[CH04] 분류

01 분류(Classification)의 개요

분류: 학습 데이터의 피처, 레이블 값 학습 → 모델 생성 → 새로운 데이터 레이블 값 예측

- 분류 ML 알고리즘
    - 나이브 베이즈(Naïve Bayes): 통계와 생성 모델 기반
    - 로지스틱 회귀(Logistic Regression): 독립변수와 종속변수 선형 관계성에 기반
    - 결정 트리(Decision Tree): 데이터 균일도에 따른 규칙 기반
    - 서포트 벡터 머신(Support Vector Machine): 개별 클래스 간의 최대 분류 마진을 효과적으로 탐색
    - 최소 근접(Nearest Neighbor) 알고리즘: 근접 거리 기준
    - 신경망(Neural Network): 심층 연결 기반
    - **앙상블(Ensemble)**: 서로 동일한(or 다른) ML 알고리즘 결합

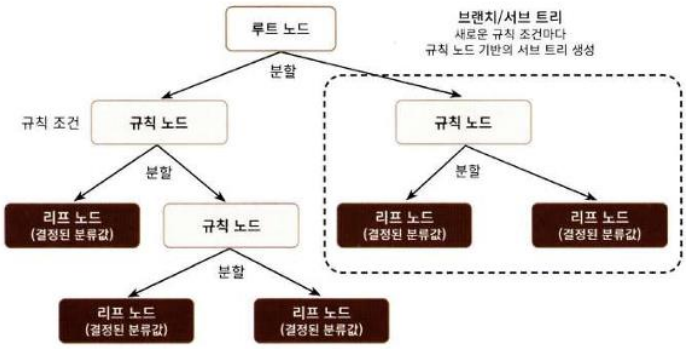
      → 배깅(Bagging) – 랜덤 포레스트(Random Forest): 뛰어난 예측 성능, 빠른 수행 시간, 유연성

      → 부스팅(Boosting) – 그래디언트 부스팅(Gradient Boosting): 정형 데이터의 분류 영역에서 가장 활용도가 높음

02 결정 트리

1) 결정 트리

- 데이터에 있는 규칙의 학습을 통해 자동으로 찾아내 트리(Tree) 기반의 분류 규칙 만듦
- 어떤 피처로 규칙을 만들어야 효율적인 분류가 될 것인지가 알고리즘의 성능 크게 좌우
- 구조
    - 규칙 노드(Decision Node): 규칙 조건
    - 리프 노드(Leaf Node): 결정된 클래스 값
    - 서브 트리(Sub Tree): 새로운 규칙 조건마다 생성



- 분할(Split)
    - 트리 깊이(depth)↑ ⇒ 예측 성능저하 가능성↑
    - ∴ 최대한 많은 데이터 세트가 해당 분류에 속할 수 있도록 규칙을 정해야 함. (결정 노드↓, 깊이↓ 위해)
    - 즉, 균일도 높은 데이터 세트를 구성하도록 분할
    - 균일도↑ ⇒ 데이터 구분하는데 필요한 정보의 양↓
    - 정보 균일도 높은 데이터 세트 먼저 선택하는 규칙 조건 만드는 분할 반복
- 균일도 측정
    - 정보 이득(Information Gain) 지수
        - 엔트로피(주어진 데이터 집합의 혼잡도) 개념 기반.
        - 서로 ≠ 값이 섞여 있으면 엔트로피 ↑, 서로 = 값이 섞여 있으면 엔트로피 ↓
        - 1 - 엔트로피 지수
        - 정보 이득 지수↑ 속성을 기준으로 분할
    - 지니 계수
        - 0(평등) <-> 1(불평등)
        - 지니 계수↓ 속성을 기준으로 분할
- 사이킷런의 결정 트리 알고리즘(DecisionTreeClassifier)
    - 지니계수 이용해 데이터 세트 분할
    - 정보 이득↑ or 지니 계수↓ 조건 찾아서 자식 트리 노드에 반복적 분할 → 데이터가 모두 특정 분류에 속할 시 분할을 멈추고 분류를 결정

