4.1~4.4

01 Classification의 개요

Supervised Learning: 명시적인 정답이 있는 데이터가 주어진 상태에서 학습하는 머신러닝 방식

Classification

1. 기존 데이터가 어떤 레이블에 속하는지 패턴을 알고리즘으로 인지

   : 학습 데이터로 주어진 데이터의 피처, 레이블값(결정값, 클래스값)을 머신러닝 알고리즘으로 학습해 모델을 생성

2. 새롭게 관측된 데이터에 대한 레이블 판별

   : 생성된 모델에 새로운 데이터 값이 주어졌을 때 미지의 레이블 값을 예측

앙상블

: 정형 데이터 예측 분석 영역에서 매우 높은 예측 성능을 보여줌, 결정 트리 알고리즘 사용

- 배깅
  - 랜덤 포레스트: 뛰어난 예측 성능, 상대적으로 빠른 수행 시간. 유연성
- 부스팅
  - 그래디언트 부스팅: 뛰어난 예측 성능, 수행시간 너무 오래걸려서 최적화 모델 튜닝 어려움
  - XgBoost, LightGBM: 그래디언트 부스팅보다 예측 성능 발전, 수행 시간 단축 ⇒ 가장 활용도 높음

결정 트리

- 매우 쉽고 유연하게 적용될 수 있음
- 데이터의 스케일링, 정규화 등 사전 가공의 영향이 매우 적음
- 예측 성능 향상을 위한 복잡한 규칙 구조 ⇒ overfitting 문제가 발생해 오히려 예측이 저하되기도..
- 앙상블에서는overfitting이 오히려 장점!
  - 예측 성능이 상대적으로 떨어지는 학습 알고리즘을 결합해 확률적 보완, 오류가 발생한 부분에 대한 가중치를 계속 업데이트함

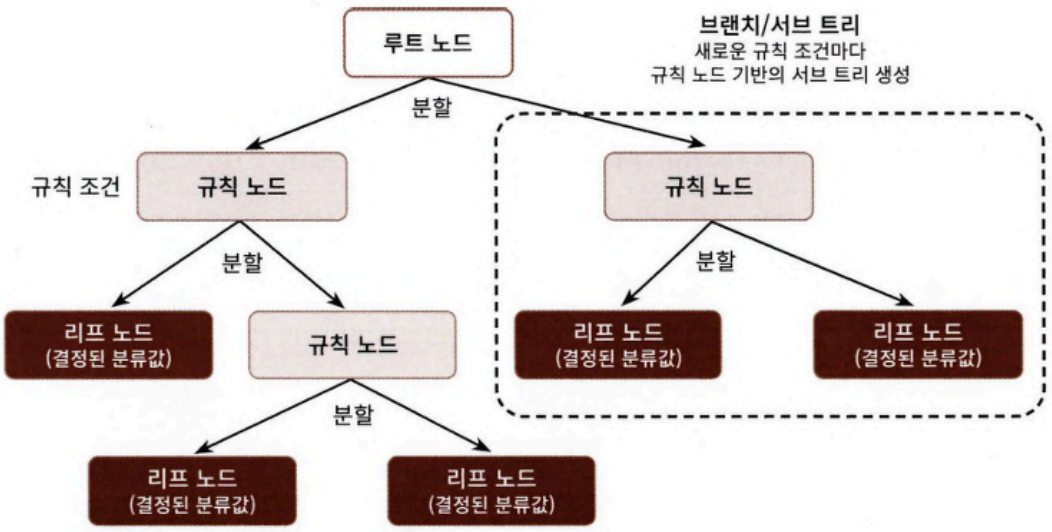
- 이때, 결정 트리가 좋은 학습기임

02 결정 트리

결정 트리

- 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내 Tree 기반 분류 규칙을 만드는 것
- if/else 기반으로 나타냄
- 성능 좌우하는 것: 데이터의 어떤 기준으로 규칙을 만들어야 가장 효율적인 분류가 될 것인가

