Chapter 06. 통계적 머신러닝 - 김혜진

6. 통계적 머신러닝

: 통계적인 모형(회귀, 베이즈, 의사결정트리 등)을 사용해 기계를 학습 시키는 것

Part6 에서는 데이터의 전체 구조를 가정하지 않는 지도학습 모델 학습예정

(전체 데이터에 맞는 형태가 정해진 모델→ 선형회귀 모델

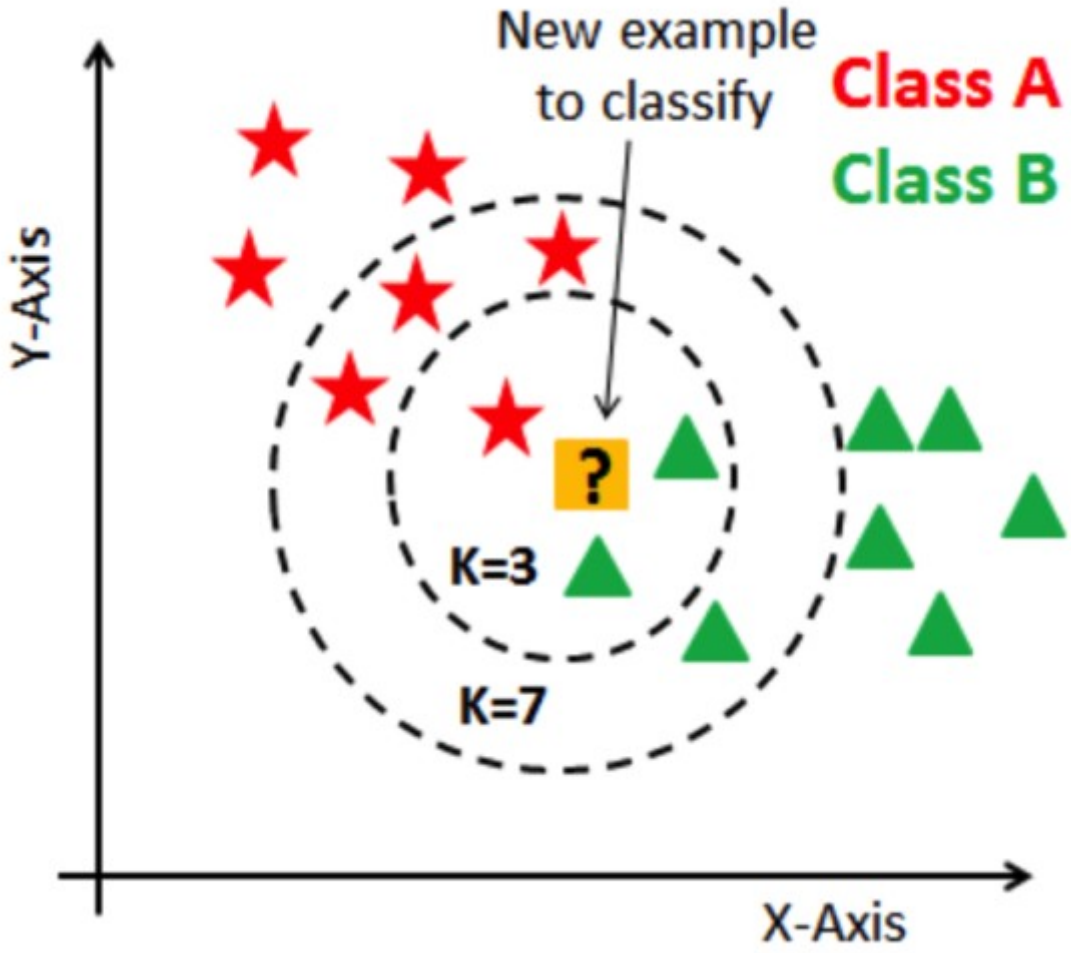
데이터에 따라 유연하게 학습하는 모델 → KNN, 트리 모델)

