Supervised Learning

분류와 회귀

- 지도학습의 두 종류.

- 분류
  - » 미리 정의된 가능성 있는 여러 클래스 레이블 중 하나를 예측하는 것
  - » 두 개의 클래스로 분류하는 이진 분류와 셋 이상의 클래스로 분류하는 다중 분류

- 회귀
  - » 연속적인 숫자 또는 부동소수점(실수) 데이터를 예측하는 것

- 출력 값의 연속성 여부가 두 기법을 구분하는 중요한 기준
  - » 일반적으로 연속성이 있으면 회귀, 없으면 분류
  - » 양적 데이터는 회귀, 범주형 데이터는 분류

일반화, 과대적합, 과소적합

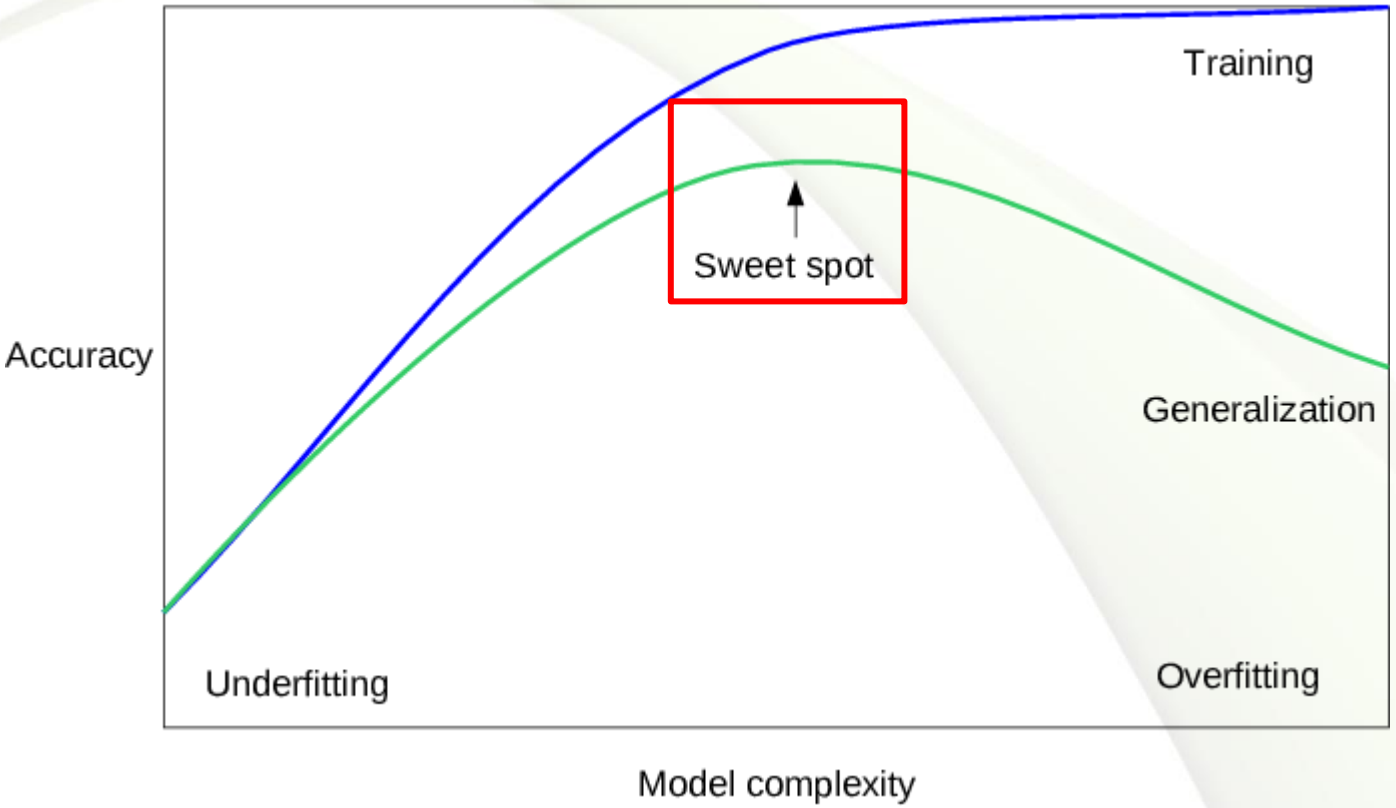
- 훈련 세트에서 테스트 세트로 일반화
  - » 모델이 처음 보는 데이터에 대해 정확하게 예측할 수 있게 되는 것
  - » 모델을 만들 때 가능한 정확하게 일반화하도록 구현해야 함

