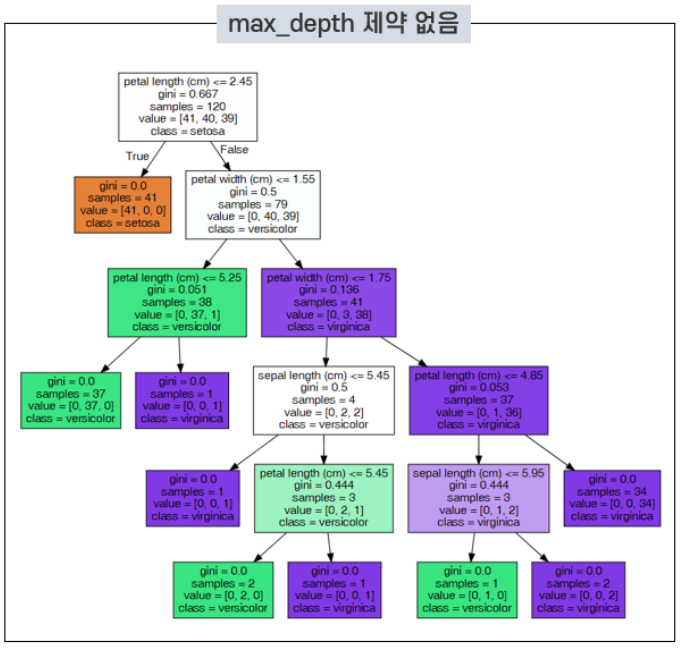
- 사이킷런은 결정 트리 알고리즘을 구현한 DecisionTreeClassifier(분류), DecisionTreeRegressor(회귀) 클래스를 제공함.
- 사이킷런의 결정 트리 구현은 CART(Classification And Regression Trees) 알고리즘 기반.
- `min_samples_split`
  - 노드를 분할하기 위한 최소한의 샘플 데이터 수로 과적합을 제어하는 데 사용됨.
  - 디폴트는 2이고 작게 설정할수록 분할되는 노드가 많아져서 과적합 가능성 증가.
- `min_samples_leaf`
  - 분할이 될 경우 왼쪽과 오른쪽의 브랜치 노드에서 가져야 할 최소한의 샘플 데이터 수.
  - 큰 값으로 설정될 경우 조건을 만족시키기 어려워 노드 분할을 상대적으로 덜 수행함.
  - min_samples_split과 유사하게 과적합 제어 용도, 그러나 비대칭적 데이터의 경우 특정 클래스의 데이터가 극도로 작을 수 있으므로 이 경우엔 작게 설정하는 것이 필요.
- `max_features`
  - 최적의 분할을 위해 고려할 최대 피처 개수. 디폴트는 None으로 데이터셋의 모든 피처를 사용해 분할 수행.
- `max_depth`
  - 트리의 최대 깊이를 규정. 디폴트는 None으로, 완벽하게 클래스 결정 값이 될 때까지 깊이를 계속 키우면서 분할하거나 노드가 가지는 데이터 개수가 min_sampes_split보다 작아질 때까지 계속 깊이를 증가시킴.
  - 깊이가 깊어지면 min_samples_split대로 최대 분할하여 과적합할 수 있으므로 적절한 제어가 필요.

결정 트리 모델의 시각화

- **Graphviz** 패키지를 이용해 시각화.
- 결정 트리는 규칙 생성 로직을 미리 제어하지 않으면 완벽하게 클래스 값을 구별해내기 위해 트리 노드를 계속해서 만들어간다.
- 이로 인해 매우 복잡한 규칙 트리가 만들어져 모델이 쉽게 과적합되는 문제점을 가진다.
- 따라서 결정 트리 알고리즘을 제어하는 대부분의 하이퍼 파라미터는 복잡한 트리가 생성되는 것을 막기 위한 용도이다.