6.1 KNN

KNN = K-Nearest Neighbors (K-최근접 이웃), 주변 데이터를 참고하여 분류/ 예측 하는 모델

KNN을 활용한 분류

> 데이터가 주어졌을 때, 이와 가까이에 있는 데이터를 살펴본 후, 다수의 데이터가 포함된 범주로 분류



출처: datacamp.com

K = 3인 경우, 새로운 데이터는 Class B 분류

K = 7인 경우, 새로운 데이터는 Class A 분류

[KNN에서 고려할 것]

1. 적절한 K의 갯수

- KNN은 K를 어떻게 정하느냐에 따라 결과 값이 변함
- 여기서 K는 연구자가 직접 찾아야 하는 하이퍼 파라미터
  - K가 너무 작은 경우 이상치나 노이즈 데이터와 이웃될 가능성 존재 (overfitting)
  - K가 너무 큰 경우 해당 데이터 주변을 통해 유의미한 결론 도출 불가 (underfitting)
  - 일반적으로 K를 1에서 20 사이로 설정, 홀수를 사용

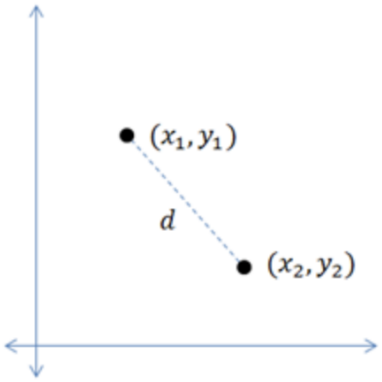
(짝수인 경우 주변 데이터의 범주가 정확히 반일 때 하나의 결과를 도출 불가)

- 데이터의 갯수나, 학습데이터에서 얼마나 좋은 성능을 보이는지 등을 확인하여 선정

2. 가까운 데이터의 기준

- 데이터(벡터) 사이의 거리 계산

a. 유클리드 거리

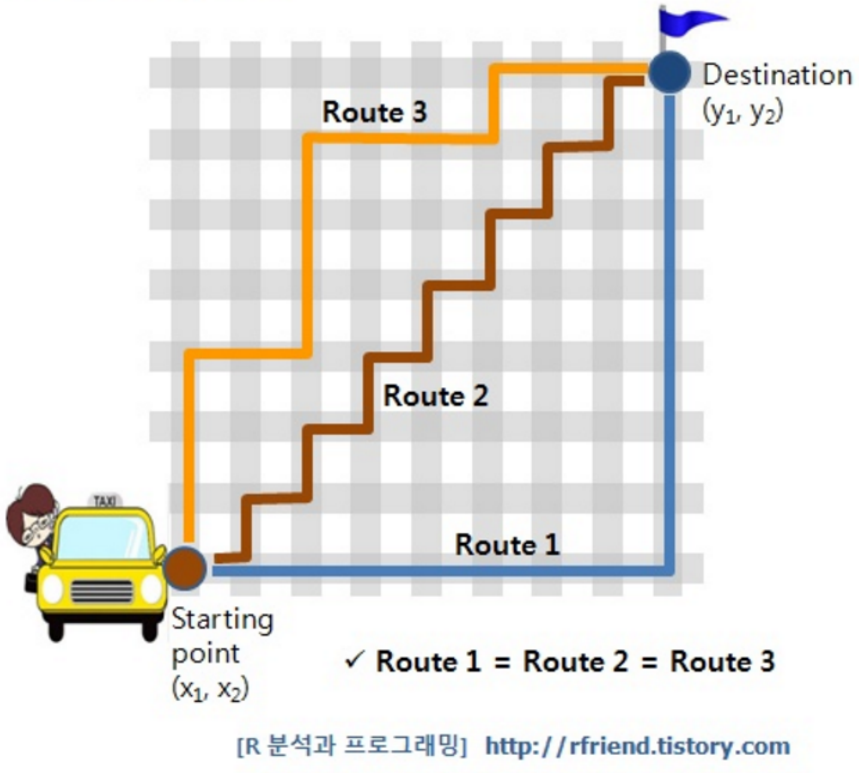


$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

출처: https://blog.naver.com/bulb17b/221863830806

피타고라스 정리를 통해 점과 점 사이의 직선거리 구할 수 있음

b. 맨해튼 거리



대각선이 아닌 한 축의 방향으로 움직인 거리

c. 마할라노비스 거리

계산에 공분산 행렬을 사용, 두 변수 간의 상관관계까지 고려하는 거리 계산 방법

→ 변수간의 상관 관계가 높으면 유용하지만 많은 계산이 필요, 복잡하다는 단점

3. 데이터의 표준화

- 일반적으로 예측변수를 표준화하여 스케일이 큰 변수들의 영향력이 너무 커지지 않도록 함

KNN을 활용한 예측(=KNN 회귀)

