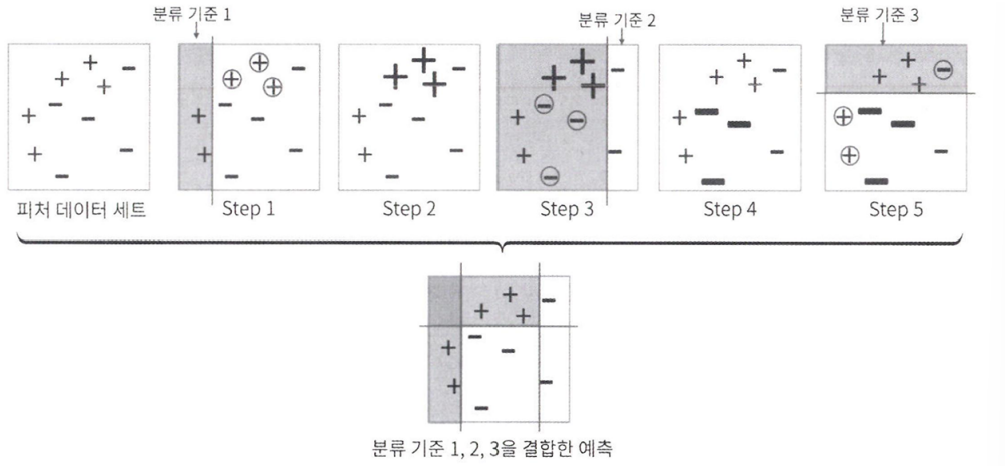
Euron 4주차 예습과제_개념정리_한송희

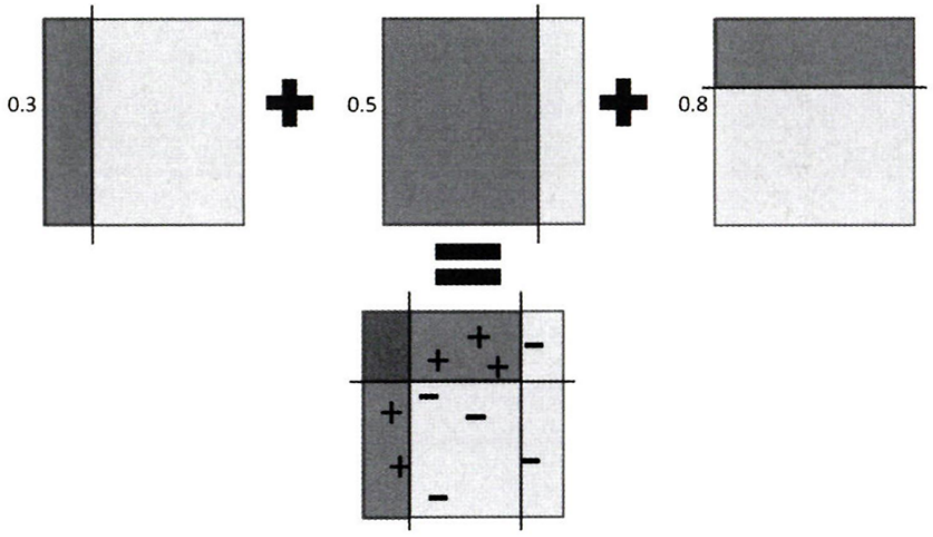
Chapter4 분류

05. GBM(Gradient Boosting Machine)

AdaBoost: 오류 데이터에 가중치를 부여하면서 부스팅을 수행하는 알고리즘



개별 약한 학습기는 각각의 가중치를 부여해 결합함



GBM: AdaBoost와 유사하나 가중치 업데이트에 Gradient Descent를 이용함

오류값=실제값-예측값

h(x)=y-F(x)를 최소화하는 방향성을 가지고 반복적으로 가중치 업데이트=Gradient Descent

하이퍼 파라미터

-loss: GD에서 사용할 비용함수

-learning_rate: GBM이 학습을 진행할 때마다 적용하는 학습률

-n_estimators: weak learner의 개수

-subsample: weak learner가 학습에 사용하는 데이터 샘플링 비

GBM장점: 과적합에도 강한 뛰어난 예측 성능을 가짐

단점: 수행 시간이 오래 걸림

06. XGBoost(eXtra Gradient Boost)

XGBoost: 트리 기반 앙상블 학습에서 가장 주목받는 알고리즘. GBM에 기반하고 있으며 GBM의 단점인 느린 수행시간과 과적합 규제가 없는 문제를 해결함