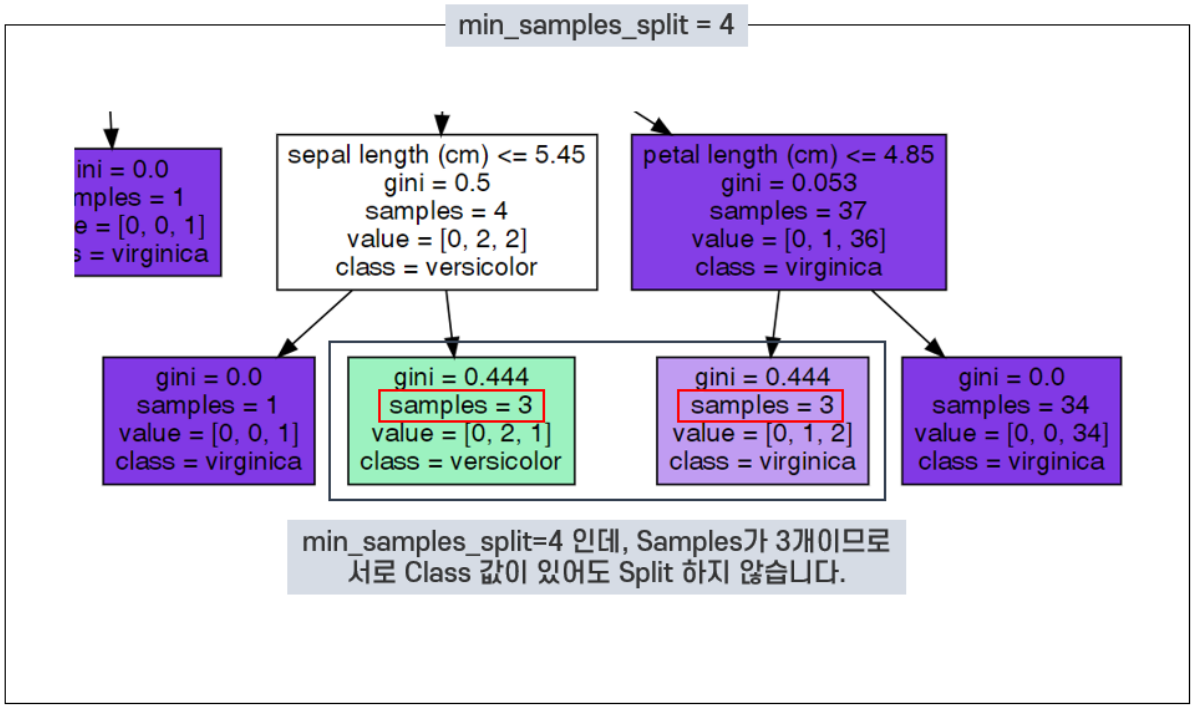
✅ max_depth 하이퍼 파라미터 변경 시

결정 트리의 최대 트리 깊이 제어



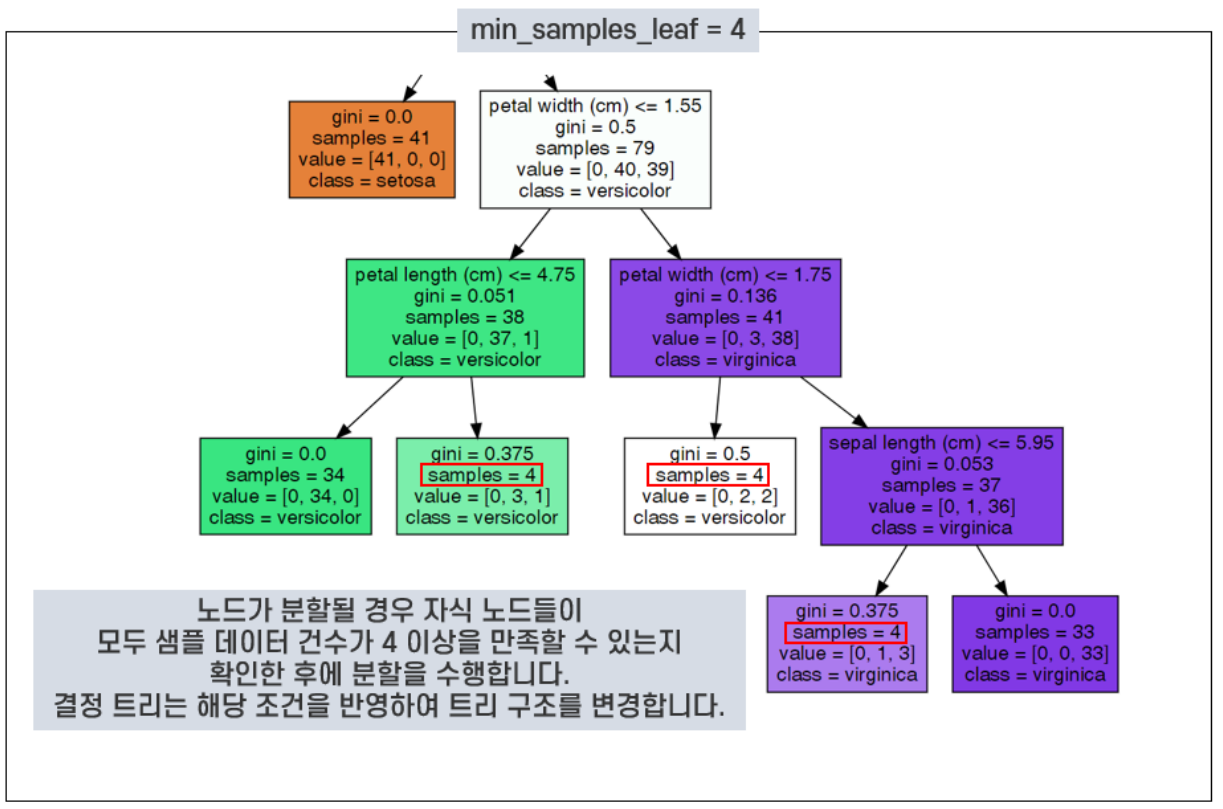
✅ min_samples_split 하이퍼 파라미터 변경 시

자식 규칙 노드를 분할해 만들기 위한 최소한의 샘플 데이터 개수



✅ min_samples_leaf 하이퍼 파라미터 변경 시

왼쪽과 오른쪽 자식 노드 각각이 가지게 될 최소 데이터 건수 지정