- 과대적합 (Overfitting)
  - » 모델이 훈련 세트에 너무 가깝게 맞춰져서 새로운 데이터에 일반화되기 어려운 경우
  - » 훈련 데이터는 잘 설명하지만 새로운 데이터에 대한 예측 정확도가 낮음

- 과소적합 (Underfitting)
  - » 모델을 지나치게 단순화해서 훈련 데이터와 테스트 데이터 모두에서 예측 정확도가 낮음

일반화, 과대적합, 과소적합

- 일반화 성능을 최대화 하는 모델을 찾는 것이 데이터 분석의 목표



- 일반적으로 데이터가 많으면 다양성을 강화하기 때문에 큰 데이터 셋을 사용하면 과대적합 없이 복잡한 모델을 만드는 것 가능

Classification

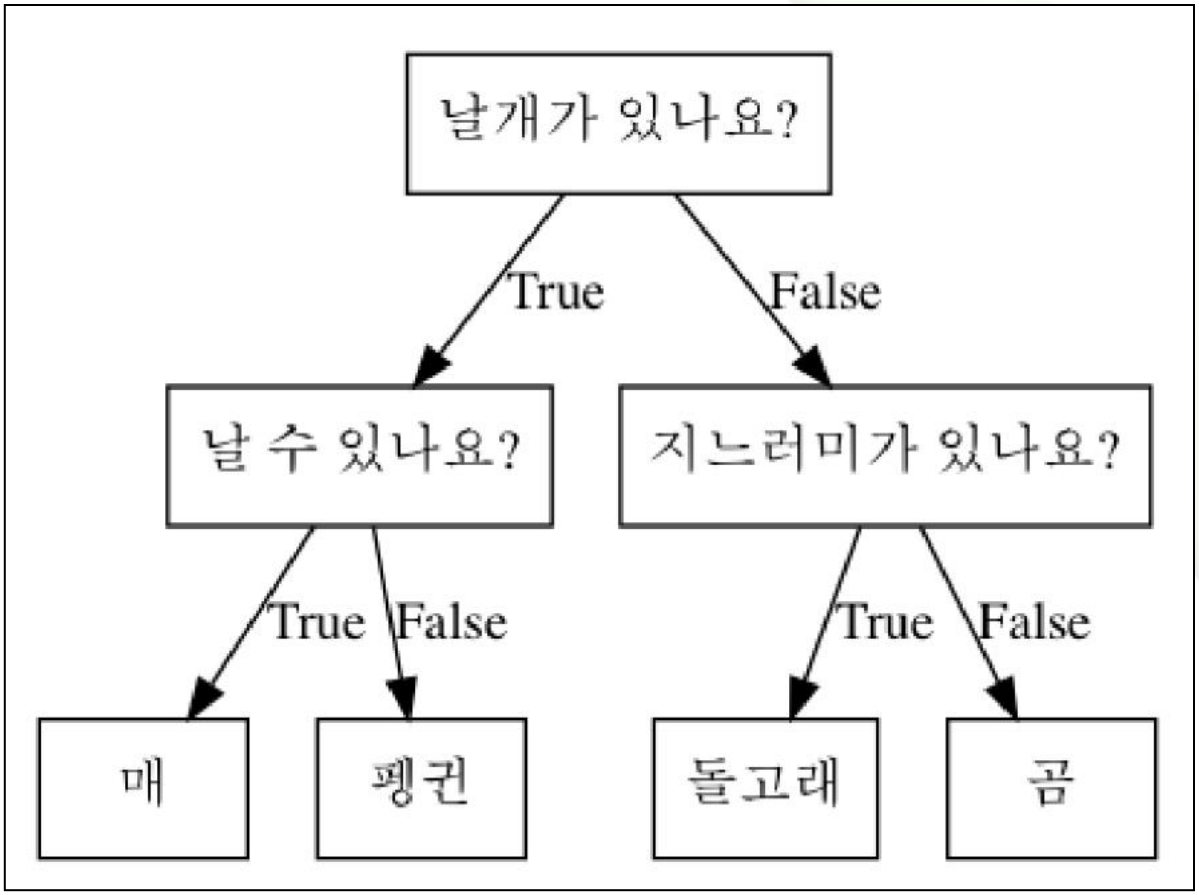
분류 알고리즘

알고리즘	설명
나이브베이즈	Bayes 통계와 생성 모델 기반
로지스틱 회귀	독립 변수와 종속 변수의 선형 관계 기반
결정 트리	데이터 균일도(또는 불순도)에 따른 규칙 기반
서포트 벡터 머신	개별 클래스 사이의 최대 마진
최근접 이웃	근접 거리 기준
앙상블	서로 다른 (또는 같은) 여러 개의 알고리즘 결합
신경망	심층 연결 기반