장점: 뛰어난 예측 성능, GBM보다 빠른 수행시간, 과적합 규제(regularization), tree pruning(가지치기), 자체 내장도니 교차 검증, 결손값 자체 처리

하이퍼 파라미티

-일반 파라미터: booster, silent, nthread

-부스터 파라미터: eta, num_boost_round, min_chilf_weight, gamma, max_depth, sub_sample, colsample_bytree, lambda, alpha, scale_pos_weight

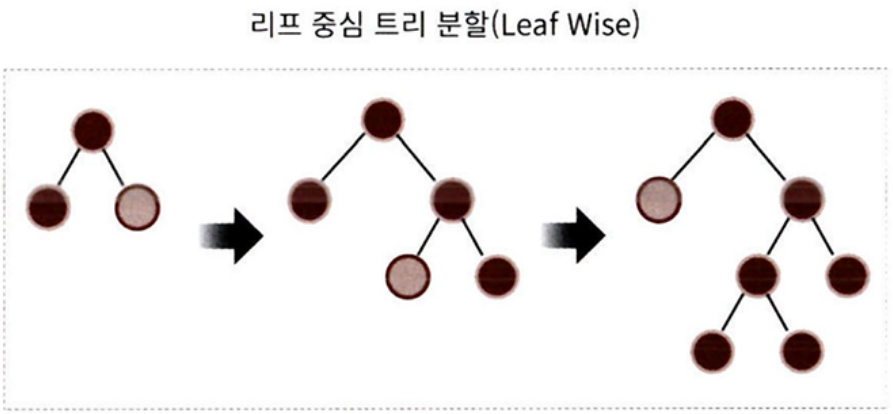
-학습 태스크 파라미터: objective, binary:logistic, multi:softmax, multi:softprob, eval_metric

07. LightGBM

LightGBM: XGBoost보다 학습에 걸리는 시간이 훨씬 적음. 메모리 사용량 상대적으로 적음. 성능 차이 없고 기능상 더 다양함. but 작은 데이터 세트에 적용할 경우 과적합이 발생하기 쉬움

일반 GBM개열의 트리 분할과 다르게 리프 중심 트리 분할 방식 사용⇒ 균형 트리 방식보다 예측 오류 손실을 최소화할 수 있음

-리프 중심 트리 분할 방식 사용



LightGBM 장점: XGBoost대비 더 빠른 학습과 예측 수행, 더 적은 메모리 사용, 카테고리형 피처의 자동 변환과 최적 분할, 대용량 데이터에 대한 뛰어난 성능, GPU지원

-하이퍼 파라미터

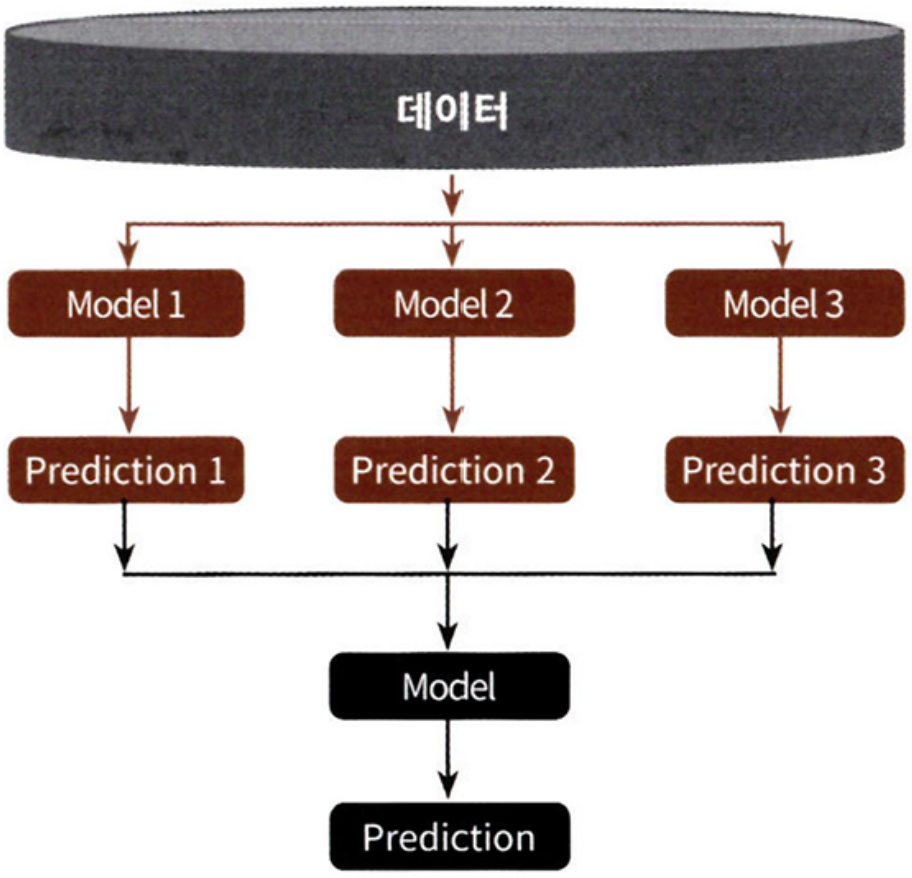
주요 파라미터: num_iterations, learning_rate, max_depth, min_data_in_leaf, num_leaves, boosting, bagging_fraction, feature_fraction, lambda_l2, lambda_l1