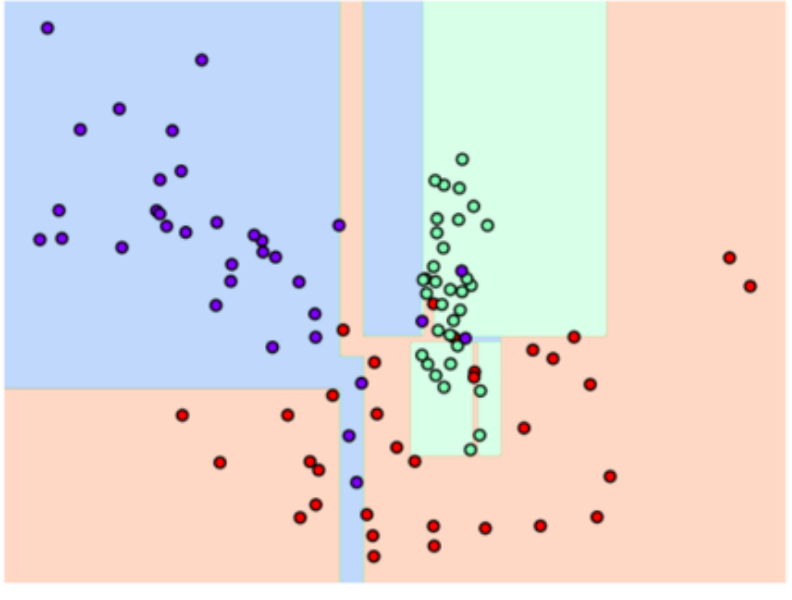


- max_depth를 조정하면 트리의 깊이가 줄어들며 더 간단한 결정 트리가 되고, min_samples_split과 min_samples_leaf를 조정하면 분할 조건이 어려워져 자연스럽게 리프 노드가 되어 더 간단한 결정 트리가 된다.
- DecisionTreeClassifier 객체의 `feature_importances_` 속성을 통해 피처의 중요한 역할 지표를 확인할 수 있음.

