05. GBM (Gradient Boosting Machine)

- 부스팅 알고리즘

여러개의 약한 학습기(weak learner)를 순차적으로 학습-예측하면서 잘못 예측한 데이터에 가중치 부여를 통해 오류를 개선해 나가면서 학습하는 방식

Ex) AdaBoost (Adaptive boosting)

그래디언트 부스트

- AdaBoost (에이다 부스트)

오류 데이터에 가중치를 부여하면서 부스팅을 수행하는 대표적인 알고리즘

- GBM (Gradient Boost Machine)

: AdaBoost와 유사

· 가중치 업데이트를 경사하강법 (Gradient Descent) 이용

오류 값 = 실제 값 - 예측값

y : 분류의 실제값

$x_1, x_2 \cdots x_n$: 피처

$F(x)$: 피처에 기반한 예측함수

오류식 $h(x) = y - F(x)$

경사하강법: $y - F(x)$ 최소화하는 방향으로 반복적으로 가중치 업데이트

: 일반적으로 랜덤 포레스트에 비해 예측 성능↑

but 수행 시간↑, 하이퍼 파라미터 튜닝 ㅠㅠ

- GBM 하이퍼 파라미터

(트리 기반 자체의 파라미터 생략)

loss

: 경사하강법에서 사용할 비용 함수

: default = 'deviance'

learning_rate

: GBM이 학습을 진행할 때마다 적용하는 학습률

: Weak learner가 순차적으로 오류값을 보정해 나가는데 적용하는 계수

: 0 ~ 1 사이 지정, default = 0.1

· 값↑ → 최소 오류값 찾기 X → 예측 성능↓

: n_estimators 와 상호 보완적으로 조합해 사용

: learning_rate ㉮ & n_estimators ㉯

→ 한계점까지 예측 성능 좋아질 수 O

→ 수행시간↑, 예측 성능 획기적 좋아지지 X

n_estimators

: Weak learner의 개수

: ↑ → 일정 수준까지 okay → 수행시간↑

: default = 100

Subsample

: Weak learner가 학습에 사용하는 데이터 샘플링 개수

: default = 1 (전체 학습 데이터 기반)

: 과적합 대비) 1보다 작게 설정

06 XGBoost (eXtra Gradient Boost)

- XGBoost
  : GBM 기반
  : GBM의 느린수행시간, 과적합 규제 부재 등의 문제 해결
- 장점
  : 뛰어난 예측 성능
  : GBM 대비 빠른 수행 시간
  : 과적합 규제
  : Tree pruning(나무 가지치기)
  : 자체 내장된 교차 검증
  : 결손값 자체 처리
- 파이썬 래퍼 XGBoost 하이퍼 파라미터
  - 일반 파라미터 : 디폴트 파라미터 값을 바꾸는 경우 거의 X
  - 부스터 파라미터 : 트리 최적화, 부스팅, 규제 등과 관련
  - 학습 태스크 파라미터 : 학습 수행 시 객체 함수, 평가를 위한 지표 등 설정
- 과적합 문제가 심각하다면
  : eta 값 낮춤(0.01~0.1)
  eta 값을 낮출 경우)
  num-round(or n-estimators)는 높여주기
  : max_depth ↓
  : min_child_weight ↑
  : gamma ↑
  : Subsample, colsample_bytree 조정
- 사이킷런 래퍼 XGBoost
  eta → learning rate
  Sub-Sample → subsample
  lambda → reg_lambda
  alpha → reg_alpha

주요 일반 파라미터	
booster	gbtree(tree based model) /gblinear(linear model) 선택, default = gbtree
silent	default = 0 출력 메세지를 나타내고 싶지 않을 경우 : 1
nthread	CPU의 실행 스레드 개수 default = CPU의 전체 스레드 다 사용

주요 부스터 파라미터	
eta [defualt = 0.3, alias: learning_rate]	GBM의 learning_rate와 동일한 파라미터 0~1 값 지정, 부스팅 스템을 반복적으로 수행할 때 업데이트되는 학습 값 defalut = 0.1 보통 0.01~0.2 값 선호
num_boost_rounds	GBM의 n_estimators와 같은 파라미터
min_child_weight [default = 1]	트리에서 추가적으로 가지를 나눌지를 결정하기 위해 필요한 데이터들의 weight의 총합 클수록 분할 자제, 과적합 조절하기 위해 사용
gamma [default = 0, alias : min_split_loss]	트리의 리프 노드를 추가적으로 나눌지를 결정할 최소 손실 감소값 해당값보다 큰 손실(loss)이 감소된 경우데 리프 노드 분리 클수록 과적합 감소 효과
max_depth [default = 6]	트리 기반 알고리즘의 max_depth와 동일 0으로 지정 -> 깊이에 제한 X 보통 3~10의 값 적용
sub_sample [default = 1]	GBM의 subsample과 동일 트리가 커져서 과적합되는 것을 제어하기 위해 데이터를 샘플링하는 비율을 지정 보통 0.5~1 값 적용
colsample_bytree [default = 1]	GBM의 max_features와 유사 트리 생성에 필요한 피처(칼럼)를 임의로 샘플링하는 데 사용 매우 많은 피처가 있는 경우 과적합 조정하는 데 적용
lambda [default = 1, alias : reg_lambda]	L2 Regularization 적용 값, 피처 개수가 많을 경우 적용을 검토, 클수록 과적합 감소 효과 O
alpha [default = 0, alias : reg_alpha]	L1 Regularization 적용 값, 피처 개수가 많을 경우 적용을 검토, 클수록 과적합 감소 효과 O
scale_pos_weight [default = 1]	특정 값으로 치우친 비대칭한 클래스로 구성된 데이터 세트의 균형을 유지하기 위한 파라미터