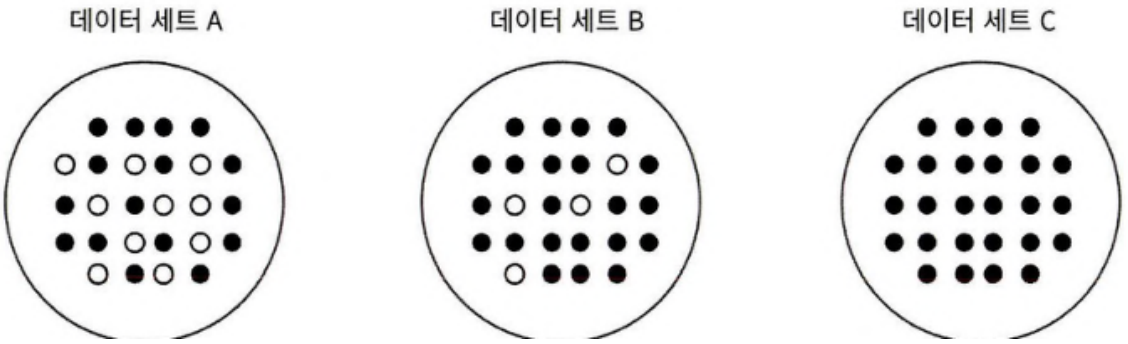
결정 트리의 구조



- 규칙 노드: 규칙 조건
  - 데이터 세트에 있는 피처가 결합해 규칙 조건 만들 때마다 규칙 노드가 만들어짐
  - 규칙 많아짐 = Tree의 depth 가 깊어짐 ⇒ overfitting
  - 해결 방법: 데이터를 분류할 때 최대한 많은 데이터 셋이 해당 분류에 속할 수 있도록 규칙을 정해야 함 ⇒ 적은 결정 노드, 높은 예측 정확도
- 리프 노드: 결정된 클래스 값
- 새로운 교칙 조건마다 서브 트리가 생성됨

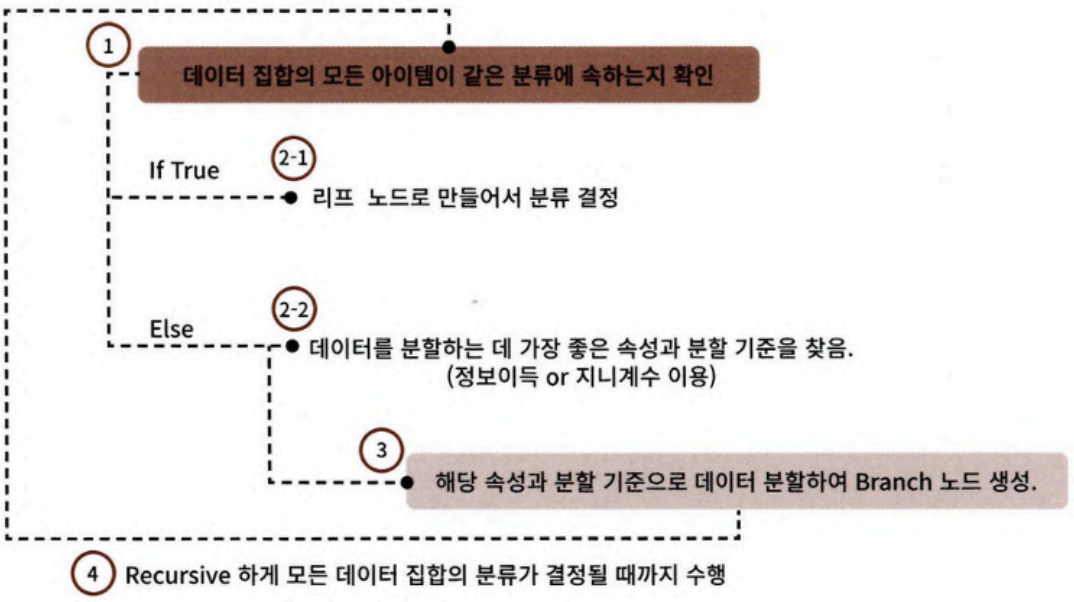
- 결정노드는 정보 균일도가 높은 데이터셋을 먼저 선택할 수 있도록 규칙 조건을 만듦
  1. 정보 균일도가 데이터셋으로 쪼개질 수 있도록 조건을 찾아 서브 데이터셋을 만듦
  2. 다시 이 서브 데이터셋에서 균일도가 높은 자식 데이터셋을 쪼갬
  3. 이걸 자식 트리로 내려가면서 반복하는 방식으로 데이터값을 예측함