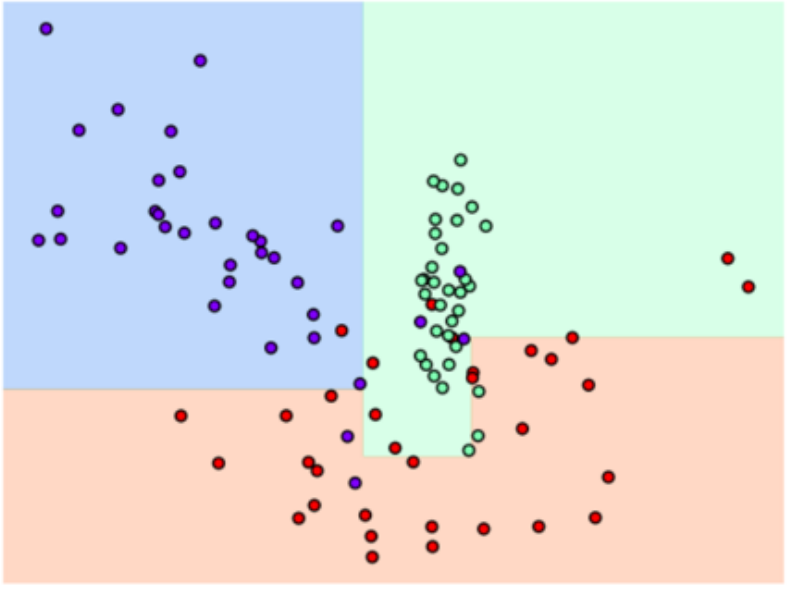
결정 트리 과적합(Overfitting)

- `make_classification()` : 분류를 위한 테스트용 데이터를 쉽게 만들 수 있게 해줌.

<엄격한 분할 기준으로 인해 복잡한 모델>



<min_samples_leaf=6으로 트리 생성 조건을 제약한 모델>



시각화를 통해 보는 결정 트리 과적합 문제. 첫 번째 모델은 일부 이상치(Outlier) 데이터까지 분류하기 위해 분할이 자주 일어나서 결정 기준 경계가 매우 많아짐. 반면, 두 번째 모델은 이상치에 크게 반응하지 않으면서 좀 더 일반화된 분류 규칙에 따라 분류됨.

- 다양한 테스트 데이터셋을 기반으로 한 결정 트리 모델의 예측 성능은 min_samples_leaf=6으로 트리 생성 조건을 제약한 모델이 더 뛰어날 가능성이 높다.
- 테스트 데이터셋은 학습 데이터셋과 다른 데이터셋인데 학습 데이터에만 지나치게 최적화된 분류 기준은 오히려 테스트 데이터셋에서 정확도를 떨어뜨릴 수 있기 때문.
- 복잡한 모델은 학습 데이터셋과 약간만 다른 형태의 데이터셋을 예측하면 정확도가 떨어진다.

3. 앙상블 학습

- **앙상블 학습(Ensemble Learning) 분류**
  - **여러 개의 분류기(Classifier)를 생성하고 그 예측을 결합함으로써 보다 정확한 최종 예측을 도출하는 기법**

> 목표 : 다양한 분류기의 예측 결과를 결합 → 단일 분류기보다 신뢰성이 높은 예측값을 얻는 것.

