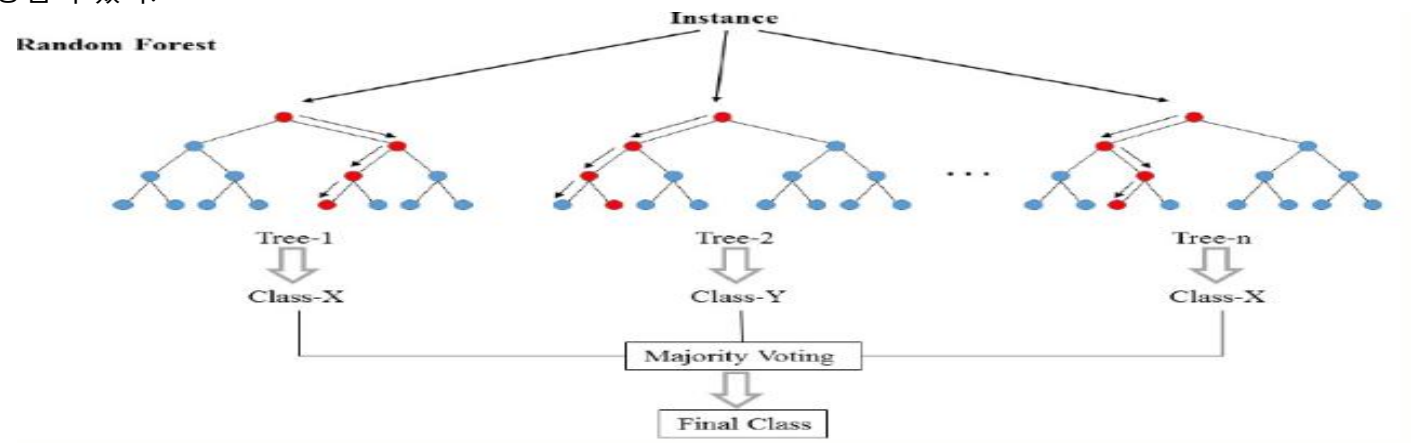
3. 결정트리의 앙상블 (Ensemble)

- '앙상블' 이란 여러 머신러닝 모델을 연결하여 더 강력한 모델을 만드는 기법
- 여러 종류의 앙상블 모델 중, 분류와 회귀 문제의 다양한 데이터 셋에 효과적이라고 입증된 랜덤 포레스트(Random Forest) 그레디언트 부스팅(Gradient Boosting)는 결정트리를 기본 요소로 사용한다.

1) 랜덤포레스트

- 결정 트리의 과대적합을 회피하기 위한 방법 중 하나.
- 기본적으로 조금씩 다른 여러 결정 트리의 묶음이라고 할 수 있다.
- 잘 작동하되 서로 다른 방향으로 과대적합된 트리를 만들고 그 결과를 평균냄으로써 과대적합된 양을 줄여 예측 성능을 유지하며, 과대적합을 줄이는 방법.
- 결정 트리를 많이 만들어야 하고, 각 트리는 예측을 잘해야하고 다른 트리와 구별이 되어야 한다. (트리 생성 시 무작위성을 주입)
- 트리 생성 시, 데이터 포인트를 무작위로 선택하는 방법 또는 분할 테스트에서 특성을 무작위로 선택하는 방법이 있다.



Algorithm Study

3. 결정트리의 앙상블 (Ensemble)
2) 랜덤 포레스트 구축

- N_estimators 변수를 통해 생성할 트리의 개수 선정.

- 트리들은 완전히 독립적으로 만들어져야 하므로 알고리즘은 각 트리가 교유하게 만들어지도록 무작위 선택.

- 데이터의 부스트랩 샘플 생성
  (n_samples 개의 데이터 포인트 중에서 무작위로 데이터를 n_samples 횟수만큼 반복 추출)
  ->한 샘플이 여러 번 중복, 누락 추출 가능
  ->부스트랩 샘플링은 트리가 조금씩 다른 데이터셋을 이용해 만들어지도록 하고 각 노드에서
  특성의 일부만 이용.

- Max_features를 통해 특성의 개수 설정(후보 특성)
  -> n_features로 설정하면 모든 특성을 고려하므로 특성 선택의 무작위성이 들어가지 않는다.
  -> 값을 크게하면 트리들은 비슷해지고 데이터에 잘 맞춰질 것이고
  낮추면 트리들은 달라지고 각 트리는 데이터에 맞추기 위해 깊이가 깊어지게 된다.

- 랜덤 포레스트의 예측은 먼저 알고리즘이 모델에 있는 모든 트리의 예측을 만든다.
  ->회귀의 경우 이 예측을 평균하여 최종 예측을 만든다.
  ->분류의 경우 약한 투표 전략을 사용한다.
  (각 트리의 예측 확률을 평균내어 가장 높은 확률을 가진 클래스가 예측값이 된다.)

- Random_state를 지정해 주지 않으면 트리가 적을 수록 변동이 커진다.

- n_jobs를 통해 사용 코어 수를 지정하여 속도 향상이 가능하지만 텍스트 데이터 같이 매우 차원이 높고 희소한 데이터에는 잘 작동하지 않아 이러한 경우 선형 모델이 더 적합하다.
  또한 속도와 메모리 사용에 제약이 있는 경우도 마찬가지이다.

