17기 분석 base세션 4주차

Boosting, Stacking

16기 분석 이보경

Ensemble

앙상블 학습이란?

≫ 여러 개의 모델을 생성하고 그 예측을 결합함으로써 보다 정확한 최종 예측을 도출하는 기법

앙상블 학습의 종류

≫ 보팅 (Voting)

≫ 배깅 (Bagging)

≫ 부스팅 (Boosting)

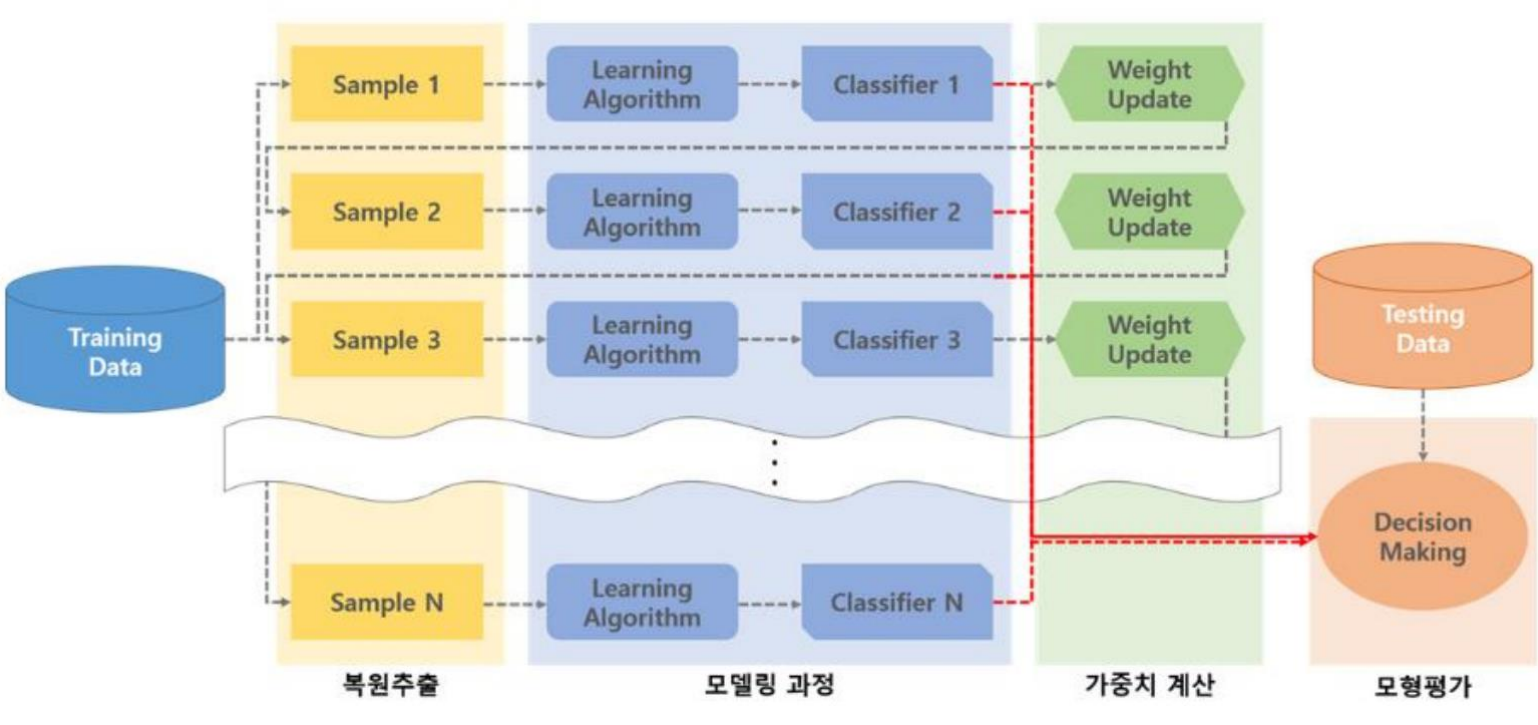
≫ 스태킹 (Stacking)