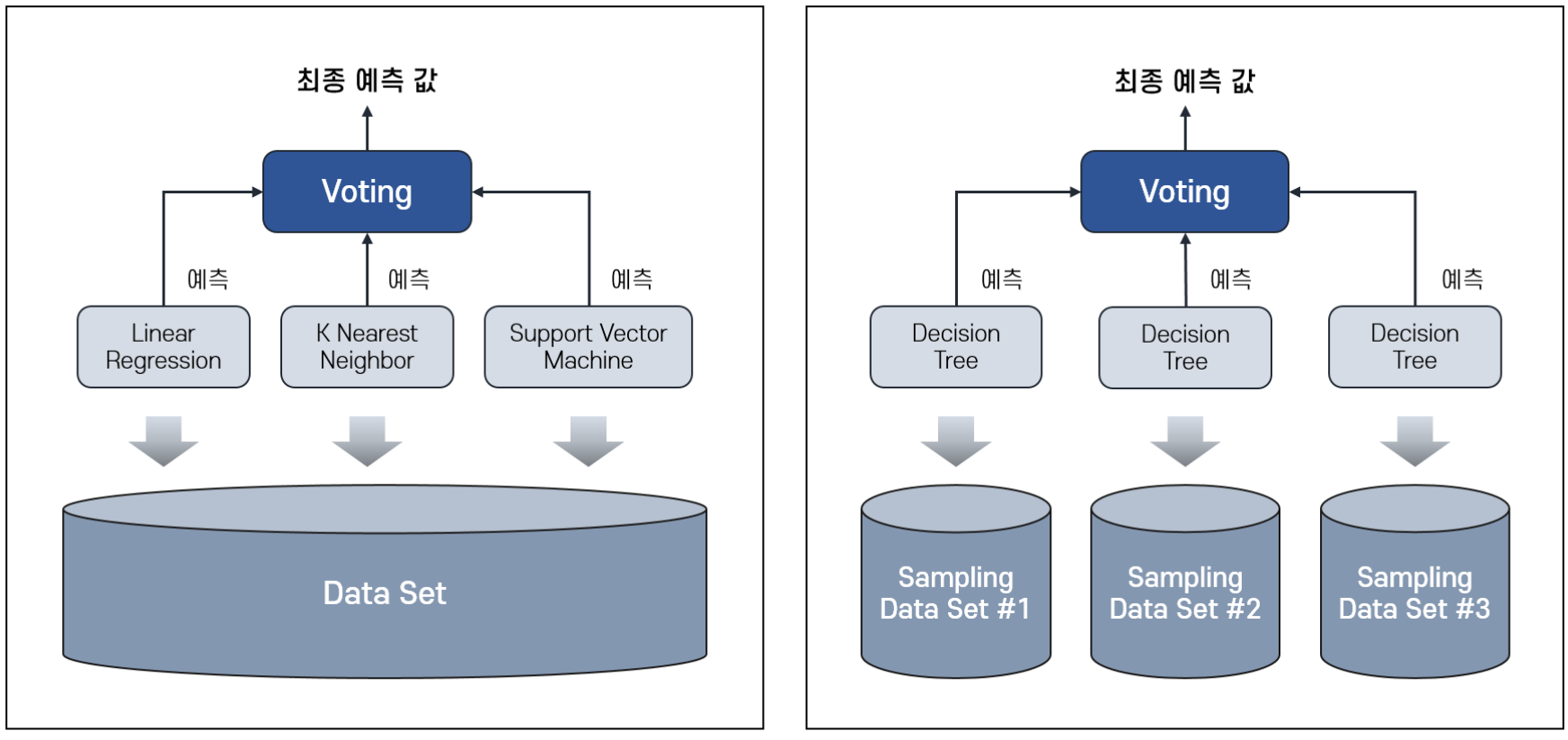
- 비정형 데이터(이미지, 영상, 음성 등)의 분류는 딥러닝이 뛰어난 성능을 보이고 있지만, 대부분의 정형 데이터 분류 시에는 앙상블이 뛰어난 성능을 보인다.
- 대표적인 앙상블 알고리즘
  - 랜덤 포레스트, 그래디언트 부스팅
  - 부스팅 계열의 앙상블 알고리즘에서 확장된 XGboost, LightGBM 등