정보 균일도 측정



균일도 높은 순서: C > B > A

- 정보이득
  - 엔트로피 기반
  - *엔트로피: 주어진 데이터 집합의 혼잡도.. 서로 다른 값이 섞여있을 수록 엔트로피가 높음
  - 정보 이득 지수 = 1 - 엔트로피 지수 ; 분할 기준 정하는 척도
  - 결정 트리는 정보 이득이 높은 속성을 기준으로 분할함
- 지니계수
  - 지니 계수가 낮을수록 데이터 균일도가 높음; 지니 계수가 낮은 속성을 기준으로 분할함
  - 사이킷런에서 구현한 DecisionTreeClassifier가 지니 계수를 이용해 데이터셋을 분할함



결정 트리 모델의 특징

- 장점
  - 균일도 기반 ⇒ 알고리즘이 쉽고 직관적
  - 룰이 매우 명확
  - 어떻게 규칙노드, 리프노드가 만들어지는지 알 수 있음
  - 시각화로 표현 가능
  - 정보 균일도만 고려하면 됨 ⇒ 각 피처의 스케이링, 정규화같은 전처리 작업 불필요
- 단점
  - overfitting으로 성능이 떨어짐 ⇒ 트리의 크기를 사전에 제한하는 튜닝이 필요함

결정 트리 파라미터

DecisionTreeClassifier: 분류, DecisionTreeRegressor: 회귀..

결정 트리 구현은 CART 기반

<DecisionTreeeClassifier 주요 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
min_samples_leaf	• 분할이 될 경우 왼쪽과 오른쪽의 브랜치 노드에서 가져야 할 최소한의 샘플 데이터 수 • 큰 값으로 설정될수록, 분할될 경우 왼쪽과 오른쪽의 브랜치 노드에서 가져야 할 최소한의 샘플 데이터 수 조건을 만족시키기가 어려우므로 노드 분할을 상대적으로 덜 수행함. • min_samples_split와 유사하게 과적합 제어 용도. 그러나 비대칭적(imbalanced) 데이터의 경우 특정 클래스의 데이터가 극도로 작을 수 있으므로 이 경우는 작게 설정 필요.

파라미터 명	설명
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

결정 트리 모델의 시각화

- Graphviz 패키지 사용
- export_graphize(): 함수 인자로 학습이 완료된 Estimator, 피처의 이름 리스트, 레이블 이름 리스트를 입력하면 학습된 결정 트리 규칙을 실제 트리 형태로 보여줌
- 리프노드
  - 최종 레이블 값이 결정되는 노드
  - 오직 하나의 클래스 값으로 최종 데이터가 구성되거나 리프 노드가 될 수 있는 하이퍼 파라미터 조건을 충족하면 됨
- 브랜치 노드
  - 자식 노드가 있음
  - 자식 노드를 만들기 위한 규칙 조건을 가지고 있음

결정 트리 실습 및 설명은 ipynb 파일에 있습니다.

결정 트리 Overfitting

make_classification(): classification을 위한 테스트용 데이터를 쉽게 만들 수 있음

- 기본 하이퍼 파라미터 설정
  - outlier 분류하기 위해 분할이 자주 일어남
  - 리프 노드 안에 데이터가 모두 균일 or 하나만 존재해야하는 엄격한 분할 기준

  ⇒ 결정 기준 경계가 많아지고 복잡함
  - 복잡한 모델; 학습 데이터셋과 약간 다른 형태를 예측하면 예측 정확도 떨어짐..
- 리프 노드 생성 규칙 완화, 하이퍼 파라미터 변경
  - 이상치에 크게 반응안함
  - 일반화된 분류 규칙에 따라 분류됨