Boosting

기본 Boosting 개요

≫ 여러 개의 모델이 **순차적으로** 학습-예측을 수행

≫ 앞에서 학습한 모델의 잘못 예측한 데이터에 **가중치(weight)를 부여**해 오류를 개선해 나가며 학습-예측 수행

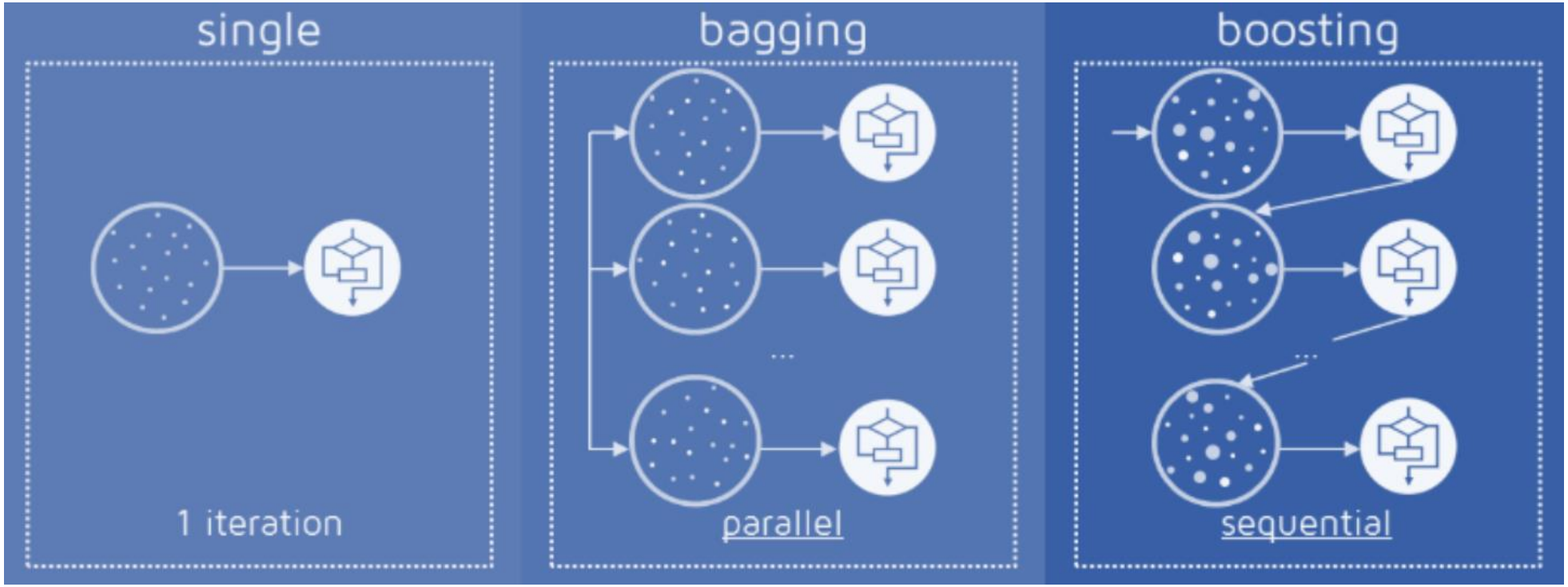


Boosting

Bagging(배깅) vs Boosting(부스팅)

≫ Boosting도 Bagging과 동일하게 복원 랜덤 샘플링을 하지만, **가중치**를 부여한다는 차이점 있음

≫ 즉, Boosting은 Bagging과 다르게 **순차적 방법**을 통해 이전 학습의 결과가 다음 학습의 결과에 영향



Boosting

장점

≫ Error에 대해 가중치를 주어 학습하기 때문에 정확도가 높음

단점

≫ 순차적으로 진행되기 때문에 속도가 느림

≫ Overfitting(과적합)의 가능성이 높음

종류

≫ 대표적으로 AdaBoost, Gradient Boost, XGBoost(eXtra Gradient Boost), LightGBM 등이 있음

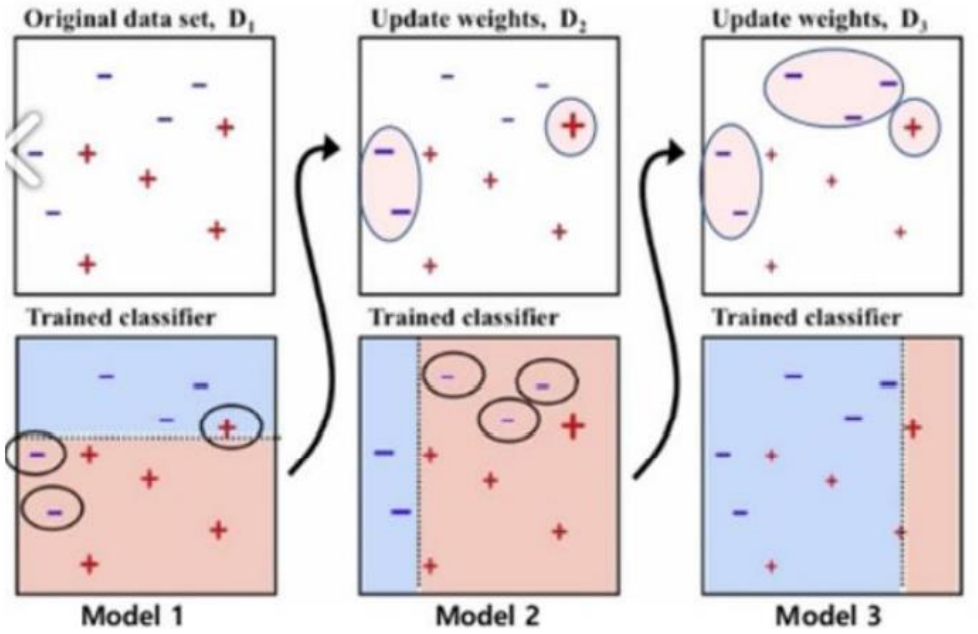
AdaBoost

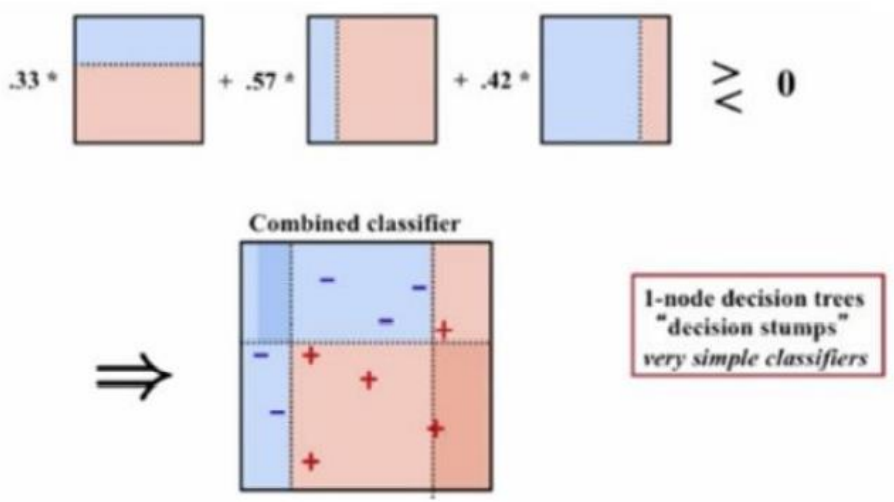
AdaBoost (Adaptive Boosting)

