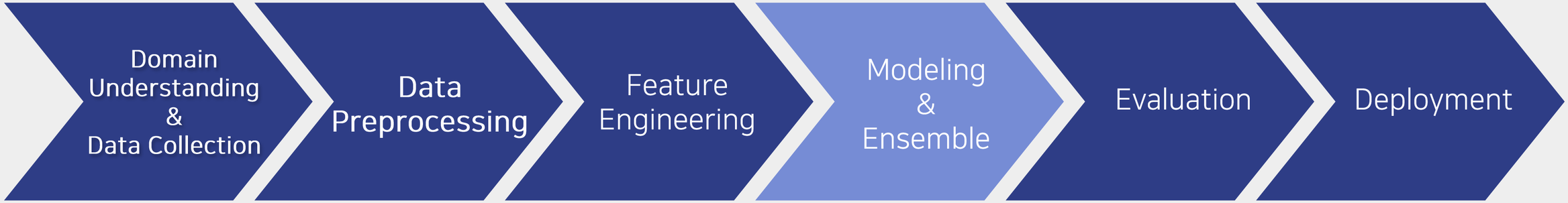
Bagging & Boosting

ML Session 6차시

CONTENTS.

Ensemble



Ensemble

여러 개의 모델(혹은 예측)을 결합하여 보다 정확한 예측을 수행하게 만드는 방법

대표적으로 Bagging, Boosting, Voting, Stacking 등이 있음

Ensemble

Ensemble 기법들 특징 요약

배깅(Bagging)	학습 데이터의 하나로 여러 개의 서브셋(subset) 데이터를 생성하여 각 서브셋으로 여러 모델 학습 학습 시 사용되는 알고리즘은 모두 같음 → 학습된 여러 모델들의 결과를 결합하여 예측 생성
부스팅(Boosting)	성능이 낮은 모델을 연속적으로 학습 → 이전 모델에서 오분류된 샘플에 가중치를 부여하여 성능 향상
보팅(Voting)	동일 데이터로 서로 다른 알고리즘의 여러 모델을 학습시켜 예측한 뒤 이를 결합 → "투표" 형태로 결합 투표 방식에 따라 하드 보팅(Hard Voting)과 소프트 보팅(Soft Voting)으로 나눔
스태킹(Stacking)	여러 다른 모델의 예측 결과를 새로운 모델('메타 모델')의 입력으로 사용하여 학습

2. Bagging

Bagging이란?

Bagging (Bootstrap Aggregating)

훈련 데이터에 복원 추출(Bootstrap)을 반복하여 얻은 서브셋(subset) 데이터로 모델 학습

이후 각 서브셋의 학습된 모델들이 각자 예측을 수행하면, **예측의 결과를 결합하여 최종 예측 출력**

서브셋을 학습하는 모델은 모두 같은 알고리즘을 사용함 (Decision Tree 등)

일반적으로 **분류 문제는** Voting, **회귀 문제는** 평균값으로 예측 결과를 결합함

Bagging이란?

Bootstrap

- Bagging에서 사용하는 sub-sampling 방법
- 훈련 데이터를 복원 추출하여 서브셋을 생성함 (test set에는 적용 X)
- 일반적으로 서브셋의 사이즈는 원본 데이터의 사이즈와 동일함
- 서브셋의 개수는 하이퍼파라미터로, 사용한 모델의 데이터 민감도와 모델 복잡도에 따라 결정 (보통 2~30개, 최대 50개)