learning task 파라미터: objective,

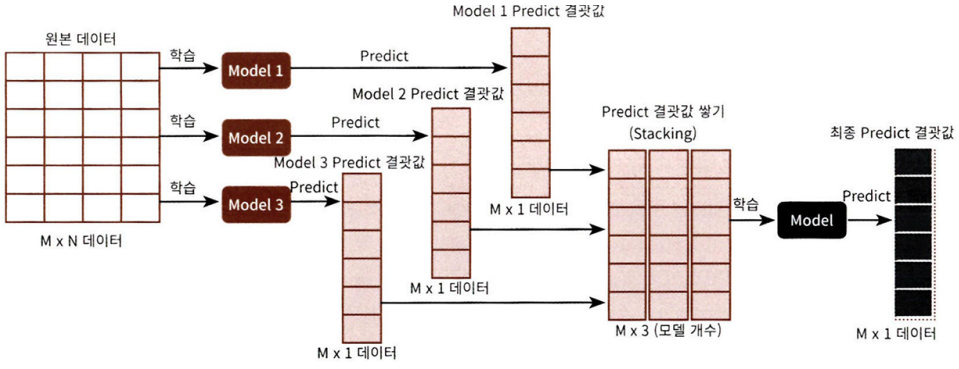
10. Stacking Ensemble

stacking: 개별적인 여러 알고리즘을 서로 결합해 예측 결과를 도출+개별 알고리즘으로 예측한 데이터를 기반으로 다시 예측을 수행함

⇒ 개별적인 기반모델+개별 기반 모델의 예측 데이터를 학습 데이터로 만들어서 학습하는 최종 메타 모델

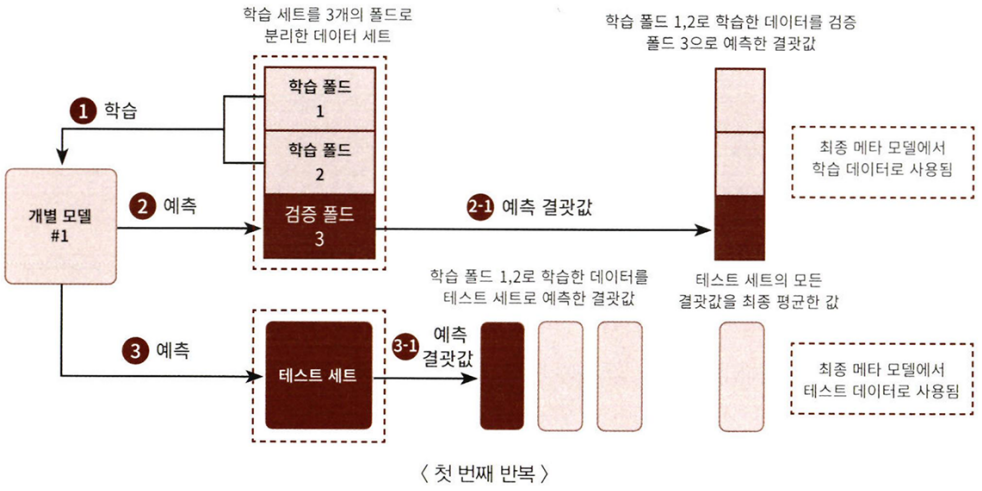


M개의 low, N개의 col를 가진 데이터 세트, ML알고리즘 모델은 3개

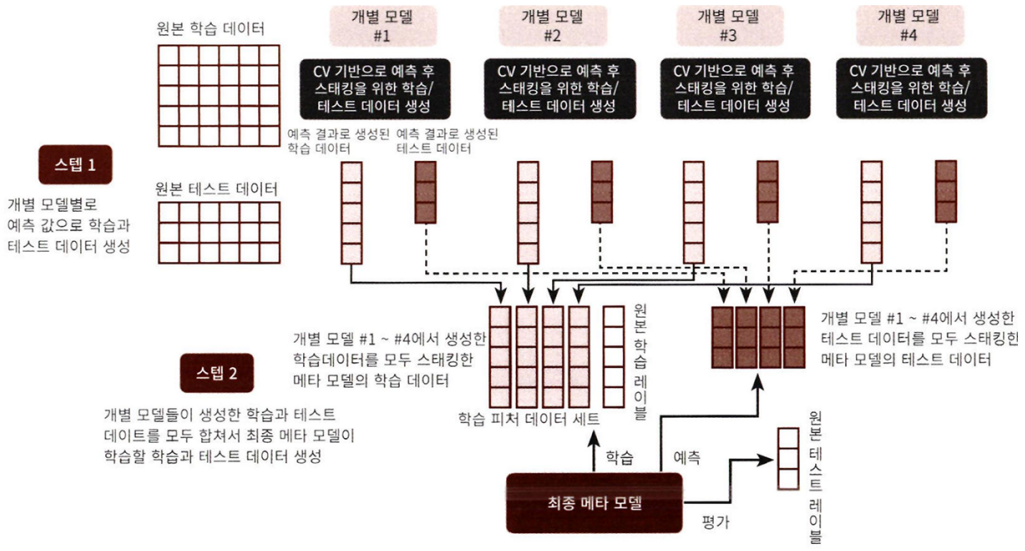


CV기반 stacking: 과적합을 개선하기 위해 최종 메타 모델을 위한 데이터 세트를 만들 때 교차 검증 기반으로 예측된 결과 데이터 셋을 이용함

step1) 각 모델별로 원본 학습/테스트 데이터를 예측한 결과 값을 기반으로 메타 모델을 위한 학습용/테스트용 데이터를 생성함