가까운 데이터들의 평균값으로 새로운 데이터를 예측

KNN을 통한 피쳐엔지니어링

- 구현이 간단하기 때문에 1차적으로 KNN을 이용하여 분류 결과에 대한 데이터를 얻고, 이를 새로운 피쳐로 전체 데이터에 추가함(피쳐 엔지니어링) → 이 데이터를 다른 분류모델을 만드는데 사용함

- 원래의 예측변수가 두 번 사용되어 다중공선성 문제?

  : KNN으로 추가된 피쳐는 소수의 근접한 데이터로부터 얻는 매우 지엽적인 정보이기 때문에 중복성이 있지 않음

6.2 트리 모델

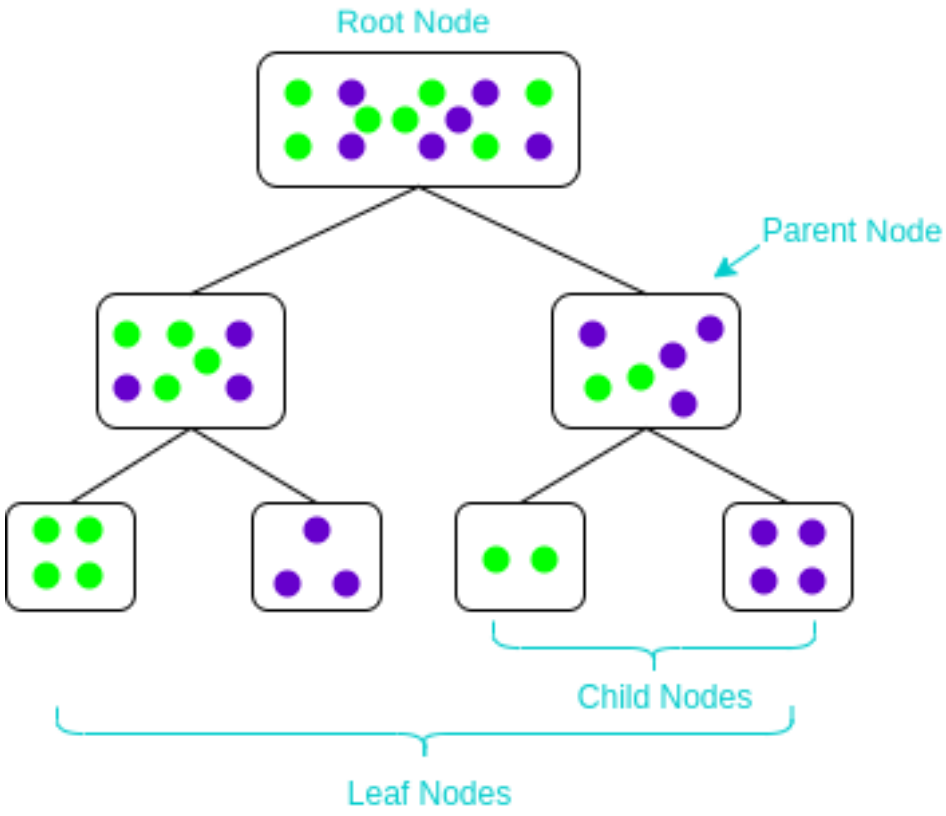
회귀 및 분석 트리(CART:Classification And Regression Tree) = 의사결정트리 = 트리