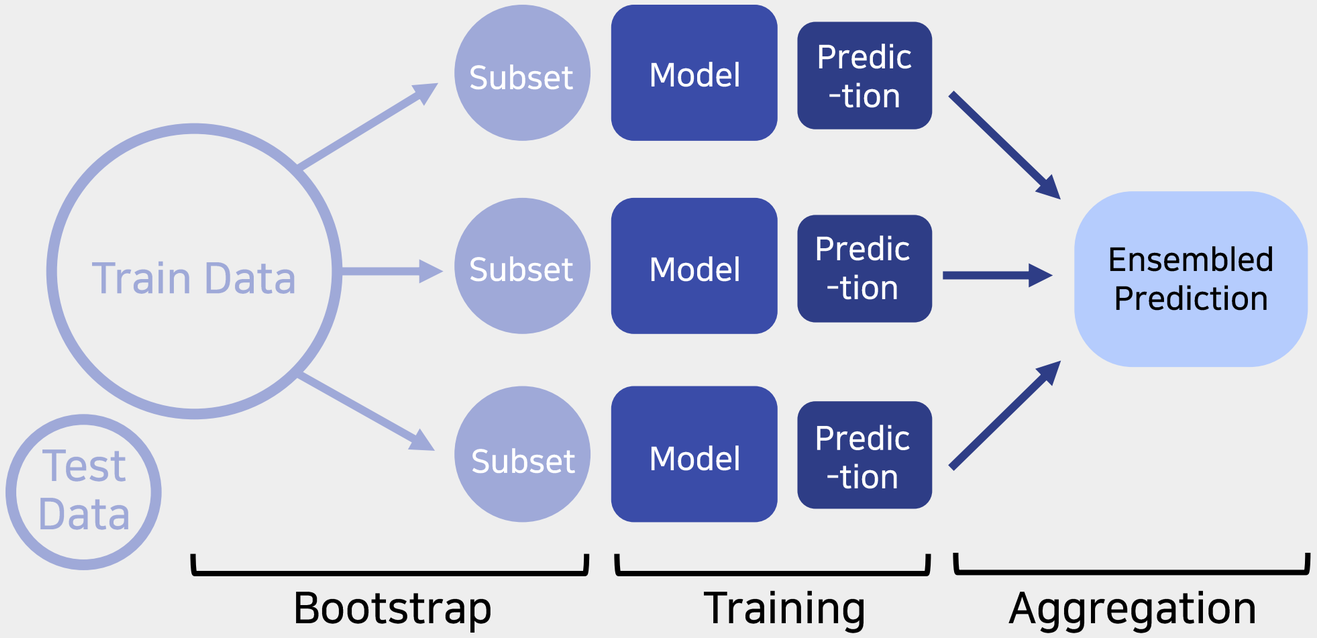
- 기본적으로 복원 추출이기 때문에 선택되지 않은 샘플(out-of-bag) 존재

  → oob 샘플을 활용하여 각 모델의 에러율을 구하고 이를 평균내어 앙상블 평가

2. Bagging

Bagging이란?

전체 매커니즘



Bagging의 특징

- **장점**
  - 개별 예측이 불안정할 때 배깅을 통해 전체 모델의 분산(variance) 감소
  - 병렬 연산이 가능하여 연산 속도 ↑
- **한계점**
  - 복원 추출을 수행하므로 여러 서브셋 간에 유사성이 높아질 수 있음
  - 상기 이유로 인해 여러 결정 트리가 비슷한 구조를 가지게 될 수 있음