학습 태스크 파라미터	
objective	최솟값을 가져야 할 손실 함수 정의 이진분류/다중분류인지에 따라 딸라짐
binary:logistic	이진 분류일 때 적용
multi:softmax	다중분류일 때 적용 레이블 클래스의 개수인 num_class 파라미터 지정해야함
multi:softprob	multi:softmax와 유사하나 개별 레이블 클래스의 해당되는 예측 확률을 반환
eval_metric	검증에 사용되는 함수 정의 기본값 : 회귀) mse 분류) error

eval_metric의 값 유형	
rmse	Root Mean Square Error
mae	Mean Absolute Error
logloss	Negative log-likelihood
error	Binary classification error rate(0.5 threshold)
merror	Multiclass classification error rate
mlogloss	Multiclass logloss
auc	Area under the curve

07 LightGBM

- LightGBM
  - : XGBoost의 장점 계승, 단점 보완
  - : 적은 데이터 세트에 적용할 경우 과적합↑ 10,000건 이하
  - : 일반 GBM 계열의 트리 분할 방법과 다르게 리프 중심 분할 방식 사용(leaf wise)

균형 트리 분할 (level wise)	리프 중심 분할 (leaf wise)
- 트리의 깊이 효과적으로↓	- 트리의 균형X
- 최대한 균형 잡힌 트리 유지하여 분할 → 트리의 깊이 최소화 ⇒ 오버피팅에 더 강함 But 시간↑(균형 맞추기위해)	- 최대 손실값을 (max delta loss) 가지는 리프노드를 지속적으로 분할하면서 트리가 깊어지고 비대칭적인 규칙 트리 생성 ⇒ 예측 오류손실 최소화

- 하이퍼 파라미터 튜닝 방안

  num_leaves의 개수를 중심으로

  min_child_samples (min_data_in_leaf),

  max_depth 함께 조정

  → 모델 복잡도 줄이기

  num_leaves

  : 개별 트리가 가질 수 있는 최대 리프의 개수

  : ↑→ 정확도↑ & 트리 깊이↑, 복잡도↑, 과적합↑

  min_data_in_leaf

  : 큰 값으로 설정 → 트리가 깊어지는 것 방지

  max_depth

  : 명시적으로 깊이의 크기 제한

LightGBM 하이퍼 파라미터

num_iterations [default = 100]	반복 수행하려는 트리의 개수 클수록 예측 성능 높아질 수 있지만 너무 크면 과적합 가능
learing_rate [default = 0.1]	0~1 사이의 값을 지정하면 부스팅 스텝을 반복적으로 수행할 때 업데이트 되는 학습률 값 n_estimators 크게, learning_rate 작게 -> 예측 성능 향상 가능
max_depth [default = -1]	트리 기반 알고리즘의 max_depth와 동일 leaf wise 기반이므로 깊이가 상대적으로 더 깊음
min_data_in_leaf [default = 200]	결정트리의 min_samples_leaf와 동일 최종 클래스 결정인 리프 노드가 되기 위해 최소한으로 필요한 레코드 수 과적합 제어
num_leaves [default = 31]	하나의 트리가 가질 수 있는 최대 리프 개수
boosting [default = gbdt]	부스팅의 트리를 생성하는 알고리즘을 기술 gbdt : 일반적인 그래디언트 부스팅 결정 트리 rf : 랜덤 포레스트
baggin_fraction [default = 1.0]	데이터를 샘플링 하는 비율(트리가 커져 과적합되는 것을 제어)
feature_fraction [default = 1.0]	개별 트리를 학습할 때마다 무작위로 선택하는 피처의 비율 과적합 제어
lambda_l2 [default = 0.0]	L2 regulation 제어 위한 값 피처 개수가 많을 경우 적용 검토, 값이 클수록 과적합 감소효과
lambda_l1 [default = 0.0]	L1 regulation 제어 위한 값 과적합 제어
objective	최솟값이 가져야 할 손실함수 정의 회귀, 다중 클래스 분류, 이진 분류인지에 따라 정해짐

11 스태킹 앙상블

- 스태킹 (Stacking)

  : 개별적인 여러 알고리즘을 서로 결합해 예측결과 도출

  : 개별 알고리즘으로 예측한 데이터를 기반으로 다시 예측을 수행

  (메타 모델: 개별 모델의 예측된 데이터 세트를 다시 기반으로 하여 학습하고 예측하는 방식)

- 스태킹 모델은 두가지 종류의 모델 필요

  ① 개별적인 기반 모델

  ② 이 개별 기반 모델의 예측 데이터를 학습 데이터로 만들어서 학습하는 최종 메타 모델

- CV 기반의 스태킹

  과적합을 개선하기 위해 최종 메타 모델을 위한 데이터 세트를 만들 때 교차 검증 기반으로 예측된 결과 데이터 세트를 이용