결정 트리

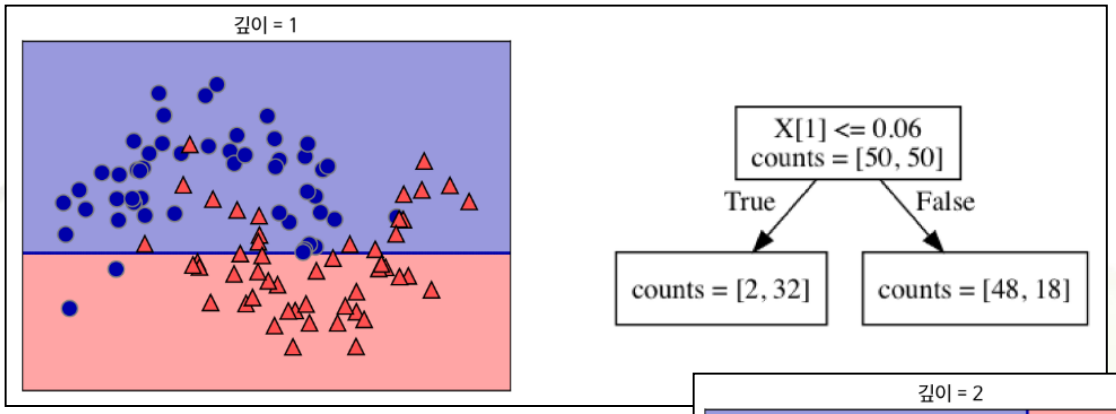
- 분류와 회귀에 광범위하게 사용되는 모델

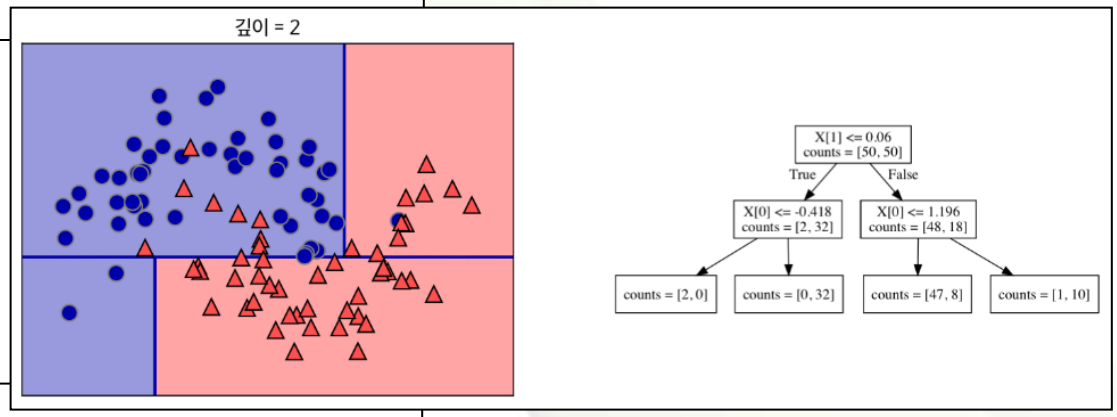
- 결정에 다다르기 위해 예/아니오 질문을 이어 나가면서 학습

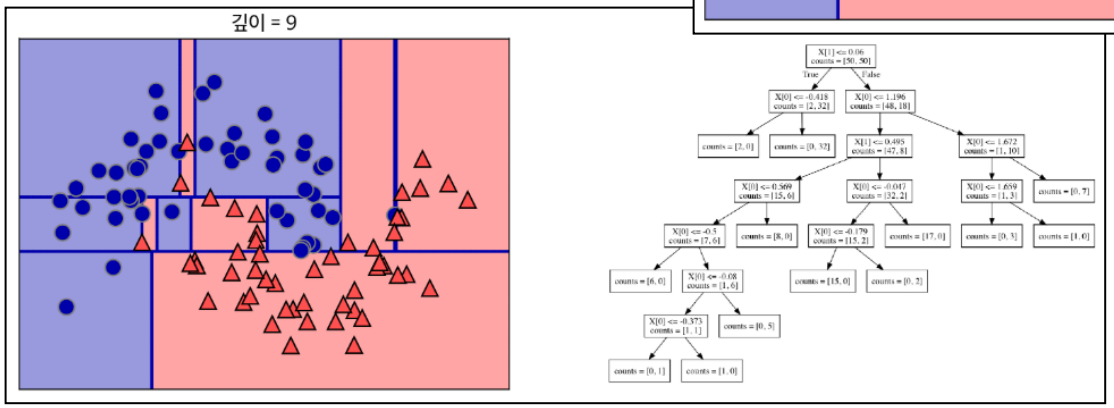


트리 구조의 모델 형성

결정 트리 만들기


깊이 = 1
X[1] <= 0.06
counts = [50, 50]
True
False
counts = [2, 32]
counts = [48, 18]