  → 전체 모델의 다양성 저하, 과적합 우려

2. Bagging

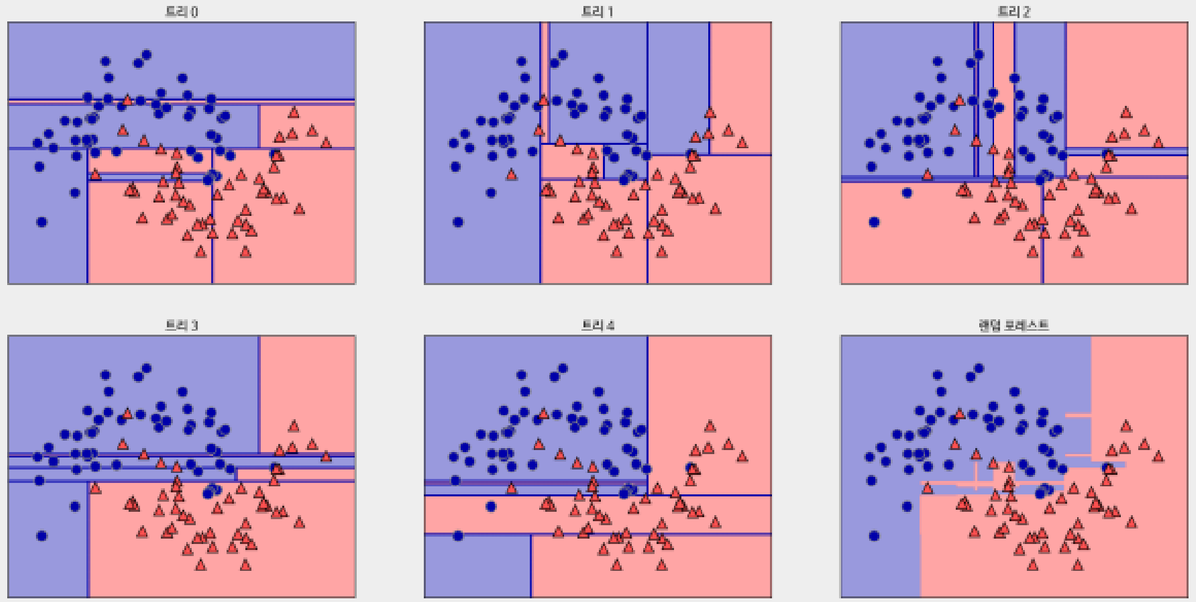
RandomForest

RandomForest

Bagging 기법을 활용한 대표적인 앙상블 모델

병렬 처리가 가능하여 큰 데이터셋에서도 비교적 잘 작동함

고차원의 sparse한 데이터는 잘 작동하지 않음



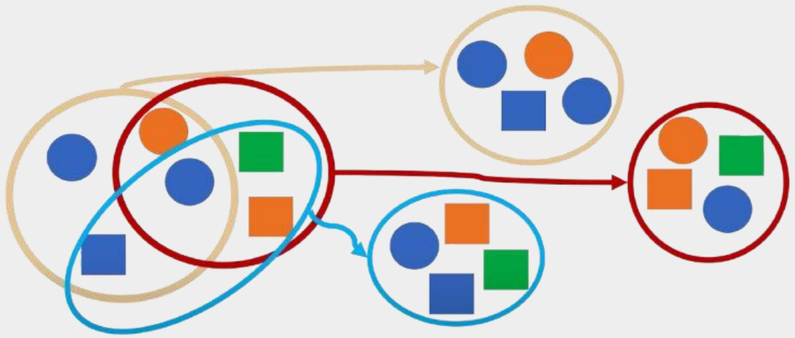
RandomForest

Bagging의 한계점 개선

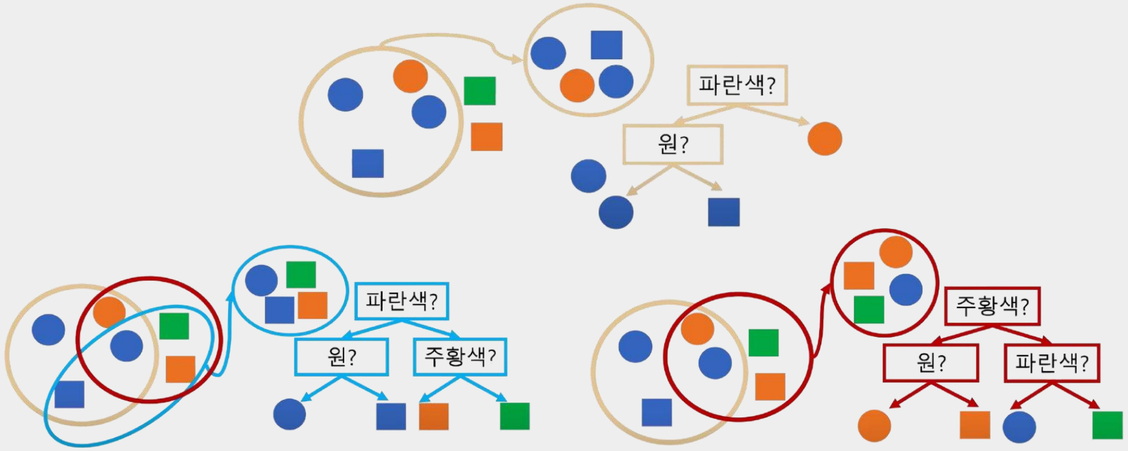
일반 DT 모델을 사용하는 Bagging은 node를 분리할 때 모든 feature를 고려하여 에러를 계산하였음