- 의사결정 규칙을 나무의 뿌리에서 가지와 잎이 뻗어 나가는 형태로 나타낸 머신러닝 모델

- 분류, 예측에 사용

- 쉽게 말하면, 스무고개를 하듯 해당 데이터들의 특성에 대한 질문들을 순차적으로 하면서 분할해나가서 최종적으로 정답 클래스에 분류하는 방법

(효율적으로 분류하기 위해서는 어떤 기준으로, 어떤 순서에 따라 분할해 나갈 지가 중요)



출처: https://www.analyticsvidhya.com/blog/2020/06/4-ways-split-decision-tree/

[트리모델에서 고려할 것]

1. 어떤 기준으로 분할을 할 것인지 = 불순도를 낮출 수 있는 방향으로 분할

- 불순도

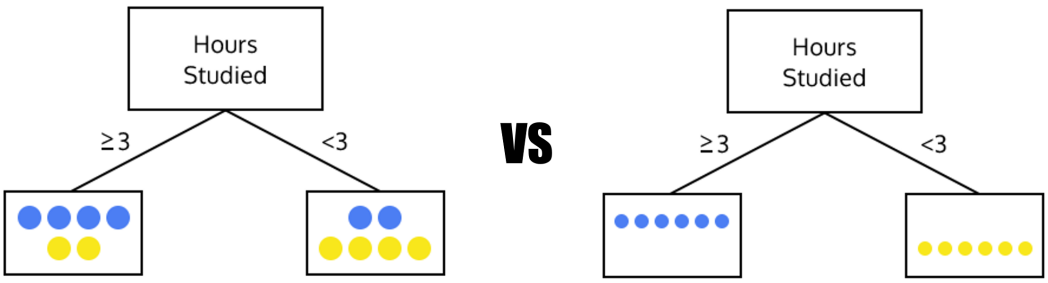
서로 다른 데이터가 섞여 있는 정도를 말함

예)

5개의 데이터 중 A클래스의 데이터가 5개, B 클래스 데이터가 0개 있다면, 데이터의 불순도가 낮음

5개의 데이터 중 A클래스의 데이터가 3개, B 클래스 데이터가 2개 있다면 데이터의 불순도가 높음

불순도 지표: '엔트로피' , '지니불순도' 등



출처: https://hleecaster.com/ml-decision-tree-concept/

- 지니불순도

트리모델에서는 데이터가 섞여 있는 정도의 기준으로 '지니불순도'를 이용함

지니불순도 = 1 - {(A 클래스의 갯수/전체 데이터의 갯수)^2 + (B클래스의 갯수/전체 데이터의 갯수)^2}

예)

5개의 데이터 중 A클래스의 데이터가 5개, B 클래스 데이터가 0개 있다면,

지니불순도 = 1 - {(5/5)^2 + (0)^2} = 0

5개의 데이터 중 A클래스의 데이터가 3개, B 클래스 데이터가 2개 있다면,

지니불순도 = 1 - {(3/5)^2 + (2/5)^2} = 1 - {0.36 + 0.04} = 0.6

2. 어떤 순서로 기준을 적용할지 = Information gain이 큰 순서대로 분할

- Information gain

분류 전 불순도 - 분류 후 불순도 = 불순도가 이전보다 얼마나 감소 했는지 확인 가능

(앞에서 지니불순도 구했기 때문에 information gain 구할 수 있음)

Information gain이 크다 = 불순도를 많이 감소시킨다 = 분류를 잘 한다

[트리 모델 빌딩]

- 재귀분할 알고리즘을 사용

- 위와 같이 어느 기준으로, 어떤 순서대로 나누는것이 좋을지 반복적으로 적용해가면서 최적의 트리 빌딩

- 큰 장점은 만들어진 모델을 시각화 할 수 있어 결과 설명에 용이하다는 것

[트리모델의 한계]

1. 근시안적으로 각 단계에서의 최적의 선택으로만 모델을 만들기 때문에, 전체적인 관점에서의 최적의 트리를 찾지는 못한다. 각 단계에서 오로지 information gain이 가장 큰 순서대로 속성을 분류하기 때문에, 상부에서 데이터를 이렇게 나누었을 때, 하부에 어떤 영향을 미치는지를 고려하지 않는다.