≫ 간단한 weak learner들이 상화보완 하도록 순차적으로 학습

≫ 이들을 조합하여 성능을 증폭

≫ 이전 weak learner가 잘못 예측한 데이터에 높은 가중치를 부여하여 더 집중하여 학습-예측





AdaBoost

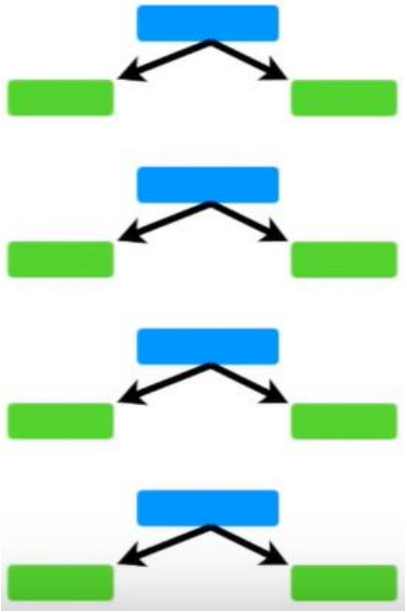
Weak Learner

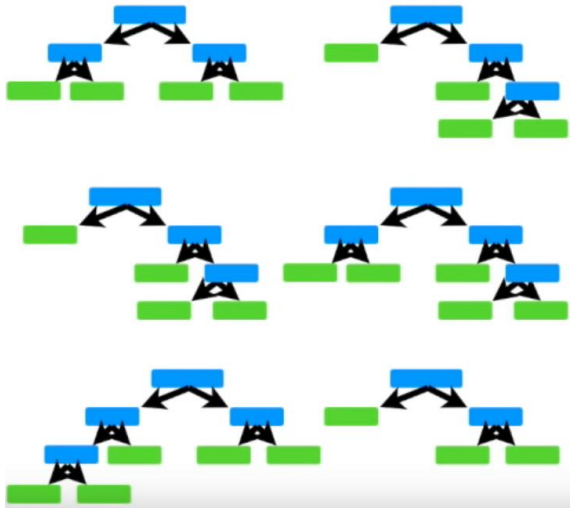
≫ Weak learner로 **Stump**를 사용 (Stump: leaf node만을 가지는 tree, 한 번의 test만 가능)

≫ Stump들을 순차적으로 사용

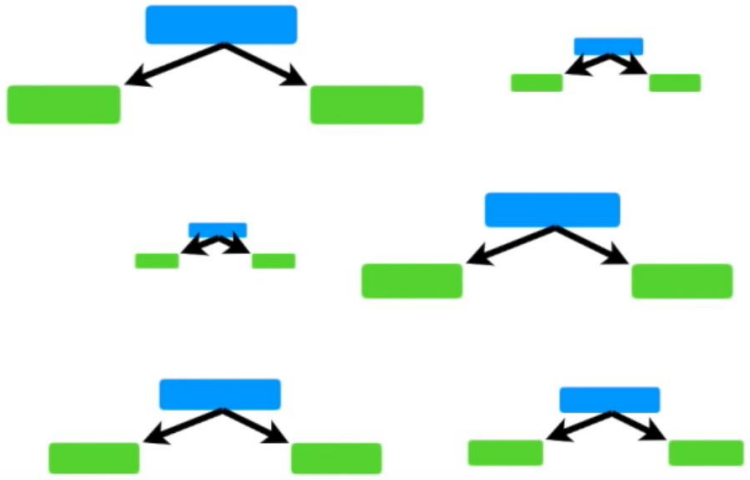
≫ 각 Stump의 error는 다음 Stump의 결과에 영향을 줌

≫ 각각의 Stump에 가중치를 부여하여 최종 예측





Random Forest 예시

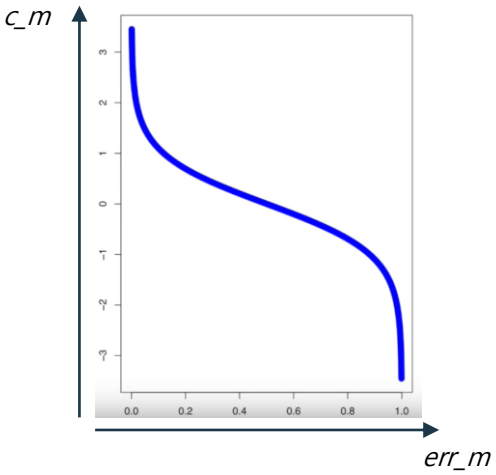


AdaBoost 예시

AdaBoost

AdaBoost 알고리즘

≫ Feature 수만큼 weak learner(stump)를 만들고, gini impurity가 낮은 stump 먼저 사용



N번 반복

≫ 샘플 데이터의 가중치 초기화 $w_i = 1/n,\ i = 1, \ldots, n$

≫ Weak learner로부터 예측한 값에서 error 계산 $err_m = \sum_{i=1}^{n} w_i \cdot \mathbb{I}(y_i \neq \hat{f}_m(\mathbf{x}_i))$

≫ Weak learner의 가중치(*c_m*) 계산 $c_m = \frac{1}{2}\log((1 - err_m)/err_m)$

≫ 기존 데이터셋의 가중치(*w_i*) 업데이트 & 정규화 $w_i = w_i \cdot \exp\left(-c_m \cdot y_i \cdot \hat{f}_m(\mathbf{x}_i)\right)$