이로 인해 데이터에 영향을 미치는 변수를 중심으로 node가 분리되는 경향 → 여러 DT의 유사성이 높아 과적합 우려

RandomForest의 경우 각 DT가 매 분할마다 전체 feature 중 일부만을 무작위로 선택하여 분할 진행



데이터 랜덤 추출



질문 랜덤 추출

2. Bagging

RandomForest

RandomForest 구조

① 학습 데이터에 Bootstrap을 반복하여 여러 서브셋 생성

② 각 서브셋에 대해 개별 Decision Tree 학습

- Decision Tree의 각 분할 때마다 모델의 전체 feature 중 일부(N개)를 무작위로 골라 분할 탐색

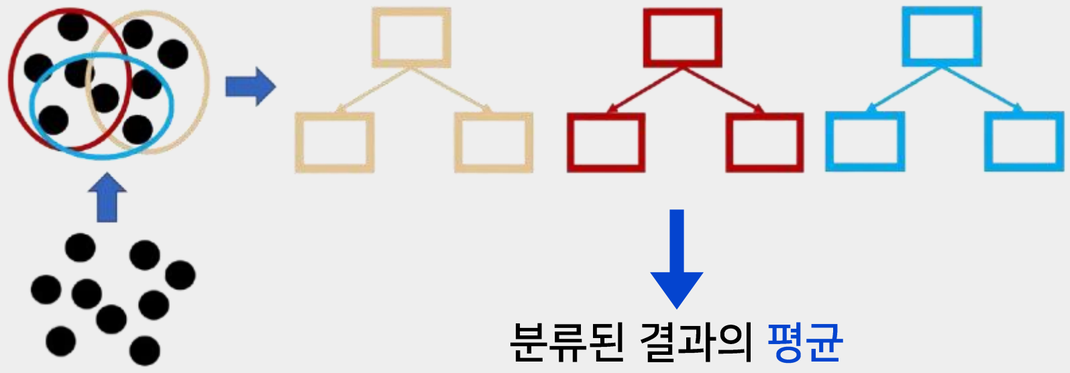
  → 선택된 N개의 feature만을 고려하여 정보 이득이 가장 높은 최적의 분할을 찾는 것

- 선택되는 feature의 개수가 적을수록 각 트리의 유사성은 낮아지고, 깊이가 깊어짐

③ 학습된 모든 개별 Decision Tree의 예측을 수행

④ 예측을 결합하여 최종 예측 생성

- 회귀 - 예측값의 평균 / 분류 - 소프트 보팅(Soft Voting)



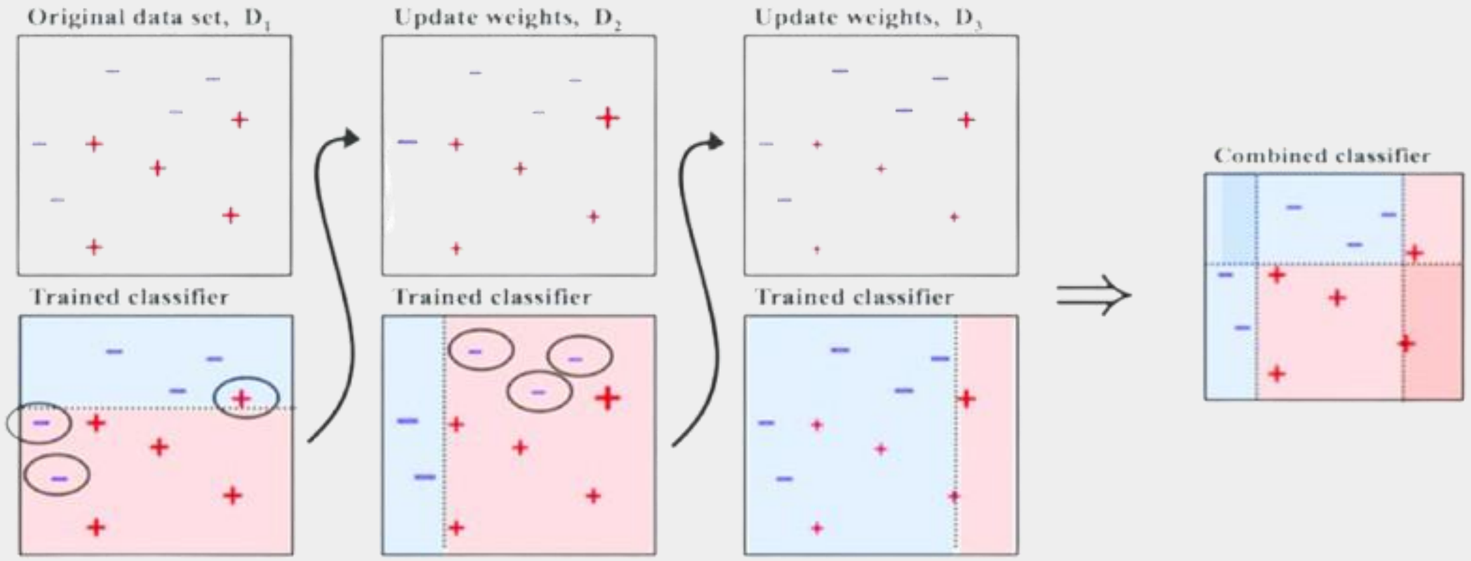
Boosting이란?