앙상블 학습의 유형

- 보팅(Voting), 배깅(Bagging), 부스팅(Boosting) 이 외 스태깅 등
- `보팅`과 `배깅`은 여러 개의 분류기가 투표를 통해 최종 예측 결과를 결정하는 방식

> **보팅과 배깅의 차이점**

보팅	서로 다른 알고리즘을 가진 분류기를 결합하는 것
배깅	각각의 분류기가 모두 같은 유형의 알고리즘 기반이지만, 데이터 샘플링을 서로 다르게 가져가면서 학습을 수행해 보팅을 수행하는 것 (대표 알고리즘 : 랜덤 포레스트 알고리즘)



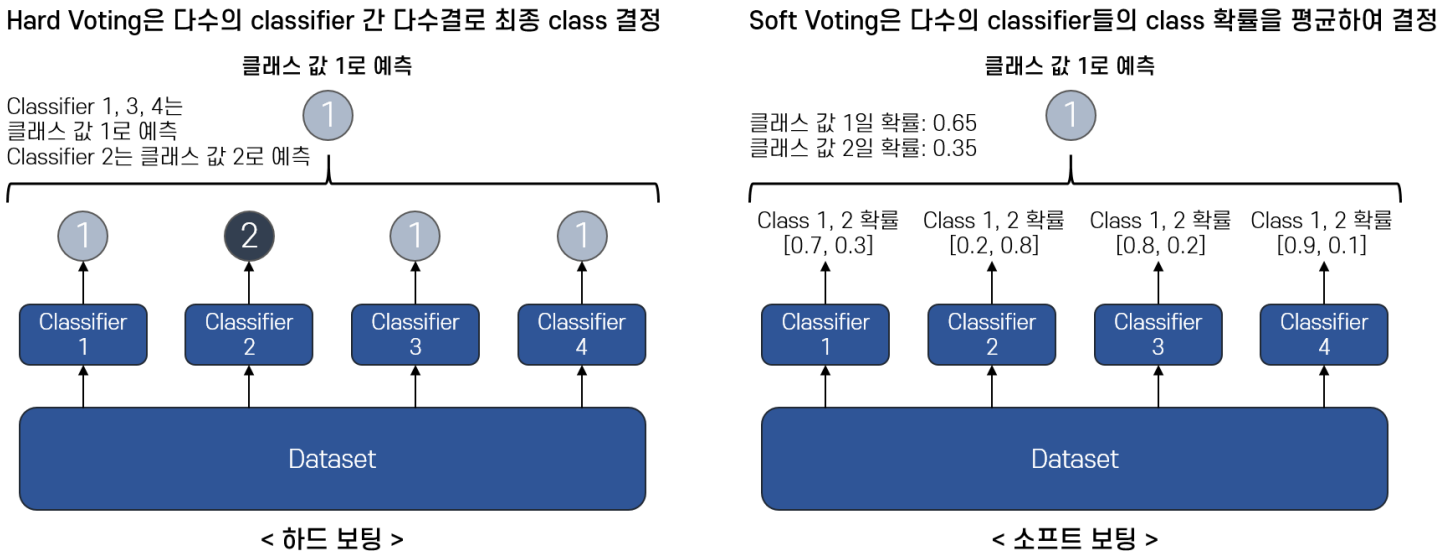
Voting 방식 Bagging 방식

- 배깅 방식에서 개별 분류기에 할당된 학습 데이터는 원본 학습 데이터를 샘플링해 추출하는데, 이렇게 개별 Classifier에게 데이터를 샘플링해서 추출하는 방식을 **부트스트래핑(Bootstrapping)** 분할 방식이라고 부름. (교차 검증과 다르게 중첩을 허용함)

- 부스팅 : 여러 개의 분류기가 순차적으로 학습을 수행하되, 앞에서 학습한 분류기가 예측이 틀린 데이터에 대해서는 올바르게 예측할 수 있도록 다음 분류기에 가중치를 부여하며 학습과 예측을 진행하는 방식.
  - 예측 성능이 뛰어나 앙상블 학습을 주도하고 있음. 대표적으로 그래디언트 부스트, XGBoost, LightGBM 이 있음.
- 스태킹 : 여러 가지 다른 모델의 예측 결괏값을 다시 학습 데이터로 만들어서 다른 모델(메타 모델)로 재학습시켜 결과를 예측하는 방법.