깊이 = 2
X[1] <= 0.06
counts = [50, 50]
True
False
X[0] <= -0.418
counts = [2, 32]
X[0] <= 1.196
counts = [48, 18]
counts = [2, 0]
counts = [0, 32]
counts = [47, 8]
counts = [1, 10]


깊이 = 9

결정 트리

- 노드 종류
  - » 규칙 노드 → 규칙 조건에 따라 하위 트리를 생성하는 노드
  - » 리프 노드 → 트리의 마지막 노드로 최종 클래스 결정

- 규칙 노드가 많아져서 트리가 깊어지는 것은 분류 결정 방식이 복잡해지기 때문에 과대적합 경향성이 높아짐

- 가능한 적은 결정 노드로 높은 예측 정확도를 도출하는 것이 중요
  - » 데이터를 분류할 때 최대한 많은 데이터 세트가 해당 분류에 속할 수 있도록 결정 노드의 규칙 결정
  - » 정보 균일도가 높은 데이터 세트를 먼저 선택할 수 있도록 규칙 설정

- 정보 균일도 측정
  - » 엔트로피 → 데이터 순도와 반비례
  - » 지니 계수 → 데이터의 균일도와 반비례

복잡도 제어

- 순수 노드 → 하나의 타겟 클래스로 구성된 노드

- 모든 리프 노드가 순수 노드가 될 때까지 진행하면 모델이 매우 복잡해지고 훈련 데이터에 과대적합됨 → 순수 노드로 이루어진 트리는 훈련 데이터에 100% 정확하게 맞는 모델

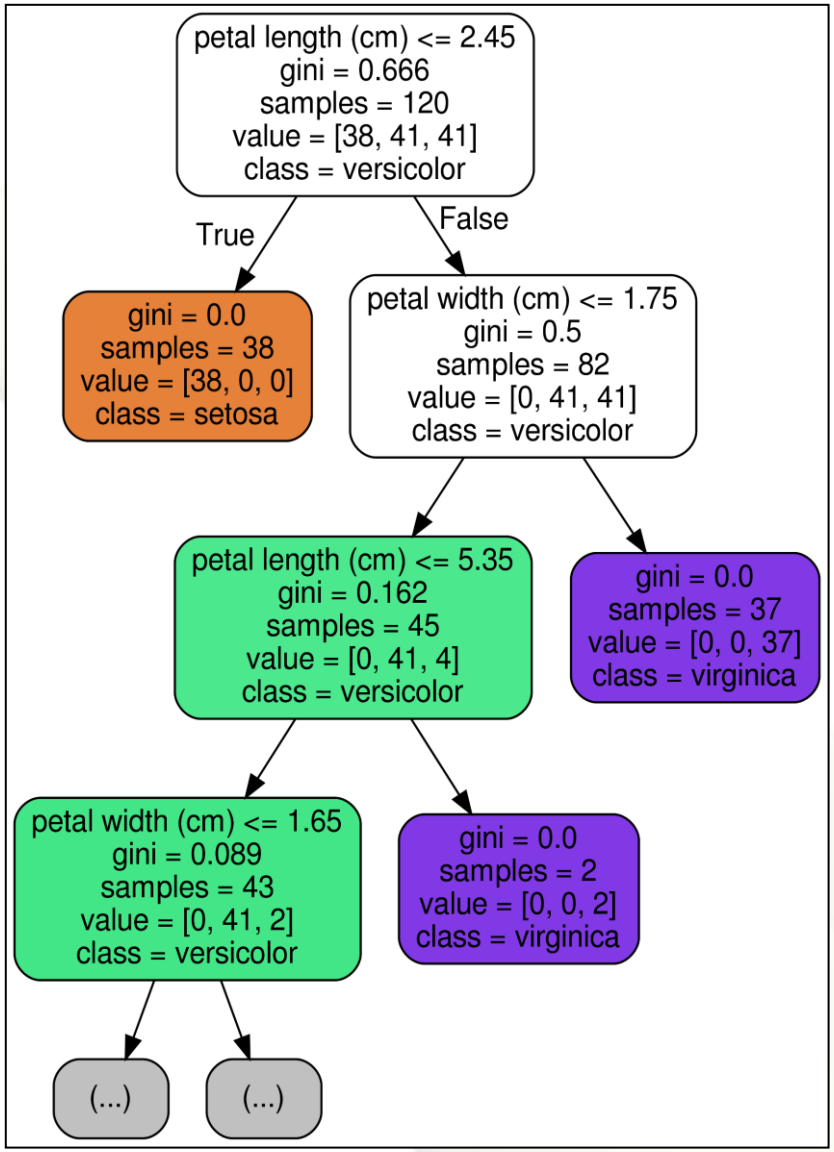
- 과대적합을 막는 방법은
  - » 트리 생성을 일찍 중단하기 (사전 가지치기)
  - » 데이터 포인트가 적은 노드를 삭제하거나 병합 (사후 가지치기)

