♥ TITLE : 4_ 분류 ♥ DATE : 2022.09.06

5 GBM (Gradient Boosting Machine)

① 개요

- 부스팅: 여러 개의 약한 학습기를 순차적으로 학습·예측하면서 잘못 예측한 데이터에 가중치 부여를 통해 오류를 개선해 나가면서 학습하는 방식

ex) AdaBoost (Adaptive boosting)

↳ 약한 학습기가 순차적으로 오류 값에 대해 가중치를 부여한 예측 결정 기준을 모두 결합해 예측 수행

- GBM
  - 경사 하강법을 이용하여 오류를 최소화할 수 있도록 가중치의 업데이트 값을 도출하는 기법
    - $h(x) = y - F(x)$
    - $h(x)$: 오류식, y: 결과, $F(x)$: 예측 함수
  - 사이킷런의 GradientBoostingClassifier 이용
- 일반적으로 랜덤 포레스트보다는 예측 성능이 조금 더 good

but 수행 시간이 오래 걸림 ⊕ 하이퍼 파라미터 튜닝이 hard

↳ 병렬 처리 자원 × (순차적 예측 오류 보정)

② 하이퍼 파라미터 & 튜닝

i) loss: 경사 하강법에서 사용할 비용 함수 지정, default = 'deviance'

ii) learning_rate
- 학습 진행 시마다 적용하는 학습률, 0~1 사이, default = 0.1
- 약한 학습기가 순차적으로 오류 값을 보정해 나가는데 적용하는 계수
- learning_rate ↓
  - 업데이트 되는 값이 작아짐 → 최소 오류 값 탐색
  - 순차적 반복 ≒ 시간 ↓↓
- learning_rate ↑
  - 최소 오류 값을 찾지 못하고 지나쳐 버릴 수도 O
  - 빠른 수행이 가능함.

iii) n_estimators
- weak learner의 개수, default = 100
- 순차적 오류 보정 ⇒ 개수가 많을수록 예측 성능이 일정 수준까지 좋아질 수 있음 but 시간 ↑

★ learning_rate를 작게 하고 n_estimators를 크게 하면 한계점까지는 예측 성능이 조금씩 좋아질 수 O

iv) subsample
- weak learner가 학습에 사용하는 데이터의 샘플링 비율
- default = 1 ⇒ 전체 학습 데이터를 기반으로 학습
- 과적합이 염려되는 경우 1보다 작은 값으로 설정

6 XGBoost (eXtra Gradient Boost)

① 개요

i) 뛰어난 예측 성능

ii) GBM 대비 빠른 수행 시간

↳ 병렬 수행 & 다양한 기능

iii) 과적합 규제

↳ 자체 과적합 규제 기능 → 조금 더 강한 내구성

iv) 나무 가지치기

↳ 더 이상 긍정 이득이 없는 분할을 가지치기해서 분할 수를 더 줄일 수 O

v) 자체 내장된 교차 검증

↳ 반복 수행 시마다 내부적으로 학습 데이터 세트와 평가 데이터 세트에 대한 교차 검증 수행 ⇒ 최적화된 반복 수행 횟수 get

e.g. 조기 중단 기능이 O

vi) 결손값 처리

↳ 결손값 자체 처리

♥ TITLE : 4_분류 ♥ DATE : 2022.09.06 ~ 08

② 파이썬 래퍼 XGBoost 하이퍼 파라미터

- 초기의 독자적인 XGBoost 프레임워크 기반의 XGBoost
- 파라미터의 유형>
  - i) 일반 파라미터 ┌ 일반적으로 실행 시 스레드의 개수 or silent 모드 등 선택
    └ default 값을 거의 변경하지 않고 사용
  - ii) 부스터 파라미터 : 트리 최적화, 부스팅, 규제 등 , 대부분의 파라미터들
  - iii) 학습 태스크 파라미터 : 학습 수행 시의 객체 함수, 평가를 위한 지표 등 설정

✶ 파이썬 p. 231 ~ 232

- 조기 중단(Early Stopping)
  - ↳ n_estimators에 지정된 부스팅 반복 횟수를 채우지 못해도 예측 오류가 더 이상 개선되지 않으면 반복을 중단
  - ↳ xgboost의 train( ) 함수에 early_stopping_rounds 파라미터를 입력하여 설정
  - ↳ 반드시 eval_set와 eval_metric이 함께 설정되어야 함.
    - eval_set: 성능 평가를 수행할 평가용 데이터 세트
    - eval_metric: 평가 세트에 적용할 성능 평가 방법