(=해당 단계에서 information gain이 가장 크지는 않더라도, 이후의 분할을 고려했을 때 더 나은 결과를 낼 수 있는 가능성이 있더라도 단순히 현재 information gain만을 기준으로 속성을 분류한다)

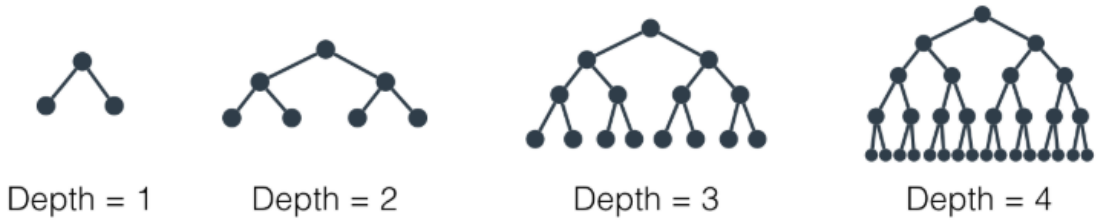
2. 학습데이터의 특성을 모조리 과하게 학습하여, 새로운 미지의 데이터가 입력 되었을 때의 분류 성능이 떨어진다. 즉, 일반화능력이 떨어진다. 따라서 과적합을 방지하기 위한 하이퍼파라미터의 조절이 필요하다.

3. 예측 정확도 면에서는 다중트리 모델인 랜덤포레스트, 부스팅 트리알고리즘이 좋은 성능 보임

[트리모델의 과적합 방지를 위한 방법]

- 학습데이터의 특성을 모두 학습하지 못하도록 제한을 걸어준다.

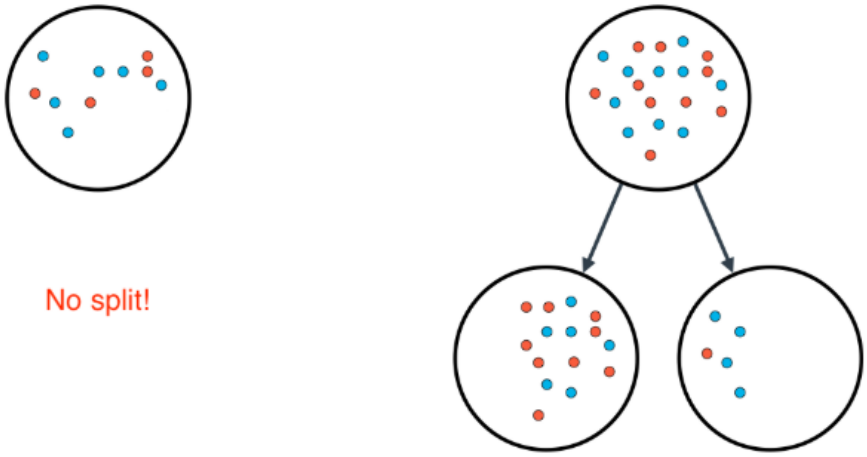
1. max_depth : 노드의 깊이를 제한 (더이상 노드가 나누어 지지 않도록)