2) 결정 트리 모델 특징

- 장점
    - 균일도 기반으로 알고리즘이 쉽고 직관적
    - 데이터 전처리 작업 필요 없음
- 단점
    - 과적합으로 정확도 떨어짐
    - ∴ 트리 크기를 사전에 제한하는 튜닝이 필요

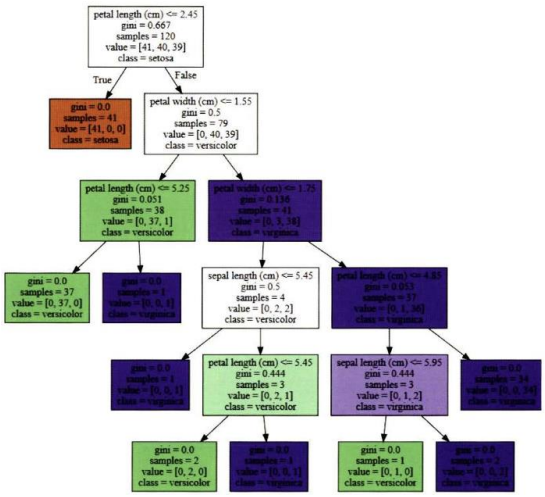
3) 결정 트리 파라미터

- DecisionTreeClassifier: 사이킷런의 분류를 위한 클래스
- DecisionTreeRegressor: 사이킷런의 회귀를 위한 클래스
- CART(Classification And Regression Trees): 분류뿐만 아니라 회귀에서도 사용 가능한 트리 알고리즘
- 사이킷런의 결정 트리 구현은 CART 알고리즘 기반

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.
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

4) 결정 트리 모델의 시각화

- Grapviz 패키지 사용
- export_graphviz( ): 위 패키지와 쉽게 인터페이스 할 수 있는 사이킷런 API
    - 파라미터
      - 학습 완료된 Estimator
      - 피처 이름 리스트
      - 레이블 이름 리스트
- 붓꽃 데이터 세트 예시



    - 리프(leaf) 노드
      - 더 이상 자식 노드가 없는 최종 클래스(레이블) 값이 결정되는 노드
      - 조건: 오직 하나의 클래스 값으로 최종 데이터가 구성되거나 리프 노드가 될 수 있는 하이퍼 파라미터 조건을 충족하면 됨
    - 브랜치(branch) 노드
      - 자식 노드가 있는 노드
      - 자식 노드를 만들기 위한 분할 규칙 조건을 가지고 있음
    - 노드 내에 기술된 지표의 의미
      - petal length(cm) <= 2.45와 같이 피처의 조건이 있는 것은 자식 노드를 만들기 위한 규칙 조건입니다. 이 조건이 없으면 리프 노드입니다.
      - gini는 다음의 value=[ ]로 주어진 데이터 분포에서의 지니 계수입니다.
      - samples는 현 규칙에 해당하는 데이터 건수입니다.
      - value = [ ]는 클래스 값 기반의 데이터 건수입니다. 붓꽃 데이터 세트는 클래스 값으로 0, 1, 2를 가지고 있으며, 0 : Setosa, 1: Versicolor, 2: Virginica 품종을 가리킵니다. 만일 Value = [41, 40, 39]라면 클래스 값의 순서로 Setosa 41개, Vesicolor 40개, Virginica 39개로 데이터가 구성돼 있다는 의미입니다.

    - ndarray 형태로 값 반환, 피처 순서대로 값 할당

- **피처 중요도**
  feature_importances_
    - DecisionTreeClassifier 객체의 속성
    - ndarray 형태로 값 반환, 피처 순서대로 값 할당
    - 피처가 트리 분할 시 정보 이득이나 지니 계수를 얼마나 효율적으로 잘 개선시켰는지를 정규화된 값으로 표현
    - 값↑, 중요도↑

5) 결정 트리 과적합(Overfitting)