Boosting

약한 학습기(weak learner)를 **연속적으로 연결 및 학습시켜 이전 모델의 예측 결과가 다음 모델에 영향을 주게 함**

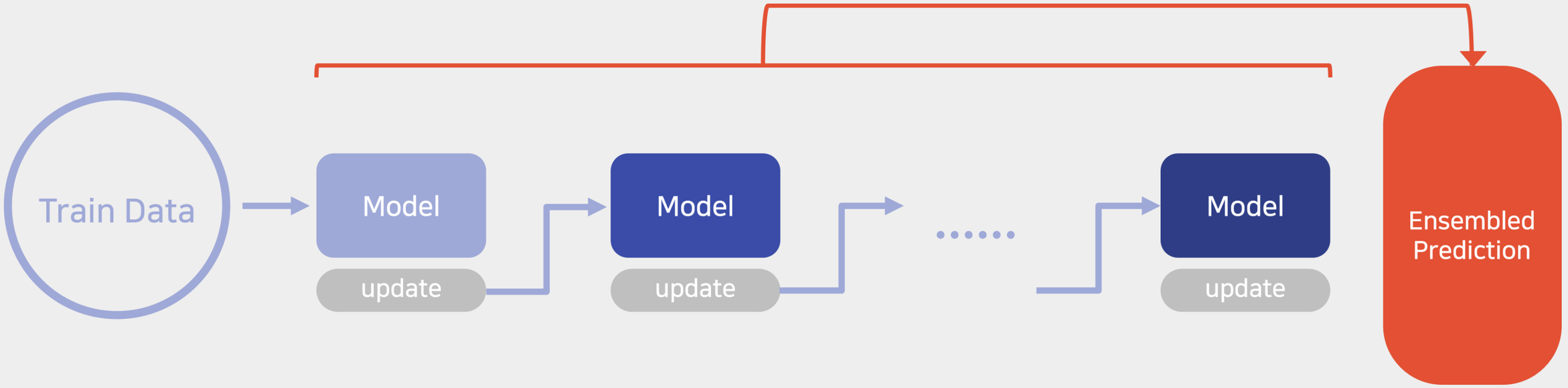
이전 모델의 예측에서 **잘못 예측된 데이터에 가중치를 줘서 오류를 개선하는 방식**으로 학습

→ 오분류된 데이터를 다음 모델에서 어떻게 반영하는지가 Boosting 계열 앙상블 모델의 차이를 결정



Boosting이란?

기본 매커니즘



Boosting이란?

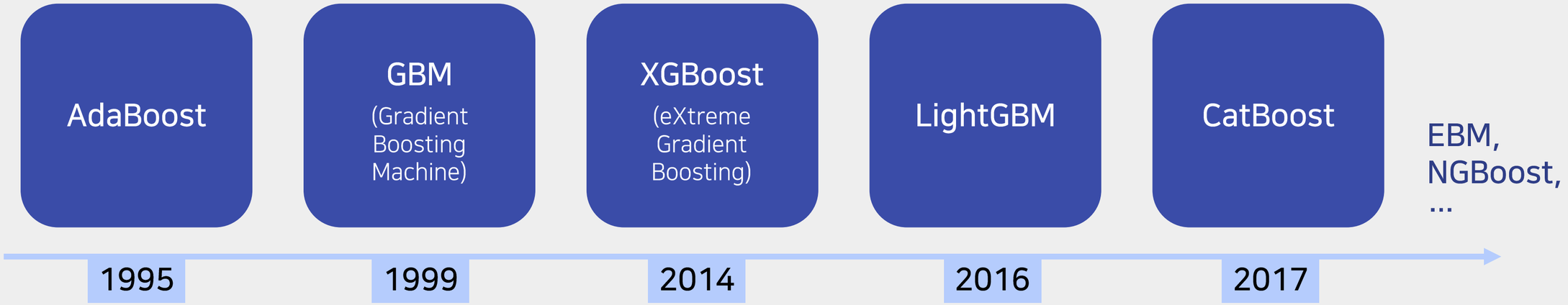
Boosting 모델의 특징

순차적(sequential)으로 앞의 모델을 보완하는 형태로 학습하므로 병렬 처리 불가 → 속도 ↓

오답을 더 잘 분류할 수 있게 학습되므로 정확도가 비교적 높지만, 노이즈나 outlier에 취약(민감)