step2) step1에서 개별 모델들이 생성한 학습용 데이터를 모두 stacking형태로 합쳐서 메타 모델이 학습할 최종 학습용 데이터 셋을 생성함



학습용 데이터를 n개의 폴드로 나눔→ 1개는 검증을 위한 데이터 폴드로 설정하고 나뉜 학습 데이터를 기반으로 개별 모델 학급→ 학습된 개별 모델은 검증 폴드 1개의 데이터로 예측하고 그 결과 저장→ n번 반복→ 이렇게 만들어진 예측 데이터는 메타 모델을 학습시키는 학습 데이터로 사용→ 학습 폴드 데이터로 학습된 개별 모델은 원본 테스트 데이터 예측하여 예측값 생성→ n번 반복→ 예측값의 평균으로 최종 결과값 생성하고 이를 메타 모델을 위한 테스트 데이터로 사용

+) HyperOpt

Grid Search방식의 수행 시간이 오래 걸리는 단점을 해결하기 위해 베이지안 최적화를 적용함

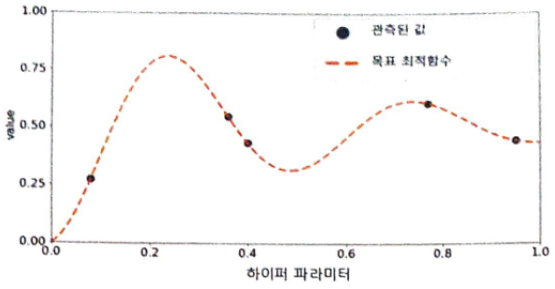
-베이지안 최적화: 목적 함수 식을 제대로 알 수 없는 블랙박스 형태의 함수에서 최대 또는 최소 함수 반환 값을 만드는 최적 입력값을 가능한 적은 시도를 통해 빠르고 효과적으로 찾아주는 방식(대체 모델+획득 함수)