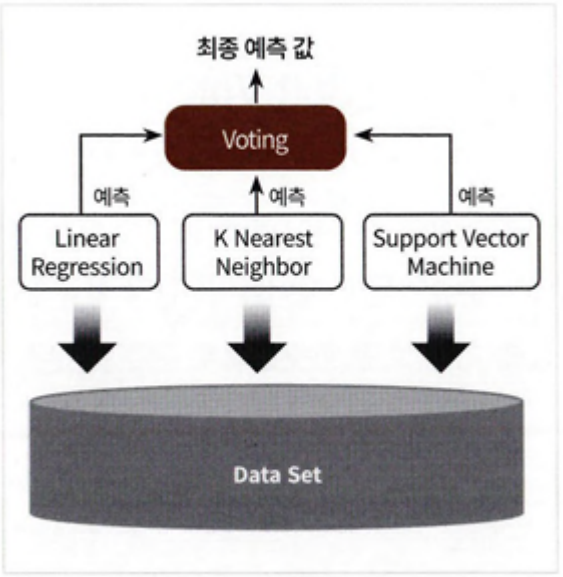
결정트리 실습 - 사용자 행동 인식 데이터셋

실습 및 설명은 ipynb 파일에 있습니다.

03 앙상블 학습

앙상블 학습 개요

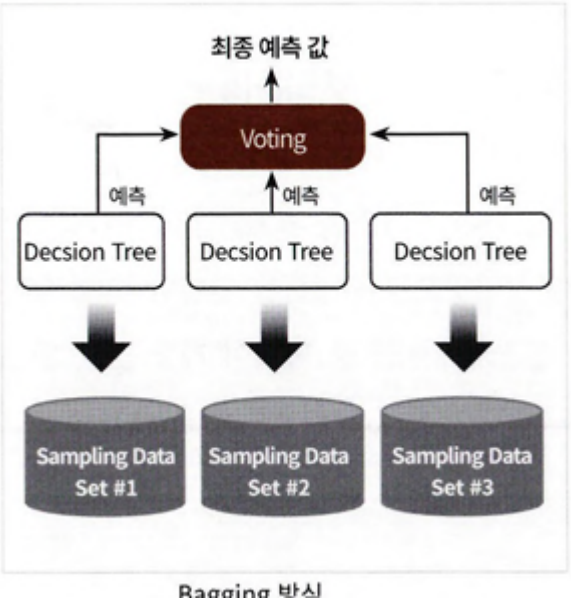
- 앙상블 학습을 통한 분류: 여러 classifier를 생성하고, 그 예측을 결합 ⇒ 정확한 최종 예측 도출
- 정형 데이터 분류에선 앙상블 성능이 뛰어남
- 유형
-
  - 보팅



Voting 방식

- 여러 개의 classifier가 투표를 통해 최종 예측 결과를 결정하는 방식
- 서로 다른 알고리즘을 가진 classifier를 결합

◦ 배깅



Bagging 방식

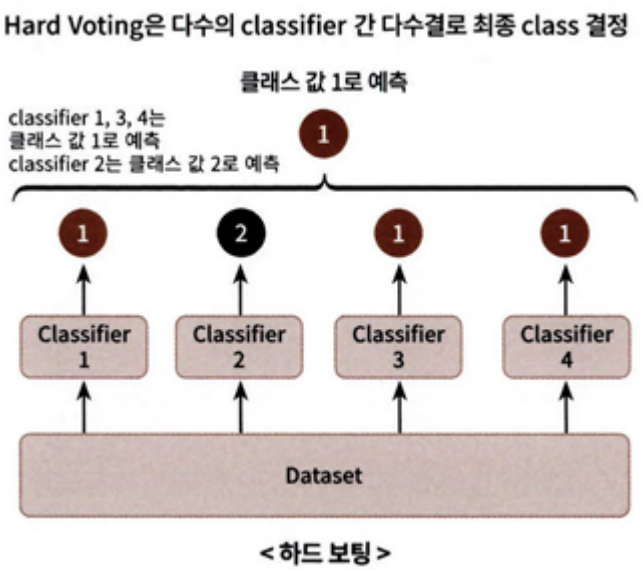
- 여러 개의 classifier가 투표를 통해 최종 예측 결과를 결정하는 방식
- classifier가 모두 같은 알고리즘 기반
- 데이터 샘플링을 다르게 가져가면서 학습

◦ 부스팅

- 여러 개의 classifier가 순차적으로 학습 수행
- 이미 학습한 classifier가 예측이 틀린 데이터를 제대로 예측할 수 있도록 그다음 classifier에서는 weight 부여
- 예측 성능이 뛰어나 앙상블 학습 주도중