상기 이유로 새로운 관측데이터(unseen data)에 대해 예측 성능이 낮을 수도 있고, 과적합 가능성 有

→ learning rate 조정, Early Stopping 사용 등의 방법으로 과적합 방지



3. Boosting

AdaBoost

AdaBoost

가장 기본적인 부스팅 모델. 오분류된 데이터에 더 큰 가중치 줘서 다음 라운드 학습에 반영하는 방식

처음에는 모든 데이터에 동일한 가중치를 주어 학습시킨 뒤, 오분류된 데이터에 가중치를 줌

→ 제대로 분류하지 못한 데이터, 즉 과소적합된 샘플에 집중하여 학습함으로써 성능 높임

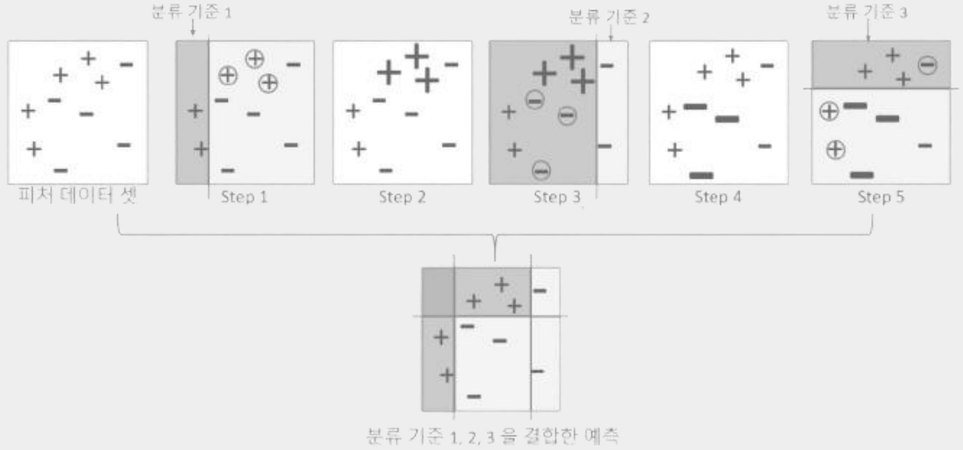
학습 과정

① 첫 모델은 모둔 훈련 데이터에 동일한 가중치 $w^{\mathrm{i}} = \frac{1}{m}$ 부여한 뒤 학습

② 학습 후 모델의 에러율 r_j 계산 $r_j = \frac{\sum_{i=1}^{m} \hat{y}_j^i \neq y\mathrm{i}^{w^{\mathrm{i}}}}{\sum_{i=1}^{m} w^i}$

③ 가중치 α_j 계산 및 부여 후 가중치를 $\sum_{j=1}^{m} w^{\mathrm{i}}$로 나누어 정규화 수행

$$\alpha_j = \eta \log \frac{1 - r_j}{r_j} \qquad w^i \begin{cases} w^i & if\ \hat{y}_j^i = y^i \\ w^i e^{\alpha_j} & if\ \hat{y}_j^i \neq y^i \end{cases}$$



④ 위의 과정 계속해서 반복 → 예측 계산 후 모델들의 가중치 합산하여 최종 예측 결과 도출 $\hat{y} = \underset{k}{\mathrm{argmax}} \sum_{j=1,\, \hat{y}_j(x)=k}^{N} \alpha_j$

GBM

GBM

경사 하강법(Gradient Descent)을 활용하여 이전 모델의 잔차(잔여 오차)를 최소화 하는 방향으로 학습

→ 잔차를 과도하게 학습하면 과대적합 우려 有

학습 과정

① 초기 예측값 설정 → 회귀는 평균값, 분류는 로그 오즈비(log odd ratio) 사용

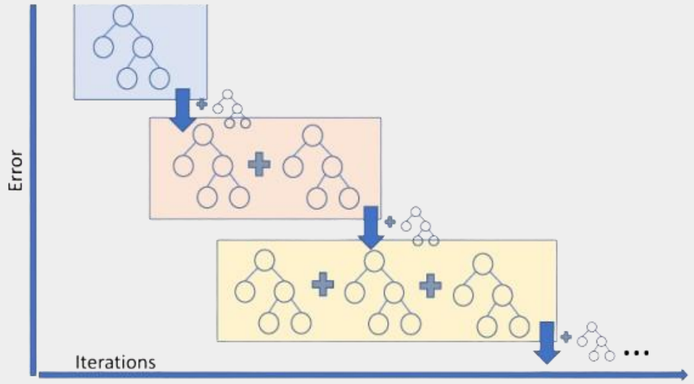
② 매 반복 스텝마다 현재 모델의 예측과 실제 값 사이의 오차(=잔차, residual) 계산

③ 현재의 잔차에 해당하는 손실 함수의 gradient 계산

④ gradient를 target으로 두고 약한 학습기(≠현재 모델) 학습 → 예측 수행

⑤ 예측 결과를 현재 모델에 추가

- 학습률(learning rate) 파라미터를 사용하여 얼마나 예측을 업데이트 할 것인지 결정
- 결론적으로 GBM은 매 반복 스텝마다 해당 스텝에서의 학습 결과를 누적하여 합하는 형태