- scikit-learn은 사전 가지치기만 지원
  - » min_samples_split, min_samples_leaf, max_features, max_depth, max_leaf_nodes 등의 모델 파라미터를 사용해서 모델 과적합 통제

결정 트리 시각화

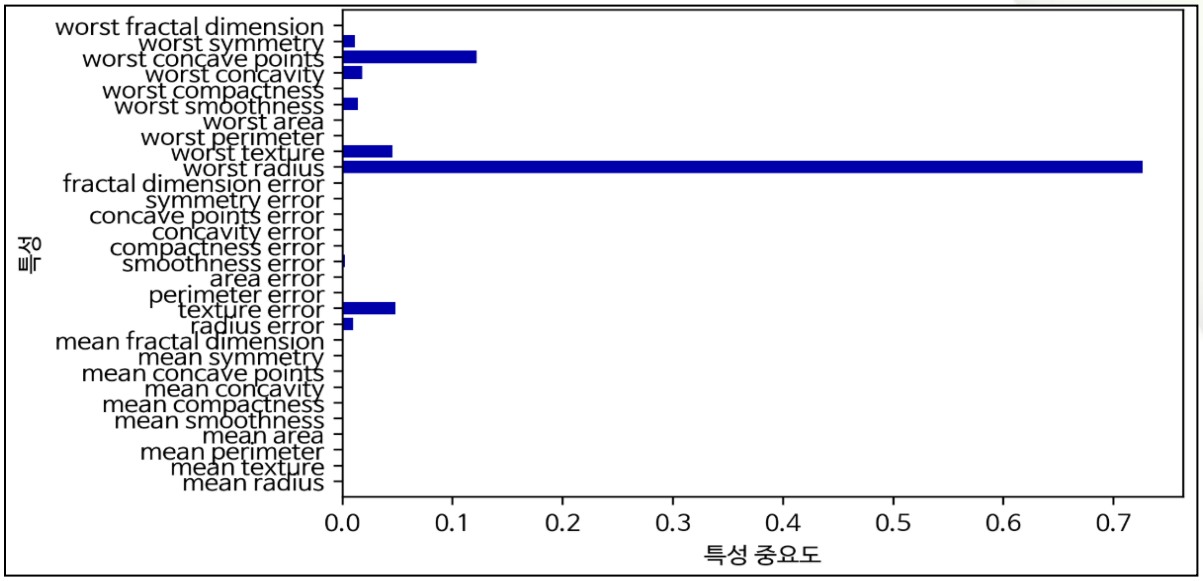
- graphviz 모듈을 사용해서 트리 시각화
  - » 알고리즘의 예측 프로세스를 이해할 수 있으며
  - » 비전문가에게 알고리즘 설명 쉬움

- 설치
  - » https://graphviz.org/download/ 에서 작업 환경에 맞는 제품 설치
  - » 환경 변수 설정 → PATH에 graphviz 실행 파일 위치 설정
  - » pip install graphviz 명령으로 파이썬 패키지 설치



트리의 특성 중요도

- 전체 트리를 살펴보는 대신 트리가 어떻게 동작하는지 요약
- 가장 널리 사용되며 트리를 만드는 결정에 각 특성이 얼마나 중요한지 평가
- 0과 1사이의 숫자로 표현
  - » 0은 전혀 사용되지 않음 / 1은 완벽하게 목표 클래스 예측
  - » 특성 중요도의 전체 합은 1



트리의 특성 중요도

- 특성 중요도가 낮은 것이 특성이 유용하지 않은 것을 의미하는 것은 아님
  - » 트리가 그 특성을 선택하지 않았다는 의미 (다른 특성이 동일한 정보를 지니고 있기 때문일 수 있음)

- 특성 중요도의 값은 항상 양수
  - » 값이 큰 것은 중요도만 제시할 뿐이며 양성 / 음성을 판단하는데 사용할 수 없음

장단점

- 장점
  - » 알고리즘이 쉽고 직관적
  - » 만들어진 모델을 쉽게 시각화 → 비전문가도 이해하기 쉬움
  - » 데이터의 스케일의 영향이 없음 → 스케일 처리 불필요

- 단점
  - » 사전 가지치기를 사용해도 과대적합되는 경향이 강해서 일반화 성능이 좋지 않음 → 대안으로 앙상블 방법 사용

앙상블 (Ensemble) 학습

앙상블 (Ensemble)