- 결정트리와 마찬가지로 max_depth로 사전 가지치기 옵션이 포함되어있다.

Algorithm Study

3. 결정트리의 앙상블 (Ensemble)
3) 그레디언트 부스팅 회귀 트리

- 이전 트리의 오차를 보완하는 방식으로 순차적으로 트리 생성.
  (무작위성 X) -> 강력한 사전 가지치기 사용

- 보통 하나에 다섯 정도의 깊지 않은 트리를 사용 (약한 학습기)
  -> 메모리를 적게 사용, 예측 빠름.
  -> 각 트리는 데이터 일부에 대해서만 예측을 잘 수행 가능하여 트리가 많이 추가될 수록 성능향상

- Learning_rate를 통해서 이전 트리의 오차를 얼마나 강하게 보정할 것인지를 설정
  -> 학습률이 크면 트리는 보정을 강하게 하여 복잡한 모델 생성

- 랜덤 포레스트와 마찬가지로 n_estimator로 트리의 개수를 설정하여 모델을 복잡, 간단하게 설정 가능
  -> 커질수록 모델의 복잡도가 커지고 훈련 세트의 보정 기회가 많아짐.

- 보통 더 안정적인 랜덤포레스트를 이용하지만, 시간이 중요하거나 머신러닝 모델에서 마지막 성능까지 쥐어짜야 하는 경우 그레디언트 부스팅을 이용하면 도움이 된다.

- 대규모 머신 러닝 문제에 그레디언트 부스팅을 적용하려면 xgboost패키지, 파이썬 인터페이스 검토가 도움이 된다.

- 랜덤포레스트와 마찬가지로 n_estimator와 max_depth를 이용하여 트리 개수, 트리 깊이를 설정한다.

- Learning_rate를 통해 오차를 보정하는 정도를 조절한다.
  -> scikit-learn 0.20버전에서는 n_iter_no_cahnage와 validation_fraction이 추가되어
  훈련 데이터에서 validation_fraction 비율만큼 검증 데이터로 이용하여 n_iter_no_change 반복동안 검증 점수가 향상되지 않으면 훈련이 종료된다.

Algorithm Study

3. 결정트리의 앙상블 (Ensemble)
4) 배깅

- Bootstrap aggregating의 줄임말.
  -> 중복을 허용한 랜덤 샘플링으로 만든 훈련 세트를 사용하여 분류기를 각기 다르게 학습.
  (랜덤포레스트의 특징과 같다)

- 분류기가 predict_proba() 매서드를 지원하는 경우 확률값을 평균하여 예측을 수행하고,
  그렇지 않은 경우 빈도가 가장 높은 클래스 레이블이 예측 결과가 된다.

- 결정트리에 배깅을 수행하는 것이 랜덤포레스트와 동일하다고 간주한다.

- Oob_score를 true로 설정하면 매개변수는 부트스트랩에 포함되지 않은 샘플을 기반으로 훈련된 모델을 평가한다.
  -> OOB 오차라고도 하는 이값을 통해 테스트 세트의 성능을 짐작가능.

- 배깅은 랜덤 포레스트와 달리 max_samples 매개변수에서 부트스트랩 샘플의 크기를 지정할 수 있다.

- 추가적으로 랜덤포레스트에서는 DescionTreeClassifier(splitter="best")를 사용하도록 고정되어 있지만, splitter="random"으로 설정하여 무작위로 분할한 후보 노드 중, 최선의 분할을 탐색 가능하다.

Algorithm Study

3. 결정트리의 앙상블 (Ensemble)

5) 엑스트라 트리

- 엑스트라 트리는 랜덤포레스트와 비슷하지만, 후보 특성을 무작위로 분할하고 최적 분할을 찾는다.
- 랜덤포레스트와 달리 splitter='random'을 이용하고 부트스트랩 샘플링은 적용하지 않는다.
  -> 랜덤포레스트와 다른 방식으로 모델에 무작위성을 주입
- 예측 방식은 랜덤 포레스트와 동일하게 각 트리가 만든 확률값의 평균을 구한다.
- 엑스트라 트리와 랜덤 포레스트는 거의 같은 성능을 내지만, 엑스트라 트리가 계산 비용이 비교적 적다.
  하지만, 무작위 분할 때문에 일반화 성능을 높이려면 많은 트리를 생성해야 되서 일반적으로
  랜덤포레스트가 선호된다.

6) 에이다 부스트

- Adaptive Boosting의 줄임말.
- 그레디언트 부스팅과 달리 이전의 모델이 잘못 분류한 샘플에 가중치를 높여 다음 모델을 훈련시킨다.
  (성능에 따라 가중치 부여)
- 예측한 레이블을 기준으로 모델의 가중치를 합산 후, 가장 높은 값을 가진 레이블 선택.
- 기본값으로 max_depth=1(분류), max_depth=3(회귀)를 이용하지만 base_estimator 매개변수에서 다른
  모델을 지정 가능하다.
  (기본값은 다른 앙상블 모델에 비해 단순하지만, 일반화 성능이 좋다.)
- 그레디언트 부스팅과 마찬가지로 순차적으로 학습하여 n_jobs 지원하지 않는다.