Maximum depth of a decision tree

2. min_samples_split : 각 노드에 최소로 있을 수 있는 sample의 갯수 제한

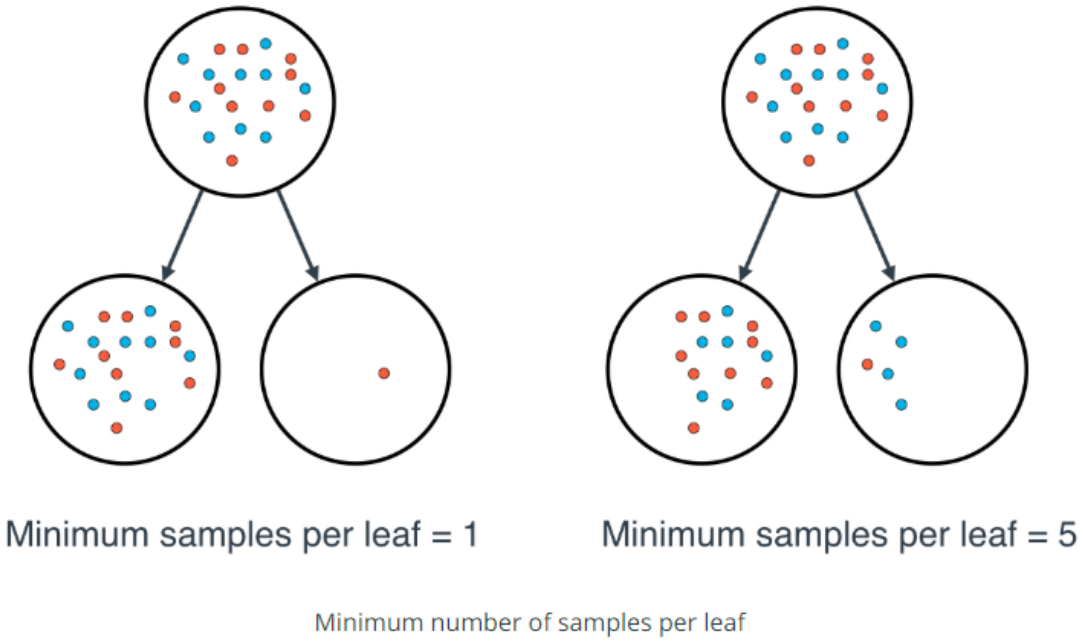
분할을 진행하기 위해서는 노드에 최소 min_sample_split의 샘플이 있어야 함. 노드의 샘플 수가 min_sample_split 샘플보다 적으면 노드가 분할되지 않고 분할 프로세스가 중지됨. 해당 노드를 통해 분할되어 생기는 리프의 샘플 수 와는 무관



Minimum number of samples to split = 11 Minimum number of samples to split = 11

Minimum number of samples to split

3. min_samples_leaf조절 : 마지막 leaf에 최소로 있을 수 있는 sample의 갯수 제한



출처: https://julienbeaulieu.gitbook.io/wiki/sciences/machine-learning/decision-trees

6.3 배깅과 랜덤 포레스트

트리모델의 한계 → **앙상블 기법** 적용 → 더 나은 성능

[앙상블 기법]

- 하나의 데이터로 여러개의 기본모델들을 만들어 서로 결합하여 모델의 정확도를 높이는 방법

- 대표적인 앙상블 기법

  - 배깅: 랜덤포레스트에 적용되는 앙상블 기법
  - 부스팅: 부스팅 트리모델에 적용되는 앙상블 기법

  * Bagging은 여러 모델이 독립적으로 만들어 지는 반면, Boosting은 이전 모델에 의해 다음 모델이 영향을 받으며 순차적으로 만들어진다는 차이가 있음

[Bagging]

* Bootstrap: 샘플링 방법으로, 매 단계에서 원본 데이터의 갯수 만큼 복원 추출을 함

* OOB sample(Out of bag sample)