- 여러 머신러닝 모델을 연결해서 더 강력한 모델을 만드는 기법으로 대부분의 정형 데이터 분류 문제에서 앙상블이 뛰어나 성능을 보임

- Random Forest, Gradient Boosting Decision Tree, XGBoost, LightBGM 등 다양한 모델이 개발되어 사용 중

- 모델 유형
  - » Voting, Bagging, Boosting, Stacking 등

- 투표 유형
  - » 하드 보팅, 소프트 보팅

앙상블 유형

- 모델 유형

유형	설명
Voting	여러 개의 서로 다른 알고리즘의 분류기가 투표를 통해 예측
Bagging	여러 개의 같은 종류의 알고리즘의 분류기가 데이터 샘플링을 다르게 해서 학습을 수행하고 투표를 통해 예측
Boosting	여러 개의 분류기가 순차적으로 학습을 수행하되 앞에서 학습한 분류기의 오류를 반영한 가중치를 부여해서 다음 분류기의 학습과 예측 진행
Stacking	여러 가지 다른 모델의 예측 결과를 모아서 학습 데이터 세트를 만들고 다른 모델로 재학습해서 결과 예측

- 투표 유형

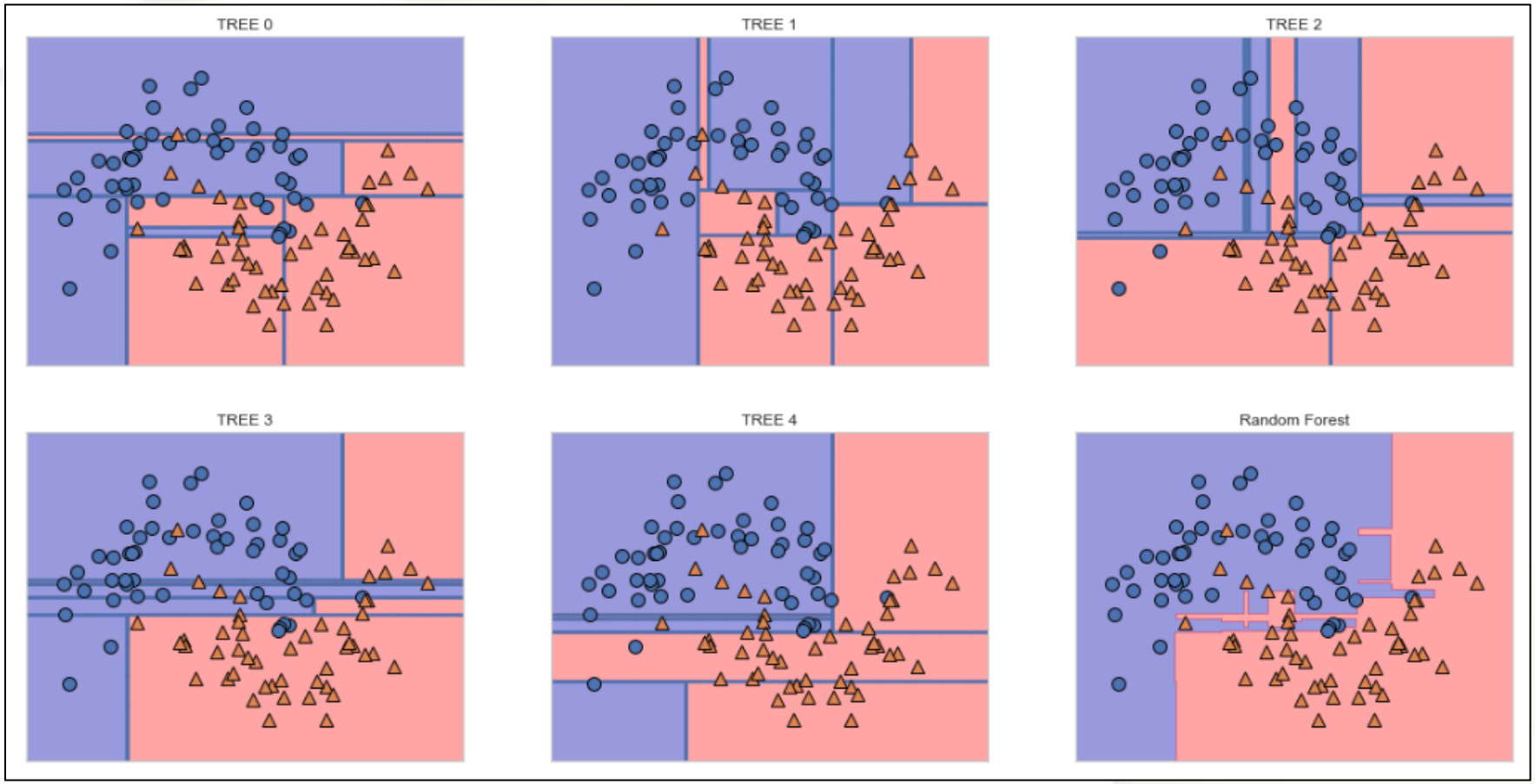
유형	설명
Hard Voting	여러 개의 서로 다른 알고리즘의 분류기가 투표를 통해 예측
Soft Voting	여러 개의 같은 종류의 알고리즘의 분류기가 데이터 샘플링을 다르게 해서 학습을 수행하고 투표를 통해 예측