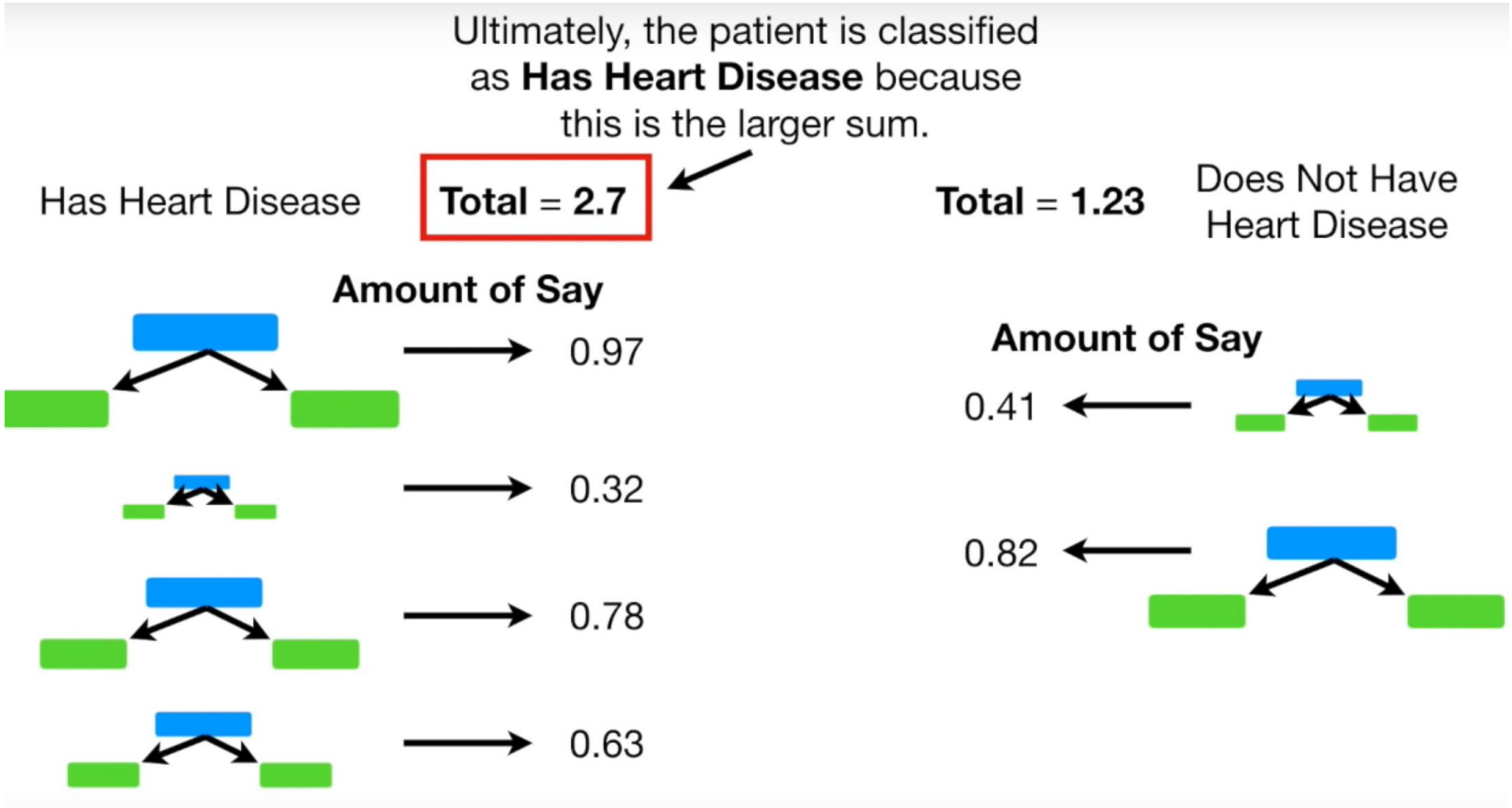
≫ 업데이트된 데이터 가중치를 고려하여 새로운 데이터셋 생성 (중복 가능)

≫ 이 데이터셋으로 다음 weak learner 학습

≫ N번의 반복 후, N개의 weak learner에 모델 가중치(*c_m*)를 주어 최종 모델 생성

AdaBoost

AdaBoost 최종 예측



Gradient Boost

Gradient Boost 개요

≫ Target을 예측하는 것이 아닌 **Residual**을 줄여 나가는 방식으로 학습

≫ Stump 대신 **Tree**로 구성 (보통 leaf가 8~32개)

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

Gradient Boost

GBM 예시 _ Step 1

≫ 보통 초기 추정 값은 평균으로 설정

≫ Residual = (실제 값 – 추정 값)