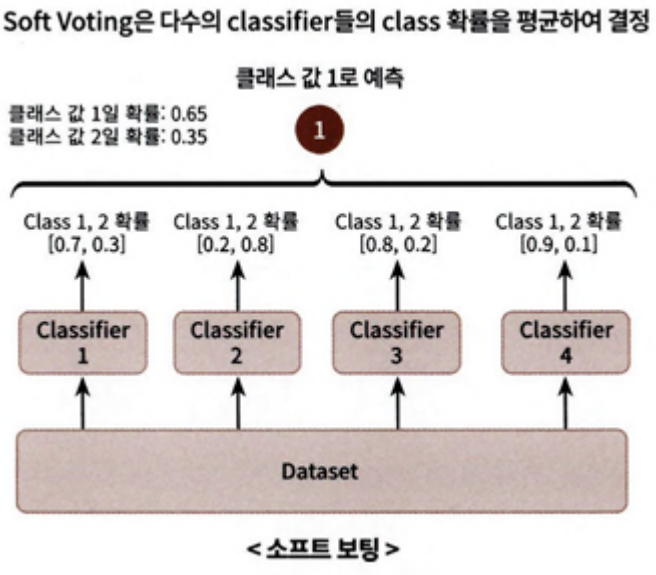
하드 보팅, 소프트 보팅

- 하드 보팅



< 하드 보팅 >

  - 다수결 원칙과 유사
  - 예측한 결괏값 중 다수의 classifier가 결정한 예측값을 최종 보팅 결괏값으로!
- 소프트 보팅



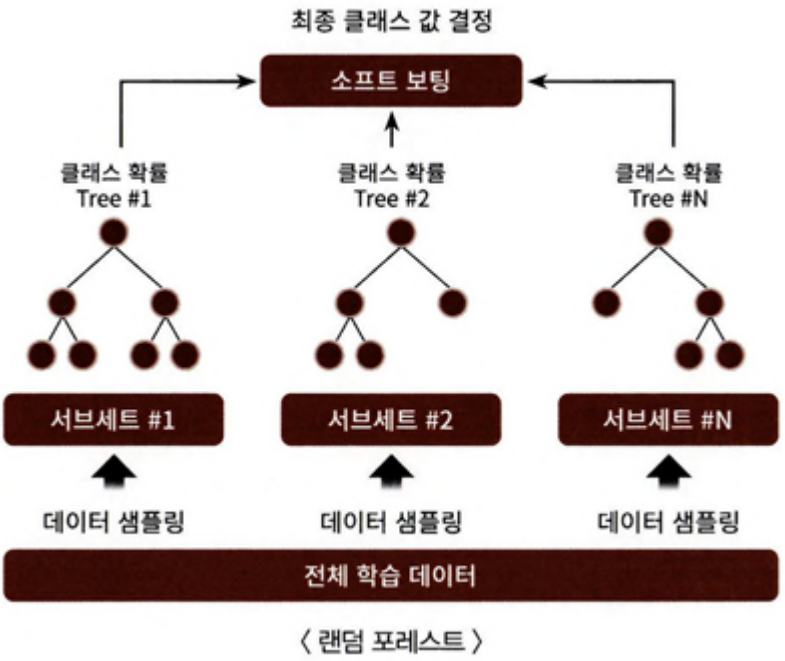
Voting Classifier

- sklearn은 VotingClassifier를 제공함
- VotingClassifier 주요 생성 인자
  - estimators: 리스트 값으로 voting에 사용될 여러 개의 classifier 객체들을 튜플 형식으로 입력 받음
  - voting: hard, soft 입력 받음, 기본값은 hard!
- 앙상블 방법은 다른 ML 알고리즘보다 뛰어난 예측 성능을 가짐
- ML 모델 평가 요소: 어떻게 높은 유연성을 갖고 현실에 대체할까? 과제: 편향-분산 트레이드 오프 해결해야 함
- 앙상블 학습: 결정 트리 알고리즘의 장점 + 단점 보완 + 편향-분산 트레이드 오프 효과 극대화