랜덤 포레스트 (Random Forest)

- Bagging의 대표적인 알고리즘
- 결정트리의 주요 단점인 훈련 데이터에 과적합되는 경향을 회피하는 방법
- 조금씩 다른 여러 결정트리의 묶음
- 기본적으로 예측력이 좋으면서 서로 다른 방향으로 과적합된 트리를 많이 만들어 그 결과를 평균 내면 과적합된 양을 줄일 수 있다는 것이 수학적으로 검증됨
- 트리들이 서로 달라지도록 트리 생성시 무작위성 주입
  - » 데이터 포인트를 무작위로 선택하는 방법
  - » 분할 테스트에서 특성을 무작위로 선택하는 방법

랜덤 포레스트 (Random Forest)

- 개별 트리와 Random Forest



랜덤 포레스트 구축

- 생성할 트리 개수 결정
  - » n_estimators 매개 변수

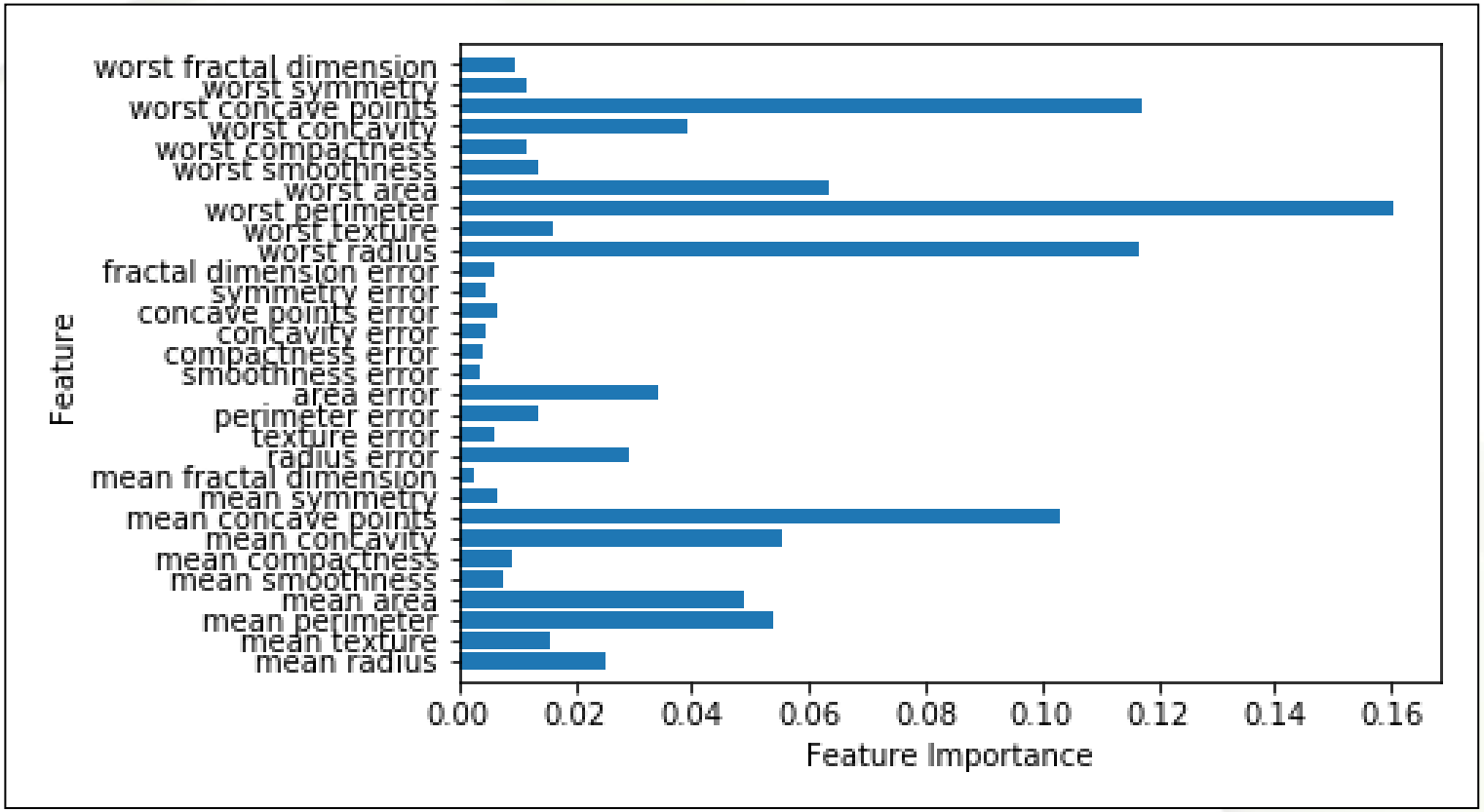
- 데이터의 부트스트랩 샘플 생성
  - » 원래 데이터 세트의 크기와 같지만 누락 및 중복을 허용하는 무작위 데이터 추출