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

Average Weight

71.2

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

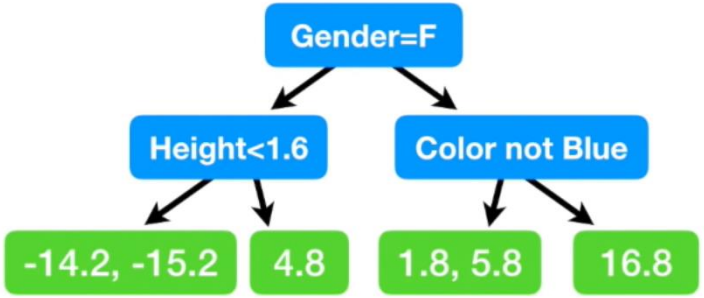
Gradient Boost

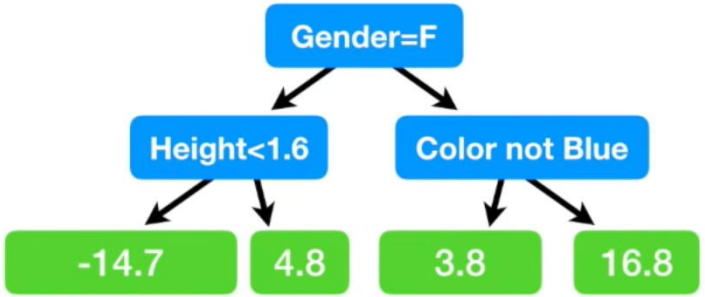
GBM 예시 _ Step 2

≫ Residual을 예측하는 트리 형성

≫ Leaf node에 다수의 residual 있는 경우 평균 값으로 치환

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2





Gradient Boost

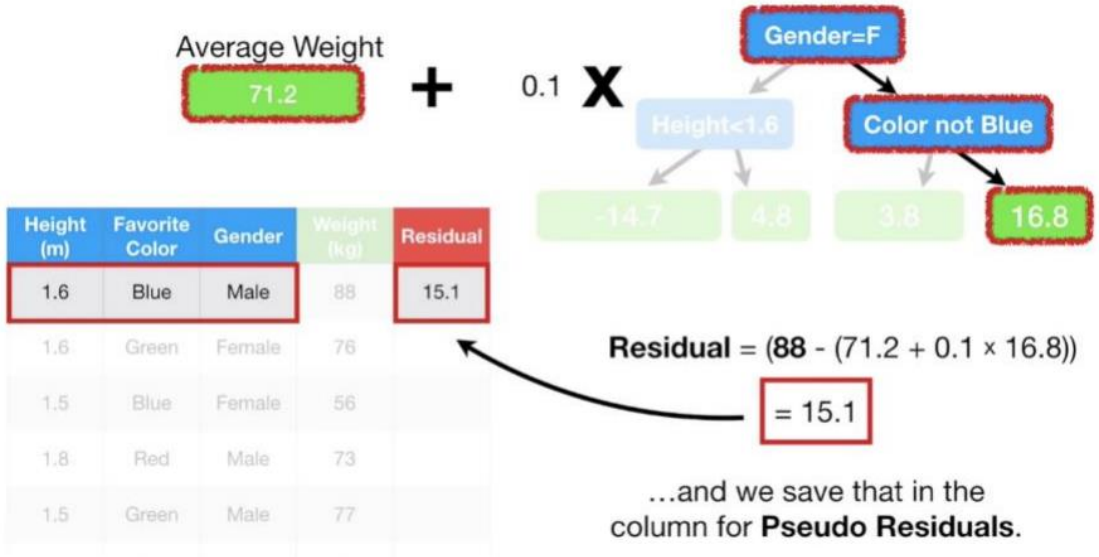
GBM 예시 _ Step 3

≫ 새로운 추정 값 = 기존 추정 값 + (Learning rate * Residual)

- Learning rate 0에서 1 사이의 값

≫ 올바른 방향으로(즉, 실제 값과 가까워지는 방향으로) 조금씩 다가가는 식으로 학습

≫ 새로운 Residual = (실제 값 – 새로운 추정 값)