XGBoost

XGBoost (eXtreme Gradient Boost)

GBM을 발전시켜 이전에 비해 속도와 성능을 향상시킨 모델

→ 정규화, 누락된 데이터 처리, 병렬 처리, 교차 검증(CV) 및 조기 종료, 가지치기 등

정규화	목적함수에 L1, L2 정규화 항(regularization term)을 추가하여 과적합 방지
병렬 처리	최적 분할 지점을 찾을 때, 모든 데이터를 특징(feature)에 대하여 정렬한 뒤 병렬로 최적의 분할 지점 계산 및 선택
결측값 처리	누락된 데이터 (결측값) 자동으로 한 쪽 노드로 보내서 처리
조기종료	성능이 개선되지 않는 경우에 조기 종료 수행하여 과적합 방지
가지 치기	트리를 확장한 뒤, 가치가 없는 노드를 제거하여 과적합 방지하고 모델의 복잡도 ↓

LGBM

LGBM (LightGBM)

XGB에 비해 빠르고 가벼운 모델 BUT 데이터셋이 작으면 (< 10000개) 과적합 우려

최적화 1: GOSS

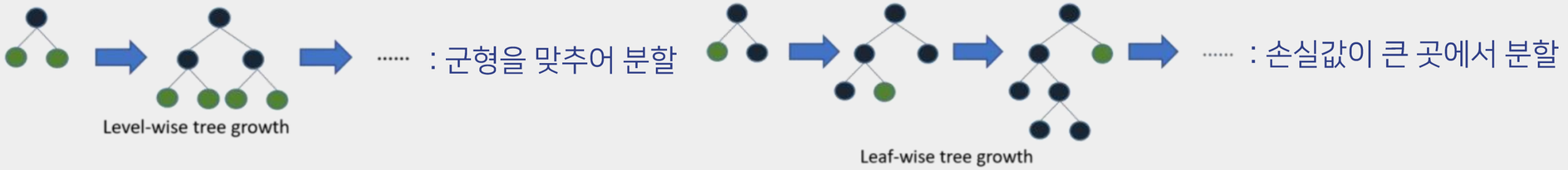
훈련이 잘 되지 않아 큰 gradient를 가지는 데이터는 보존하고, 작은 gradient를 가지는 데이터는 일부만을 무작위 추출하여 학습 → 오분류된 데이터에 더 큰 중요도 부여하여 학습

최적화 2: EFB

feature 간의 상호 배타적인 관계를 활용하여 feature를 묶음 단위로 합침
(one-hot feature는 하나의 1을 제외하고 나머지는 0의 값을 가지므로 이를 합쳐서 효율적으로 처리하는 것)

Leaf-wise

대부분의 기존 알고리즘(RF, XGB 등)은 트리를 수평으로 확장하는 depth-wise 방식 사용
leaf-wise 알고리즘을 사용하여 loss를 줄이는 데 집중, 속도 ↑ BUT 그만큼 과적합에 민감해짐



CatBoost

CatBoost (Unbiased boosting with Categorical features)

범주형(categorical) 변수가 많을 때 효과적 → 범주형 변수를 평균 인코딩 기법으로 알아서 변환하여 학습

Level-wise

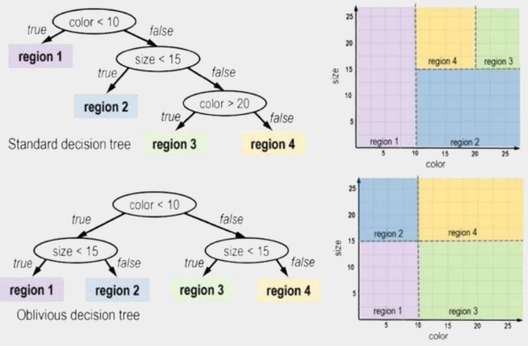
트리 생성시 level-wise 방식으로 확장

Ordered-Boosting

기존 부스팅 모델이 모든 훈련 데이터를 대상으로 잔차를 계산했던 것과 달리,
학습 데이터의 일부만으로 잔차 계산을한 뒤 다음 라운드로 넘어가는 방식
즉, 학습에 사용하는 데이터를 순서대로 차츰 늘려가며 Boosting을 진행
→ 이때 데이터를 샘플링하여 랜덤하게 가져옴 (Random Permutation)