- 전체 특성으로 예측하지 않고 각 노드에서 후보 특성을 무작위로 선택한 후 이 후보들 중에서 최선의 예측 도출

- 모든 트리의 예측을 만든 후
  - » 회귀의 경우에는 이 예측들을 평균하여 최종 예측 도출
  - » 분류의 경우에는 약한 투표 전략 사용 → 가능성 있는 출력 레이블의 확률 제공 → 가장 높은 확률을 가진 클래스가 예측 범주

랜덤 포레스트 구축

- 속성별 중요도 정보 도출 가능



랜덤 포레스트 (Random Forest)

- 장점
  - » 성능이 매우 뛰어나고
  - » 매개변수 튜닝을 많이 하지 않아도 잘 작동하며
  - » 데이터의 스케일을 맞출 필요도 없음

- 단점
  - » 텍스트 데이터와 같이 매우 차원이 높고 희소한 데이터에는 잘 작동하지 않음 → 선형 모델이 더 적합
  - » 선형 모델에 비해 많은 메모리를 사용하며 훈련과 예측이 느림

그래디언트 부스팅 결정 트리

- 여러 개의 결정 트리를 묶어 강력한 모델을 만드는 방법으로 트리가 많이 추가될수록 예측 성능 향상

- 회귀와 분류 모두에 사용 가능

- 이전 트리의 오차를 보완하면서 순차적으로 트리 생성
  - » 이전 트리의 오차를 얼마나 강하게 보정할 것인지 설정 (learning_rate)

종류	설명
AdaBoost	오류 데이터에 가중치를 부여하면서 부스팅 수행
GraditngBoosting	가중치 업데이트에 경사 하강법 적용

- 무작위성 없음 → 대신 과적합화를 막기 위해 사전 가지치기 사용

- 1 ~ 5 정도의 낮은 트리를 사용 → 메모리 사용량이 적고 예측도 빠름

그레디언트 부스팅 결정 트리

- 장점
  - » 특성의 스케일 조정 필요 없음

- 단점
  - » 매개변수를 잘 조정해야 의미 있는 결과 도출
  - » 훈련 시간이 오래 걸림
  - » 희소한 고차원 데이터에 대해 잘 작동하지 않음

XGBoost (eXtra Gradient Boost)

- 트리 기반 앙상블 학습에서 가장 인기 있는 알고리즘 중 하나
- 분류 영역에서 다른 머신러닝 알고리즘에 비해 상대적으로 뛰어난 예측 성능
- GBM 기반 → 느린 수행 시간을 개선하고 과적합 규제 기능을 추가하는 등의 개선 적용
- 핵심 라이브러리는 C/C++로 작성
  - » 자체 파이썬 패지지와
  - » scikit-learn과 호환 가능한 wrapper 클래스 제공
- 설치
  - » pip install xgboost (or conda install –c anaconda py-xgboost)

LightGBM

- XGBoost와 함께 가장 인기 있는 부스팅 계열 알고리즘
- 대부분의 트리 모델이 사용하는 균형 트리 분할 방식을 사용하지 않고 중심 트리 분할 방식 사용
- scikit-learn과 호환 가능한 wrapper 제공
- 장점
  - » XGBoost의 학습 시간이 오래 걸리는 문제를 보완
  - » 메모리 사용량이 상대적으로 적음
- 단점
  - » 적은 데이터 세트 (일반적으로 10,000건 이하의 데이터 세트)로 학습할 경우 과적합 되기 쉬움

Stacking Ensemble

- 여러 개의 모델을 결합해서 예측 결과 도출
- 개별 알고리즘으로 수행한 예측 데이터를 기반으로 다시 예측 수행 → 메타 모델
- 일반적으로 많이 사용하는 모델 X