대체 모델: 획득 함수로부터 최적 함수를 예측할 수 있는 입력값을 추천 받은 뒤 이를 기반으로 최적 함수 모델을 개선

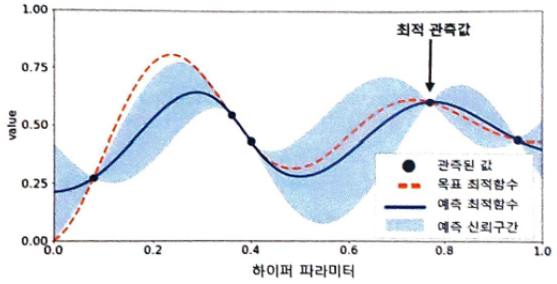
획득 함수: 개선된 대체 모델을 기반으로 최적 입력값을 계산

과정

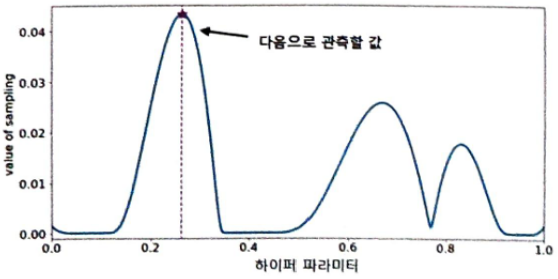
- Step 1: 최초에는 랜덤하게 하이퍼 파라미터들을 샘플링하고 성능 결과를 관측합니다. 아래 그림에서 검은색 원은 특정 하이퍼 파라미터가 입력되었을 때 관측된 성능 지표 결괏값을 뜻하며 주황색 사선은 찾아야 할 목표 최적함수 입니다.



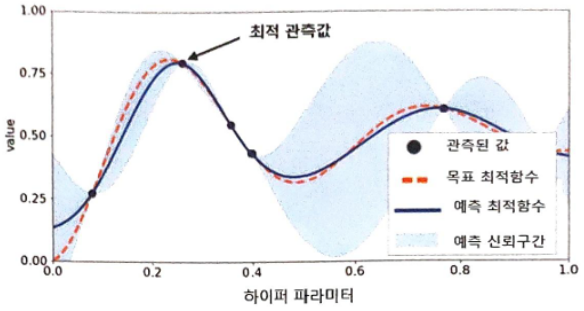
- Step 2: 관측된 값을 기반으로 대체 모델은 최적 함수를 추정합니다. 아래 그림에서 파란색 실선은 대체 모델이 추정한 최적 함수 입니다. 옅은 파란색으로 되어 있는 영역은 예측된 함수의 신뢰 구간입니다. 추정된 함수의 결괏값 오류 편차를 의미하며 추정 함수의 불확실성을 나타냅니다. 최적 관측값은 y축 value에서 가장 높은 값을 가질 때의 하이퍼 파라미터입니다.



- Step 3: 추정된 최적 함수를 기반으로 획득 함수(Acquisition Function)는 다음으로 관측할 하이퍼 파라미터 값을 계산합니다. 획득 함수는 이전의 최적 관측값보다 더 큰 최댓값을 가질 가능성이 높은 지점을 찾아서 다음에 관측할 하이퍼 파라미터를 대체 모델에 전달합니다.



- Step 4: 획득 함수로부터 전달된 하이퍼 파라미터를 수행하여 관측된 값을 기반으로 대체 모델은 갱신되어 다시 최적 함수를 예측 추정합니다.



<실습>

search_space: 입력값의 검색공간

hp.quniform: label로 지정된 입력값 변수 검색 공간을 최솟값에서 최댓값까지 q의 간격을 가지고 설정

hp.uniform: 0부터 최댓값 upper까지 random한 정숫값으로 검색 공간 설정

hp.logunifomr: exp(uniform(low,high))값을 반환하며, 반환 값의 log변환 된 값은 정규분포 형태를 가지는 검색 공간 설정

hp.choice: 검색 값이 문자열 또는 문자열과 숫자값이 섞여 있을 경우 설정

def objective_func 등으 ㅣ형태로 목적 함수 설정

fmin(objective, space,algo, max_evals,trials)함수: 베이지안 최적화 기법

fn: 목적함수

space: 검색 공간 딕셔너리

algo: 베이지안 최적화 적용 알고리즘

max_evals: 최적 입력값을 찾기 위한 입력값 시도 횟수

trials: 최적 입력값을 찾기 위해 시도한 입력값 및 해당 입력값의 목적 함수 반환값 결과를 저장

rstate: fmin()을 수행할 때 마다 동일한 결과값을 가지게 설정=random seed