04 Random Forest

랜덤 포레스트 개요 및 실습

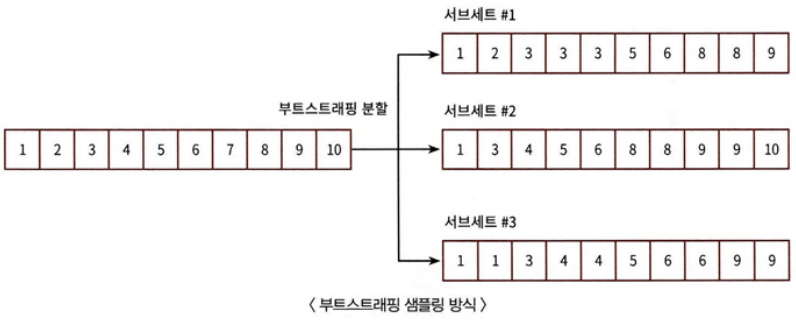
- 배깅: 같은 알고리즘으로 여러 개의 classifier를 만들어서 보팅으로 최종 결정하는 알고리즘 ex) 랜덤 포레스트
- 특징
  - 앙상블 알고리즘 중에서 비교적 빠른 수행 속도
  - 다양한 영역에서 높은 예측 성능
  - 결정 트리의 쉽고 직관적인 장점을 그대로 갖고있음
- 과정



〈 랜덤 포레스트 〉

1. 여러 개의 결정 트리 classifier가 전체 데이터에서 배깅 방식으로 각자 데이터를 샘플링
2. 개별적으로 학습 수행
3. 모든 classifier가 보팅을 통해 예측 결정

- 부트스트래핑 분할 방식

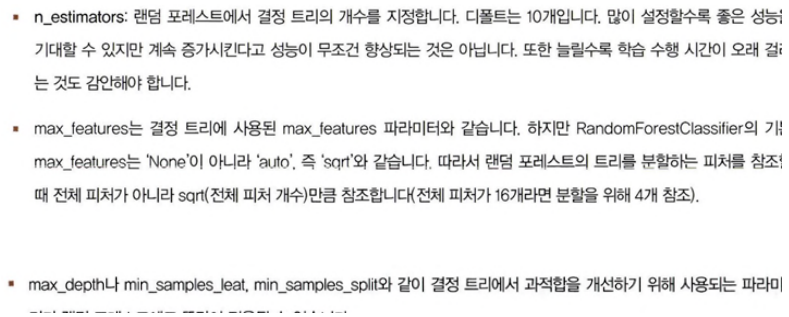


〈부트스트래핑 샘플링 방식〉

- 여러 개의 데이터셋을 중첩되게 분리하는 것
- 여러 개의 작은 데이터셋을 임의로 만듦 → 개별 평균의 분포도 측정
- subset데이터: 부트스트래핑으로 데이터가 임의로 생성
- subset 데이터 수 = 전체 데이터 수 이지만, 개별 데이터가 중첩되어 만들어짐
- 랜덤 포레스트: 데이터가 중첩된 개별 데이터셋에 결정 트리 classifier를 각각 적용하는 것
- sklearn은 RandomForestClassifier를 제공함

랜덤 포레스트 하이퍼 파라미터 및 튜닝

- 트리 기반 앙상블 알고리즘의 단점
  - 하이퍼 파라미터가 너무 많음 ⇒ 튜닝을 위해 시간 많이 소모됨
  - 많은 시간을 소모했어도 튜닝 후 예측 성능이 크게 향상되는 경우는 적음..
  - 결정 트리에서 사용되는 하이퍼 파라미터


- n_estimators: 랜덤 포레스트에서 결정 트리의 개수를 지정합니다. 디폴트는 10개입니다. 많이 설정할수록 좋은 성능
기대할 수 있지만 계속 증가시킨다고 성능이 무조건 향상되는 것은 아닙니다. 또한 늘릴수록 학습 수행 시간이 오래 걸
는 것도 감안해야 합니다.
- max_features는 결정 트리에 사용된 max_features 파라미터와 같습니다. 하지만 RandomForestClassifier의 기
max_features는 'None'이 아니라 'auto', 즉 'sqrt'와 같습니다. 따라서 랜덤 포레스트의 트리를 분할하는 피처를 참조
때 전체 피처가 아니라 sqrt(전체 피처 개수)만큼 참조합니다(전체 피처가 16개라면 분할을 위해 4개 참조).
- max_depth나 min_samples_leat, min_samples_split와 같이 결정 트리에서 과적합을 개선하기 위해 사용되는 파라미

GridSearchCV() 실습 및 설명은 ipynb 파일에 있습니다.