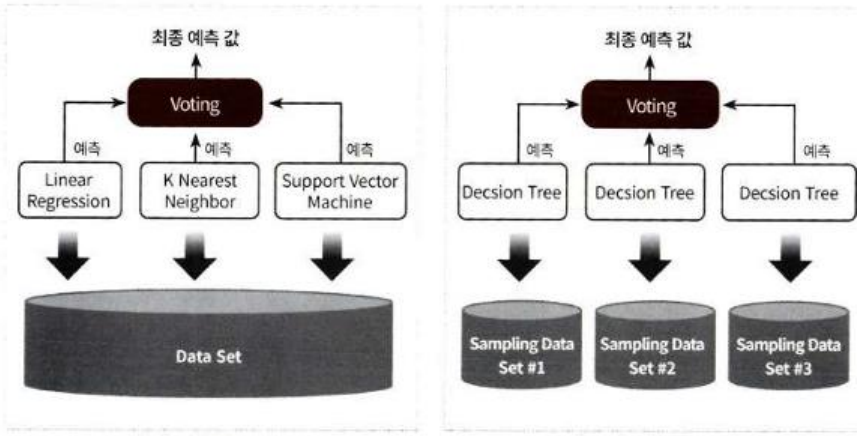
- 결정 트리 기본 하이퍼 파라미터 설정: 리프 노드 안 데이터가 모두 균일하거나, 하나만 존재해야 하는 엄격한 분할 기준
  → 일부 이상치(Outlier) 데이터 분류 위해 분할 자주 일어남
- 하이퍼 파라미터 조절로 노드 생성 규칙 완화
  → 이상치에 덜 반응, 이 모델이 더 뛰어날 가능성이 높음

03 앙상블 학습

1) 앙상블 학습

- 목표: 다양한 분류기의 예측 결과 결합 → 단일 분류기보다 신뢰성이 높은 예측값 얻는 것
- 정형 데이터 분류에서 딥러닝보다 뛰어난 성능
- 랜덤 포레스트, 그래디언트 부스팅 알고리즘, XGBoost, LightGBM, 스태킹(Stacking)
- 쉽고 편하면서 강력한 성능 보유
- 앙상블 학습 유형
    - **보팅(Voting)**
      - 여러 개의 **다른** 알고리즘 가진 분류기가 투표를 통해 최종 예측 결과 결정
      - **같은** 데이터 세트 학습, 예측한 결과로 보팅

- **배깅(Bagging)**
  - 여러 개의 **같은** 알고리즘 가진 분류기가 투표를 통해 최종 예측 결과 결정
  - **다른** 데이터 세트 학습, 부트스트래핑(Bootstrapping) 분할 방식
  : 원본 학습 데이터를 샘플링해 개별 분류기에 할당, 교차 검증과 달리 데이터 세트 간에 중첩 허용

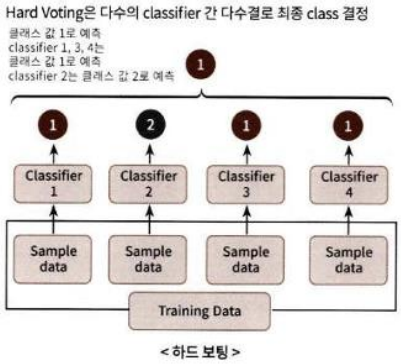


Voting 방식 Bagging 방식

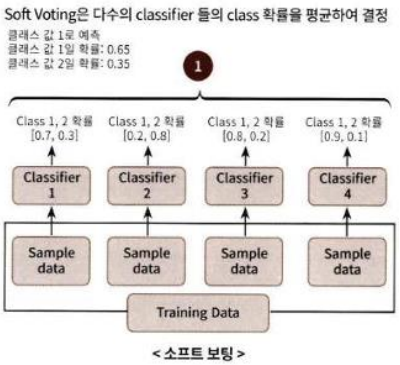
- **부스팅(Boosting)**
  - 순차적으로 학습 수행
  - 앞에서 예측 틀린 데이터 다음 분류기에서 **가**중치(Weight) 부여
  - 그래디언트 부스트, XGBoost(eXtra Gradient Boost), LightGBM(Light Gradient Boost)
- 스태깅
  - 여러 가지 다른 모델의 예측 결과값을 다시 학습 데이터로 만들어서 다른 모델(메타 모델)로 재학습해 결과 예측