  Bootstrap에 의하여 뽑히지 않은 샘플

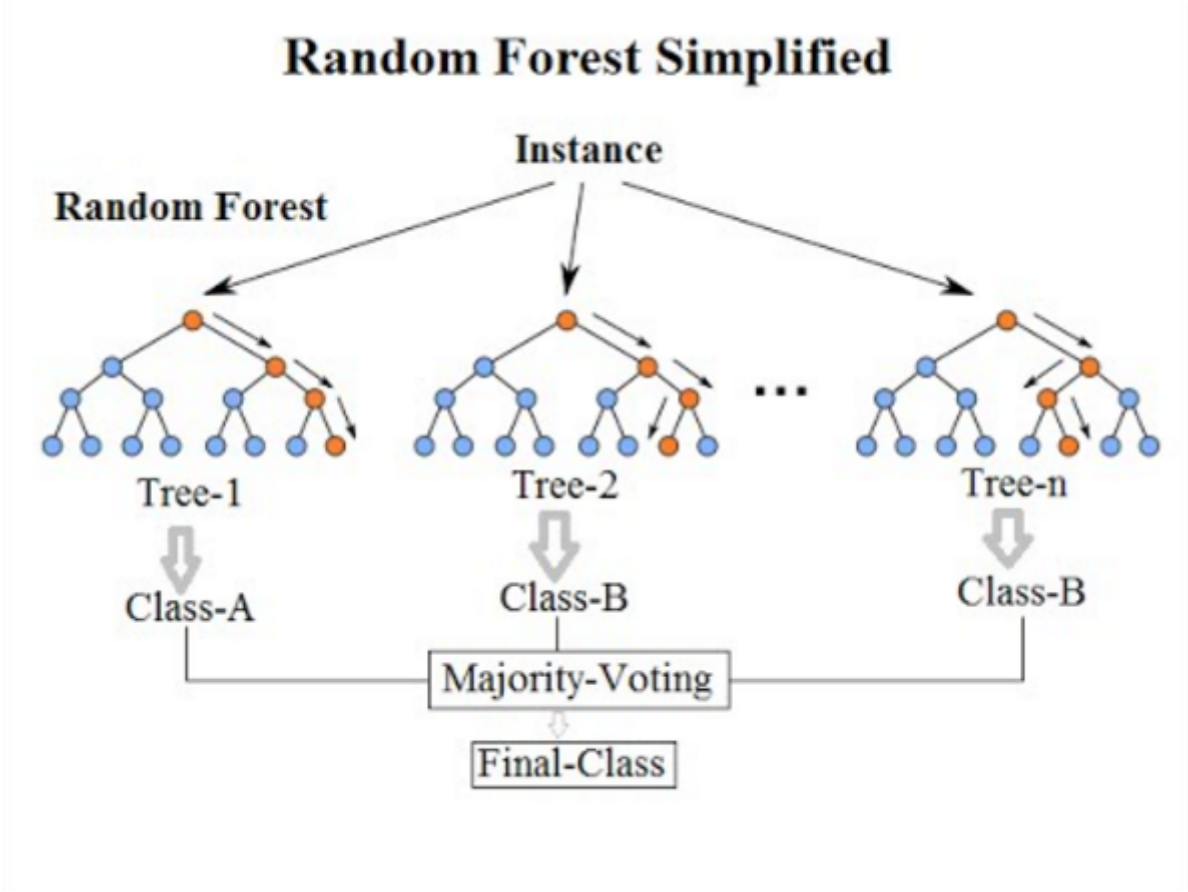
Bootstrap이 복원추출이므로, 중복되는 추출이 얼마나 발생했느냐에 따라 OOB sample이 정해짐

복원추출 과정을 무한히 실행한다고 하였을때, 수학적으로 36.8%에 해당하는 샘플이 복원추출에 사용되지 않음. 이는 각 트리의 학습에 사용되지 않은 데이터이기 때문에, 이를 이용하여 각 트리의 성능을 테스트 하는데에 사용할 수 있음

Bootstrap Aggregating: 전체 데이터에서 샘플을 복원추출하여 여러 모델을 만들고, 최종 단계에서 이 모델들의 예측결과를 집계(aggregate)하여 최종 예측을 결정하는 방법으로 과적합을 방지하기 위함

집계의 방법: 분류모델의 경우 다수결 투표 통한 다수의 예측결과, 회귀모델의 경우 결과들의 평균값

그림예시



출처: By Venkata Jagannath - https://community.tibco.com/wiki/random-forest-template-tibco-spotfirer-wiki-page, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=68995764

[랜덤포레스트]

- 과적합을 일으키는 트리모델의 한계를 배깅을 적용하여 극복하고자 하는 분류모델

- 랜덤성을 이용하여 여러 개의 작은 트리를(기본모델) 만든 뒤, 데이터를 모델에 입력하였을 때, 트리 각각의 분류 예측값들 중 가장 많이 나온 값으로 최종 예측

- 랜덤포레스트가 상대적으로 과적합을 피할 수 있는 이유

  기본모델을 만들 때 적용되는 두 가지 랜덤성

  1. Bootstrap: 전체 데이터에서 랜덤으로 샘플을 복원추출하여 기본모델 학습에 사용
  2. 각 기본모델에 사용되는 feature들 또한 랜덤으로 선택되어 분기를 수행함

     (전체 피쳐가 n이라면, 일반적으로 루트n 으로 피쳐의 갯수를 설정, max_features = auto 가 default)