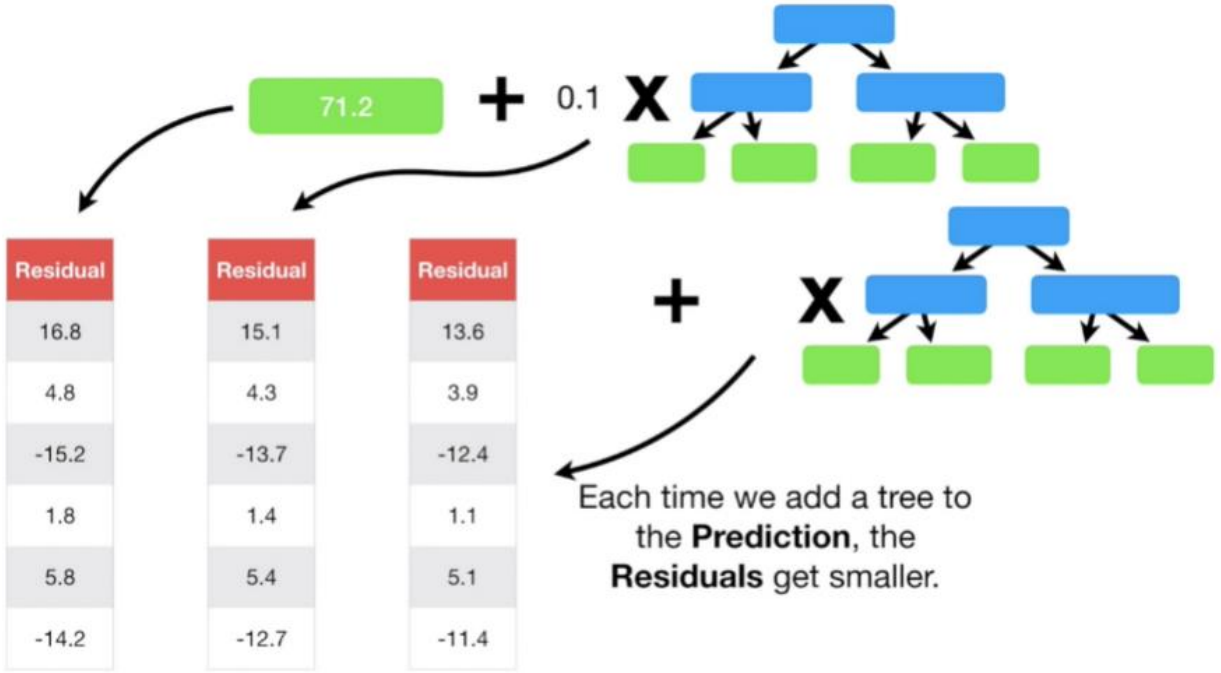
Gradient Boost

GBM 예시 _ Step 4

≫ 앞서 과정을 반복

- 사전에 hyperparameter로 설정해 놓은 반복 횟수에 도달하거나, 더 이상 residual이 작아지지 않을 때까지

≫ Residual이 점차 감소, 즉 예측 정확도가 높아짐

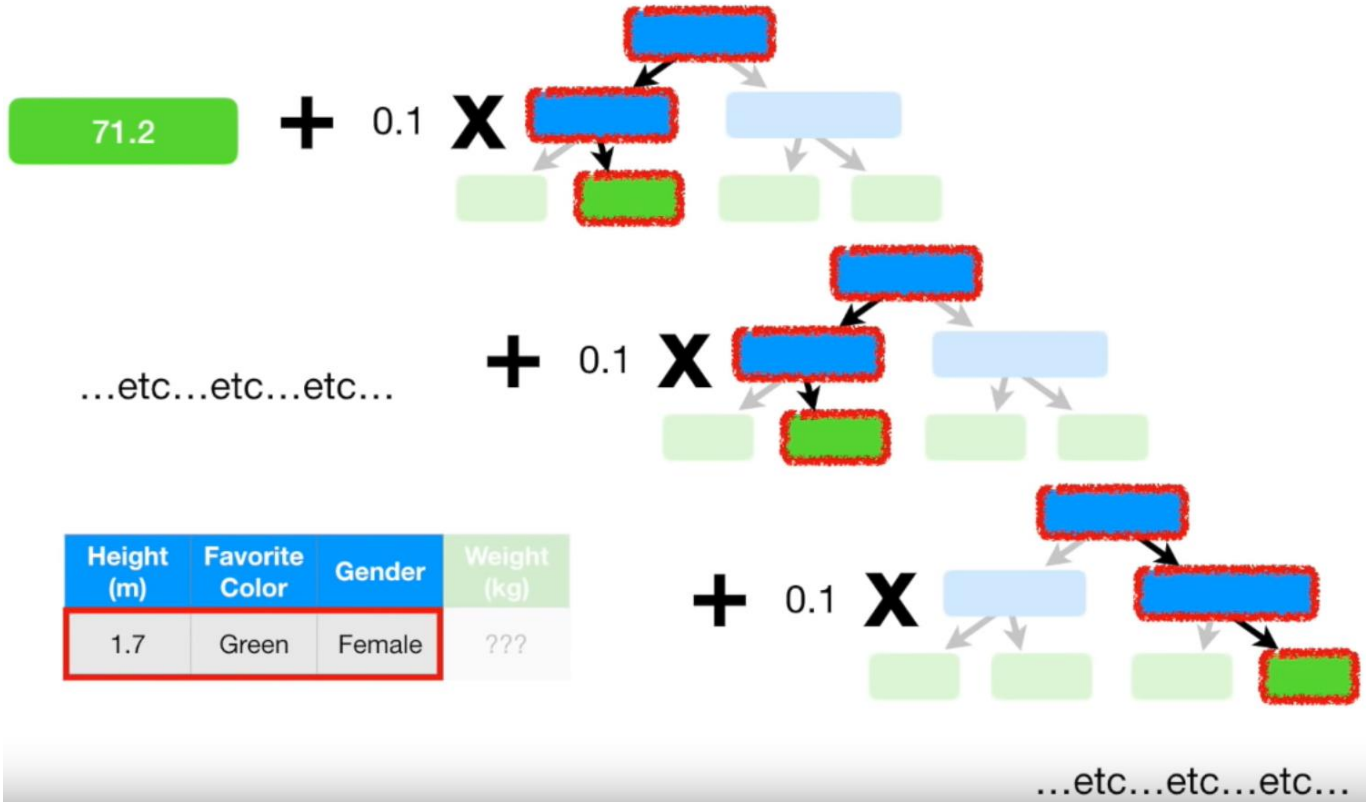


Gradient Boost

GBM 예시 _ 예측

≫ 모든 반복이 완료되면 최종적으로 Gradient Boost 모델 구축 완료

≫ 새로운 데이터가 주어지면 예측 값 구할 수 있음



Gradient Boost

단점

≫ 순차적인 학습으로 인해 많은 시간 소요

≫ 과적합이 일어나기 쉬움

≫ 과적합을 막기 위해 정밀한 hyperparameter 조정이 필요

XGBoost

장점 및 특징

≫ 뛰어난 예측 성능

≫ GBM 대비 빠른 수행 시간

- 병렬 CPU 환경에서 병렬 학습이 가능
- 일반적인 GBM에 비해 수행시간이 빠르다는 것이지, 다른 머신러닝 알고리즘에 비해 빠르다는 의미는 아님