- plot_importance( ): 피처 중요도 시각화
- to_graphviz( ) : 규칙 트리 구조 그리기
- CV( ) : 교차 검증 수행 후 최적 파라미터 구하기
  - i) params : 딕셔너리 / 부스터 파라미터
  - ii) dtrain : DMatrix / 학습 데이터
  - iii) num_boost_round : int / 부스팅 반복 횟수
  - iv) nfold : int / 폴드 개수
  - v) stratified : bool / Stratified K-fold 수행 여부
  - vi) metrics : str or str list / 교차 검증 시 모니터링할 성능 평가 지표
  - vii) early_stopping_rounds : int / 조기 중단 활성화

7 LightGBM

- XGBoost에 비해 학습에 걸리는 시간이 적음 & 메모리 사용량 적음
- 적은 데이터 세트(일반적으로 10000건 이하)에 적용할 경우 과적합이 발생하기 쉬움
- 리프 중심 트리 분할 방식을 사용
- Categorical feature의 자동 변환과 최적 분할
  - ↳ one-hot encoding 등을 사용하지 않고도 feature를 최적으로 변환하고 이에 따른 노드 분할 수행

① 하이퍼 파라미터

✶ 파이썬 p. 247 ~ 248

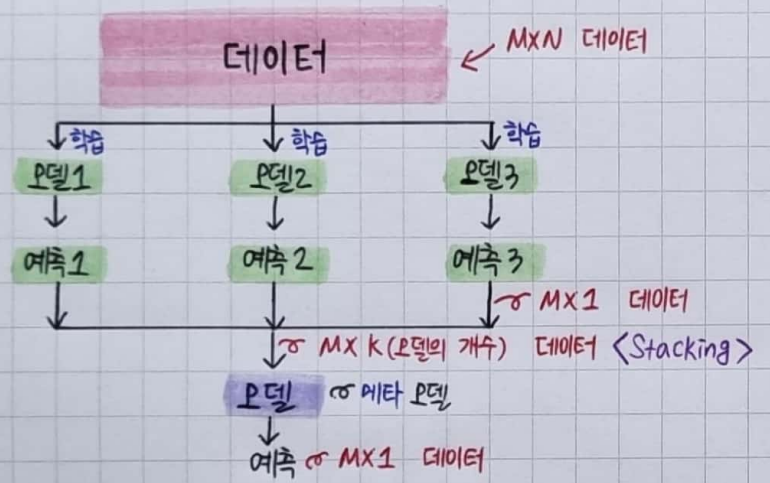
② 하이퍼 파라미터 튜닝

- ↳ num_leaves의 개수를 중심으로 min_child_sample , max_depth를 함께 조정
  ⇒ 모델의 복잡도 ↓ , 과적합 방지
- ↳ learning_rate를 작게 하면서 n_estimators를 크게 하는 것

♥ TITLE : 4_분류 ♥ DATE : 2022.09.12

10 스태킹 앙상블

- ① 개별 알고리즘의 예측 결과 데이터 세트를 최종적인 메타 데이터 세트로 만들어 별도의 ML 알고리즘으로 최종 학습을 수행하고 테스트 데이터를 기반으로 다시 최종 예측을 수행하는 방식 ⇒ ② 메타 모델
- 일반적으로 성능이 비슷한 모델을 결합해 좀 더 나은 성능 향상을 도출하기 위해 적용



- **CV 세트 기반의 스태킹**
  - 최종 메타 모델을 위한 데이터 세트 생성 시 교차 검증 기반으로 예측된 결과 데이터 세트를 이용 ⇒ 과적합 개선
  - **Step**
    - i) 각 모델별로 원본 학습/테스트 데이터를 예측한 결과 값을 기반으로 메타 모델을 위한 학습용/테스트용 데이터 생성
    - ii) 개별 모델들이 생성한 데이터를 모두 스태킹 형태로 합쳐서 메타 모델이 학습할 데이터 세트 생성

    ⇒ 메타 모델은 최종적으로 생성된 학습 데이터 세트와 원본 학습 데이터의 레이블 데이터를 기반으로 학습한 뒤, 최종적으로 생성된 테스트 데이터 세트를 예측하고, 원본 테스트 데이터의 레이블 데이터를 기반으로 평가