보팅 유형 - 하드 보팅(Hard Voting)과 소프트 보팅(Soft Voting)

- **하드 보팅** : 일반적인 다수결 원칙과 비슷함. 예측한 결괏값들 중 다수의 분류기가 결정한 예측값을 최종 보팅 결괏값으로 선정하는 것.
- **소프트 보팅** : 분류기들의 레이블 값 결정 확률을 모두 더하고 이를 평균해서 이들 중 확률이 가장 높은 레이블 값을 최종 보팅 결괏값으로 선정하는 것.
- 일반적으로 하드 보팅보다는 소프트 보팅이 예측 성능이 좋아서 더 많이 사용됨.



- 보팅과 스태킹 등은 서로 다른 알고리즘을 기반으로 하고 있지만, 배깅과 부스팅은 대부분 결정 트리 알고리즘을 기반으로 함.

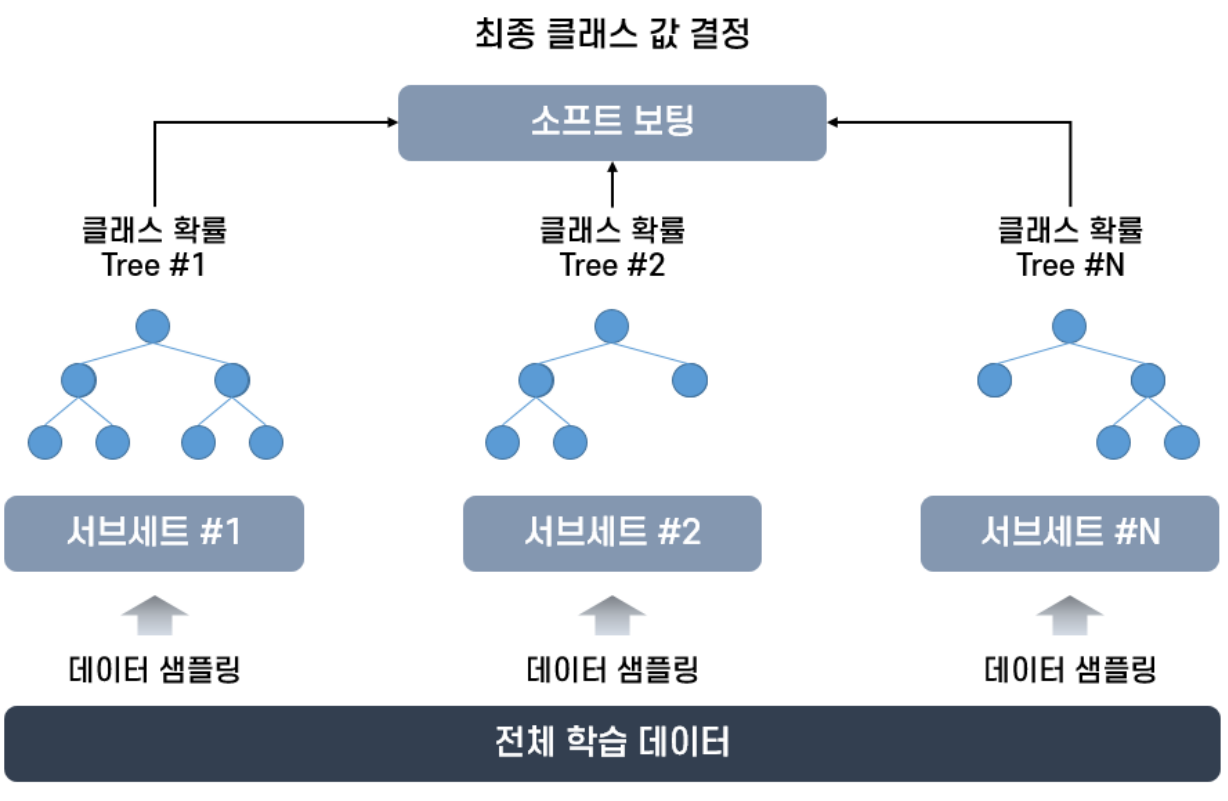
- 결정 트리 알고리즘은 정확한 예측을 위해 학습 데이터의 예외 상황에 집착한 나머지 과적합이 발생할 수 있는데 앙상블 학습을 통해 이 단점을 수십~수천 개의 매우 많은 분류기를 결합해 다양한 학습을 시키며 극복하고 있음.
    - 결정 트리 알고리즘의 장점은 그대로 취하고 단점은 보완하며 편향-분산 트레이드오프의 효과를 극대화할 수 있다.