≫ 규제(Regularization)를 추가하여 과적합에 강한 내구성 가짐

≫ 조기 중단(early stopping) 가능

- 지정한 부스팅 반복 횟수에 도달하지 않더라도 예측 오류가 더 이상 개선되지 않으면 중지하여 수행 시간 개선

≫ 결측치에 민감하지 않음

LightGBM

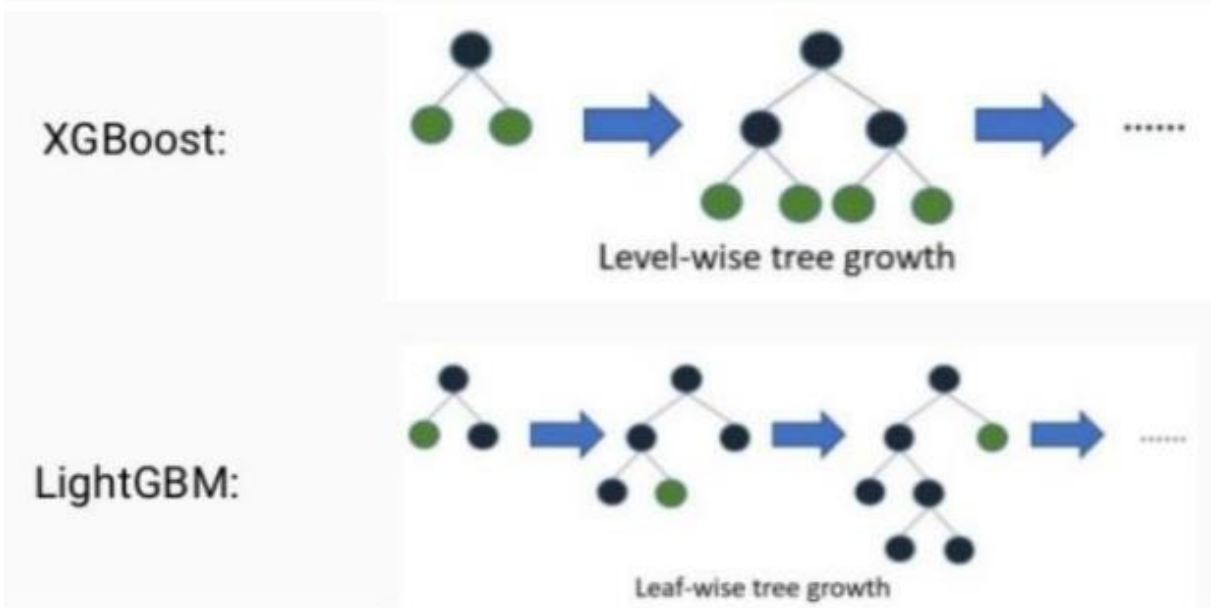
장점 및 특징

≫ XGBoost 또한 GBM보단 빠르지만 여전히 시간이 오래 걸리는 문제 발생

≫ LightGBM은 XGBoost과 유사한 성능 & 훨씬 빠른 속도 & 적은 메모리 사용량

≫ 리프 중심 분할(Leaf Wise) 방식 사용

- 트리의 균형을 맞추지 않고 최대 손실 값을 갖는 리프 노드를 지속적으로 분할
- 트리의 깊이가 깊어지고 비대칭적인 규칙 트리 생성

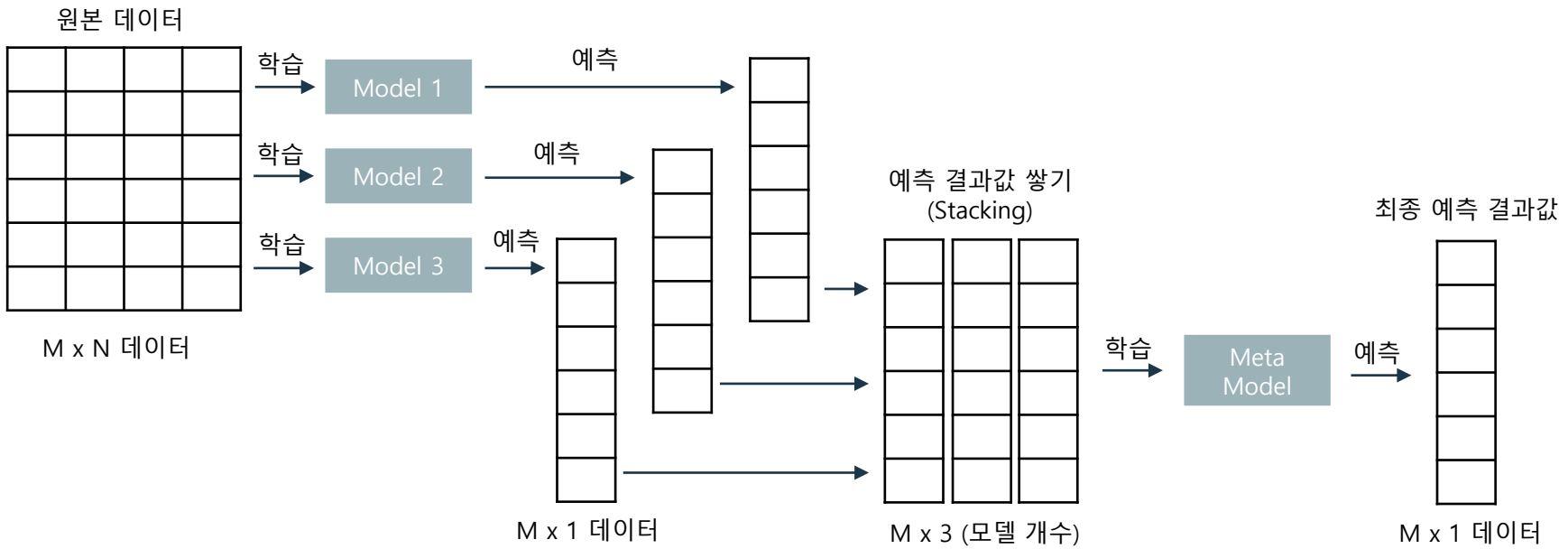


Stacking

기본 Stacking

≫ 개별적인 여러 알고리즘의 예측 결과값을 스태킹 형태로 쌓아 최종 메타 데이터 세트를 만들어 별도의 ML 알고리즘(최종 메타 모델)으로 최종 학습을 수행

≫ 개별 모델로 서로 다른 모델들을 사용 가능 (Ex. SVM, 선형회귀, 신경망 등)