2) 보팅 유형 – 하드 보팅(Hard Voting)과 소프트 보팅(Soft Voting)

- 하드 보팅(Hard Voting)
  - 다수결의 원칙: 예측 결과값 중 **다수**의 분류기가 결정한 예측값이 최종 보팅 결과값
- 소프트 보팅(Soft Voting)
  - 분류기들의 레이블 값 결정 **확률 합을 평균**해 확률이 가장 높은 레이블 값이 최종 보팅 결과값
  - 하드 보팅보다 예측 성능이 높음!



<하드 보팅>



<소프트 보팅>

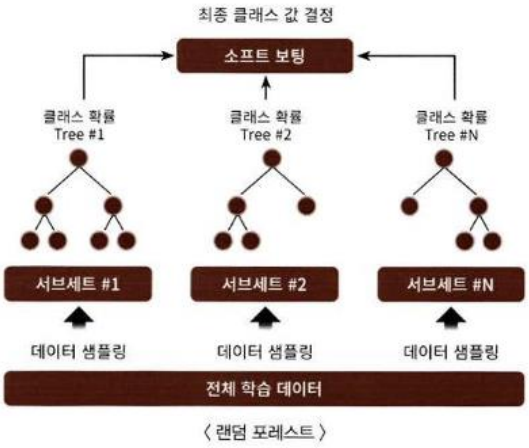
3) 보팅 분류기(Voting Classifier)

- Voting Classifier: 사이킷런의 보팅 방식 앙상블 구현 클래스
    - 주요 파라미터
      - estimator: 보팅에 사용될 여러 Classifier 객체들을 튜플 형식의 리스트 값으로 입력받음
      - voting: 'hard' = 하드 보팅(디폴트), 'soft' = 소프트 보팅
- 앙상블 방법은 전반적으로 단일 ML 알고리즘 보다 뛰어난 예측 성능 가짐
- ML 모델의 평가 요소: 얼마나 높은 유연성으로 현실 대처를 할 수 있는지 ∴ 편향-분산 트레이드 오프 극복은 ML 모델의 중요 과제
- 앙상블에서 결정 트리 알고리즘의 과적합을 많은 분류기 결합으로 극복

04 랜덤 포레스트

1) 랜덤 포레스트의 개요

- 앙상블 알고리즘 중 비교적 빠른 수행 속도를 가짐
- 다양한 영역에서 높은 예측 성능
- 기반 알고리즘은 결정 트리로, 결정 트리의 쉽고 직관적인 장점 그대로 가짐
- 여러 개의 결정 트리 분류기가 배깅 방식으로 각각의 데이터를 샘플링해 학습 수행 → 모든 분류기가 보팅을 통해 예측 결정



〈 랜덤 포레스트 〉

- 개별 트리 학습 데이터 세트 = 랜덤 트리 서브세트(Subset) 데이터(부트스트래핑으로 데이터가 임의로 만들어짐)
    - 부트스트래핑(bootstrapping) 분할 방식: 여러 개의 데이터 세트를 중첩되게 분리
    - 서브세트의 데이터 건수는 전체 데이터 건수와 동일하지만, 개별 데이터가 중첩되어 만들어짐
- RandomForestClassifier: 사이킷런의 랜덤 포레스트 기반 분류 지원 클래스

2) 랜덤 포레스트 하이퍼 파라미터 및 튜닝

- 트리 기반 앙상블 알고리즘 단점: 하이퍼 파라미터가 많아 튜닝을 위한 시간 소모↑ + 튜닝 후 예측 성능의 큰 향상을 기대할 수 없음
- 랜덤 포레스트 하이퍼 파라미터

- 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_features는 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_depth나 min_samples_leaf와 같이 결정 트리에서 과적합을 개선하기 위해 사용되는 파라미터가 랜덤 포레스트에도 똑같이 적용될 수 있습니다.