4. 랜덤 포레스트

랜덤 포레스트의 개요 및 실습

- 배깅의 대표 알고리즘 : **랜덤 포레스트**
    - 앙상블 알고리즘 중 비교적 빠른 수행 속도를 가지며 여러 영역에서 높은 예측 성능을 보임.
    - 기반 알고리즘 : **결정 트리** → 결정 트리의 쉽고 직관적인 장점을 그대로 가짐.
        - 일반적으로 앙상블의 기본 알고리즘은 결정 트리를 사용함.

랜덤 포레스트	여러 개의 결정 트리 분류기가 전체 데이터에서 배깅 방식으로 각자의 데이터를 샘플링해 개별적으로 학습을 수행한 뒤 최종적으로 모든 분류기가 보팅을 통해 예측 결정을 하게 됨.



랜덤 포레스트

- 랜덤 포레스트는 개별 분류기의 기반 알고리즘은 결정 트리이지만 개별 트리가 학습하는 데이터셋은 전체 데이터에서 일부가 중첩되게 샘플링된 데이터셋임.
- 이렇게 여러 개의 데이터셋을 중첩되게 분리하는 것을 **부트스트래핑(bootstrapping) 분할 방식**이라고 한다.
- 위의 사진에서 랜덤 포레스트의 서브세트(subset) 데이터는 부트스트래핑으로 데이터가 임의로 만들어짐. → 만들어진 개별 데이터셋에 결정 트리 분류기를 각각 적용하는 것이 랜덤 포레스트.

부트스트래핑(Bootstrapping) 분할 방식

랜덤 포레스트 하이퍼 파라미터 및 튜닝

- `n_estimators` : 랜덤 포레스트에서 결정 트리의 개수를 지정합니다. 디폴트는 10개입니다. 많이 설정할수록 좋은 성능을 기대할 수 있지만 계속 증가시킨다고 성능이 무조건 향상되는 것은 아닙니다. 또한 늘릴수록 학습 수행 시간이 오래 걸리는 것도 감안해야 합니다.
- `max_features` : 결정 트리에 사용된 max_features 파라미터와 같습니다. 하지만 RandomForestClassifier의 기본 max_features는 'None'이 아니라 'auto' 즉 'sqrt'와 같습니다. 따라서 랜덤 포레스트의 트리를 분할하는 피처를 참조할 때 전체 피처가 아니라 sqrt(전체 피처 개수)만큼 참조합니다.(전체 피처가 16개라면 분할을 위해 4개 참조)
- `max_depth` 나 `min_samples_leat` , `min_samples_split` 와 같이 결정 트리에서 과적합을 개선하기 위해 사용되는 파라미터가 랜덤 포레스트에도 똑같이 적용될 수 있습니다.

- 랜덤 포레스트는 CPU 병렬 처리도 효과적으로 수행되어 빠른 학습이 가능하기 때문에 그래디언트 부스팅보다 예측 성능이 약간 떨어지더라도 랜덤 포레스트로 일단 기반 모델을 먼저 구축하는 경우가 많음.
- `feature_importances_` 속성을 이용해 알고리즘이 선택한 피처의 중요도를 알 수 있음.