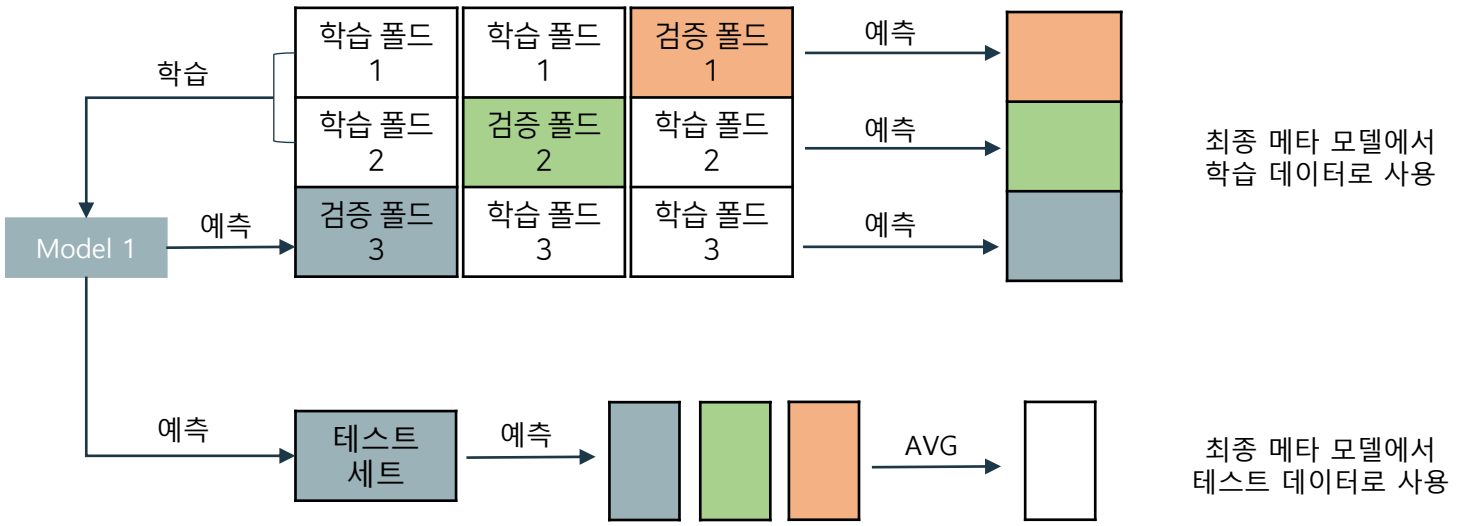


Stacking

CV 세트 기반 Stacking

≫ 과적합을 개선하기 위해 최종 메타 데이터 세트를 만들 때 K-Fold CV를 활용

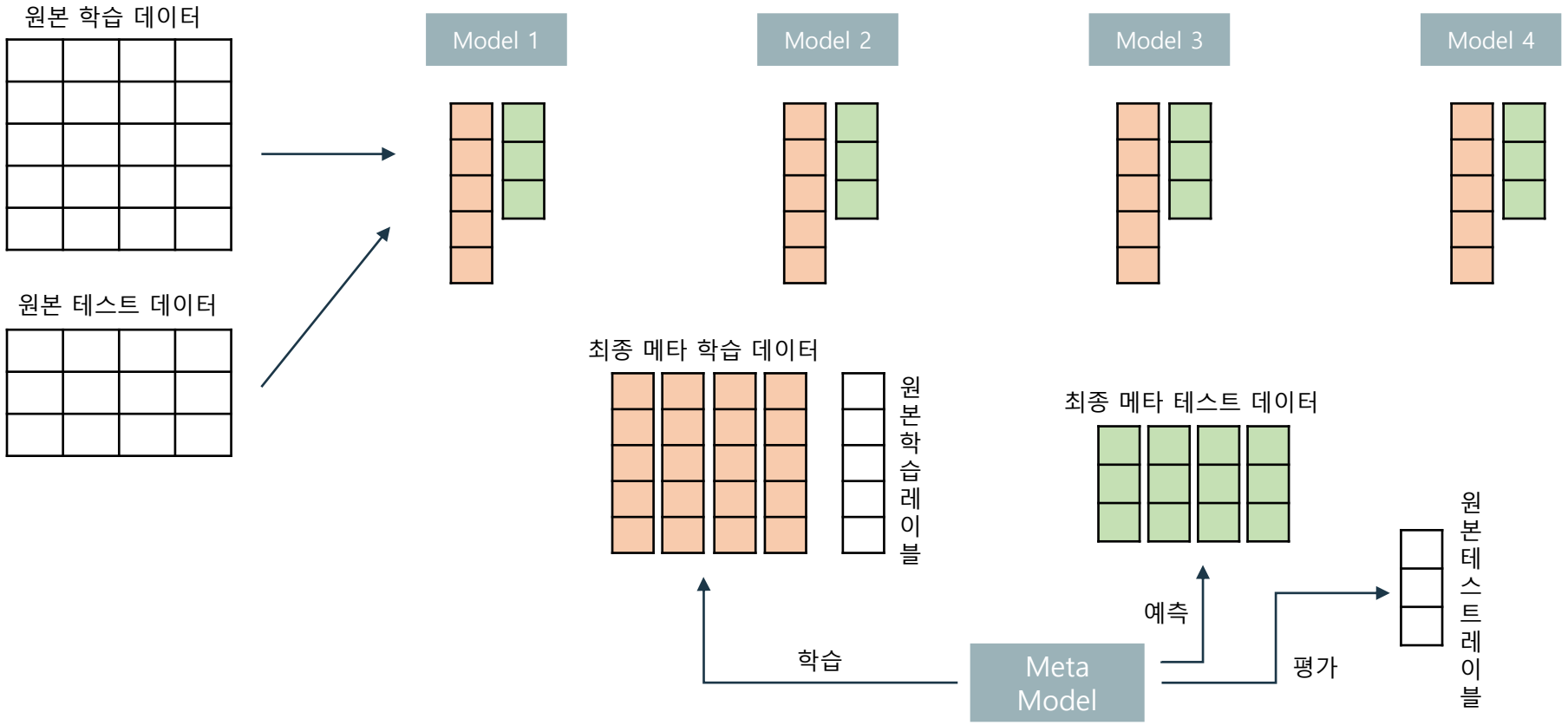
≫ 개별 모델들이 각각 K-Fold로 메타 모델을 위한 학습 스태킹 데이터와 예측을 위한 테스트 스태킹 데이터 생성



Stacking

CV 세트 기반 Stacking

≫ 생성된 학습 스태킹 데이터와 예측을 위한 테스트 스태킹 데이터 기반으로 메타 모델 학습 및 예측 수행



Reference

≫ 권철민 (2020). 파이썬 머신러닝 완벽 가이드 (개정판). 파주: 위키북스.

≫ https://bkshin.tistory.com/entry/머신러닝-14-AdaBoost?category=1057680

≫ https://bkshin.tistory.com/entry/머신러닝-15-Gradient-Boost?category=1057680

≫ Youtube: StatQuest, “AdaBoost, Clearly Explained”

≫ Youtube: StatQuest, “Gradient Boost Part1: Regression Main Ideas”

Thank you