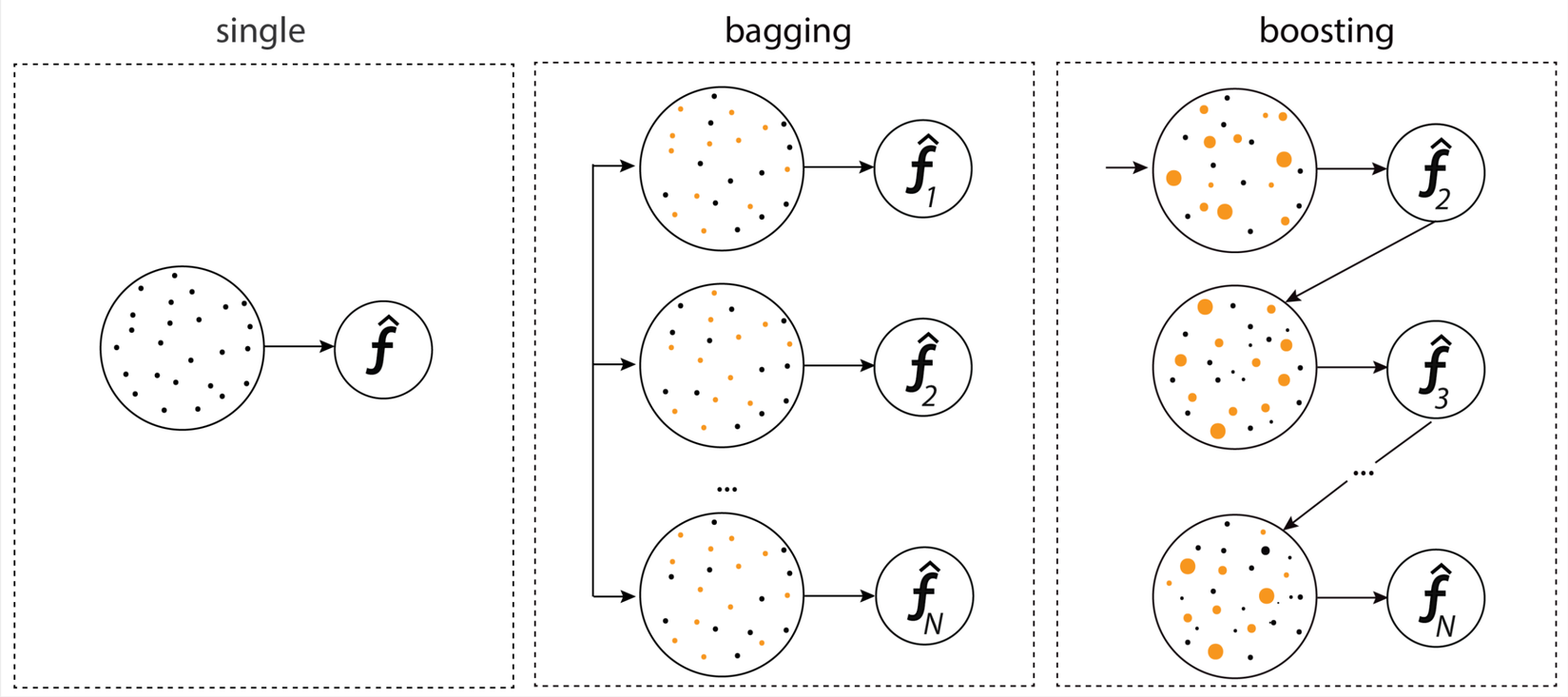
Symmetric Trees

모든 분기점에서 가능한 모든 분할의 경우를 평가하는 것이 아니라,
대칭적인 트리 구조를 사용하여 같은 level에서 동일한 분할 기준 사용
속도 ↑, 데이터 민감도 ↓



Bagging과 Boosting의 차이점

Bagging과 Boosting의 차이점



REFERENCE

https://bkshin.tistory.com/entry/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D-11-%EC%95%99%EC%83%81%EB%B8%94-%ED%95%99%EC%8A%B5-Ensemble-Learning-%EB%B0%B0%EA%B9%85Bagging%EA%B3%BC-%EB%B6%80%EC%8A%A4%ED%8C%85Boosting

https://brunch.co.kr/@chris-song/98

https://swalloow.github.io/bagging-boosting/

https://ekdud7667.tistory.com/entry/Ensemble-%EA%B0%9C%EC%9A%94Bagging-Boosting-Stacking

https://hyunlee103.tistory.com/25

THANK YOU

ML Session 6차시