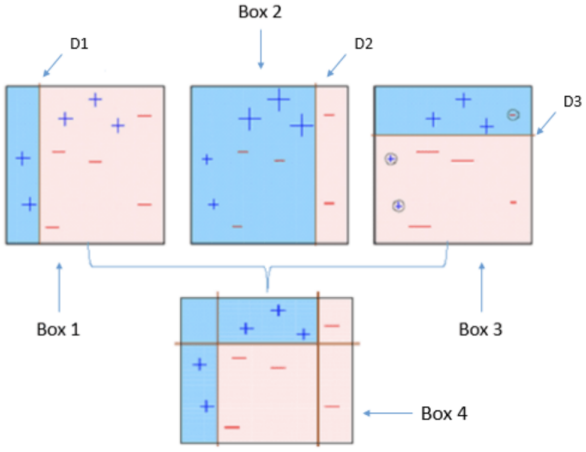
- 단순한 트리모델보다는 예측 정확도가 높지만 시각화를 통한 해석은 불가능

- 여전히 오버피팅의 가능성 존재하므로 최종모델은 교차검증으로 최적의 하이퍼파라미터 찾을 것

6.4 부스팅

[Boosting]

sequential한 weak learner(약한 모델)들을 여러 개 결합하여 예측 혹은 분류 성능을 높이는 알고리즘 (앞의 모델이 맞았는지 틀렸는지에 따라 다른 가중치 부여하며, 틀렸을 때 가중치를 더 높게 주어 다음모델에서는 해당 에러를 더 잘 처리하도록 함)



목표: + 및 - 클래스를 분류하는 것

1. Box 1: 첫 번째 분류기는 D1에 수직선(분할)을 만든다. D1의 왼쪽에 있는 것은 +이고 D1의 오른쪽에 있는 것은 -이다. 그러나 이 분류기는 세 개의 + 점을 잘못 분류한다

2. Box 2: 두 번째 분류기는 세 개의 + 잘못 분류된 점(+의 더 큰 크기 참조)에 더 많은 가중치를 부여하고 D2에 수직선을 만든다. 여기서 D2의 왼쪽은 +이고, 오른쪽은 -이다. 그래도 3개의 점을 잘못 분류하였다.

3. Box 3: 다시, 세 번째 분류기는 잘못 분류된 세 지점에 더 많은 가중치를 부여하고 D3에 수평선을 만든다. 그러나 이 분류기는 원의 점을 올바르게 분류하지 못한다.

4. Box 4: weak learner(박스 1, 2, 3)의 가중치 조합한다. 모든 데이터를 정확하게 분류하였다.

이처럼 부스팅 알고리즘의 기본 아이디어는 순차적으로 약한 모델을 만들고, 다양한 기능의 중요성과 매개 변수에 대한 결론을 내린 다음, 이 결론을 사용하여 새롭고 강력한 모델을 만들어 이전 모델의 잘못된 분류 오류를 줄이는 것

[XGBoost]

- 가장 많이 사용되는 부스팅 트리모델

- 이전의 트리모델에서 오분류된 샘플에 더 많은 가중치를 두고 연속적으로 다음 트리를 만들어나가는 방식

- 일반적인 Gradient Boosting Model은 순차적으로 Weak learner가 가중치를 증감하는 방법으로 학습키기때문에 전반적으로 속도가 느림 → XGBoost는 병렬 학습으로 GBM에 비해 빠른 수행성능

- Regularization(정규화)의 도입으로 GBM의 과적합 문제 보완

    * 정규화: 모델의 복잡도가 증가할수록 손실함수에 패널티를 주는 방식으로 과적합 방지

    * 정규화 파라미터인 alpha(L1 정규화,맨해튼 거리), lambda(L2 정규화,유클리드 거리) 크게 하면, 모델이 복잡해질수록 많은 패널티 주어 결과적으로 얻는 트리의 크기가 작아짐

- XGBoost에는 다양한 하이퍼파라미터가 존재하는데, 이를 잘 사용하면 부스팅 모델 중 최고의 성능 발휘 가능