5. GBM(Gradient Boosting Machine)

GBM의 개요 및 실습

부스팅 : 여러 개의 약한 학습기를 순차적으로 학습-예측하면서 잘못 예측한 데이터에 가중치를 부여해 오류를 개선해 나가면서 학습하는 방식 ex) AdaBoost, 그래디언트부스트 p.222

에이다부스트 원리 p.223

GBM: 부스팅 알고리즘의 하나. 에이다부스트와 유사하나 가중치 업데이트를 경사 하강법을 이용하는 것이 큰 차이점. 분류, 회귀 다 가능.

경사하강법 : 오류식 h(x) = y − F(x)을 최소화하는 방향성으로 반복적으로 가중치 값을 업데이트하는 것. 회귀에서 다룰 예정

GradientBoostingClassifier : 사이킷런에서 제고하는 GBM 기반 분류 API

GBM이 랜덤포레스트보다 예측성능이 조금 뛰어난 경우가 많음. 그러나 수행 시간이 오래 걸리고 하이퍼 파라미터 튜닝 노력도 더 필요.

GBM 하이퍼 파라미터 및 튜닝

GBM의 하이퍼파라미터

: n_estimators, max_depth, max_features, loss, learning_rate, subsample, … p.225

GBM 등장 이후 많은 알고리즘들이 GBM을 기반으로 새롭게 만들어지고 있음. 머신러닝 세계에서 가장 각광 받고 있는 GBM 기반 패키지는 XGBoost와 LightGBM임.

6. XGBoost(eXtra Gradient Boost)

XGBoost 개요

XGBoost : 트리 기반 앙상블 학습에서 가장 각광받고 있는 알고리즘 중 하나. 분류에서 일반적으

로 다른 머신러닝보다 뛰어난 예측 성능을 나타냄. GBM 기반이지만 GBM의 단점인 느린 수행 시간 및 과적합 규제 부재 등의 문제를 해결해서 각광을 받음. 특히 병렬 CPU 환경에서 병렬학습이 가능해 기존 GBM보다 빠르게 학습을 완료. 자체적으로 교차 검증, 성능 평가, 피처 중요도 등의 시각화 기능을 가지고 있음.

조기 중단 기능이 있어 n_estimators에 지정한 부스팅 반복 횟수에 도달하지 않더라도 예측 오류가 더 이상 개선되지 않으면 반복을 끝까지 수행하지 않고 중지함. p.227, 233

XGBoost의 핵심 라이브러리는 C/C++로 작성되어 있고, 파이썬 패키지를 제공.

초기에는 xgboot 파이썬 패키지는 사이킷런과 호환되지 않는 패키지로 XGBoost 고유의 프레임워크를 파이썬 언어 기반에서 구현한 것으로 별도의 API였음. 그러나 파이썬 기방의 머신러닝 이용자들이 사이킷런을 많이 이용해 래퍼 클래스 제공.

XGBClassifier, XGBRegressor : XGBoost 패키지의 사이킷런 래퍼 클래스. 다른 Estimator과 사용법이 같음.

파이썬 네이티브 XGBoost는 고유의 API와 하이퍼 파라미터를 이용.

XGBoost 설치하기

파이썬 래퍼 XGBoost 하이퍼 파라미터 p.230

XGBoost는 GBM과 유사한 하이퍼 파라미터를 동일하게 가지고 있으며 여기에 조기 중단, 과적합 규제 위한 하이퍼 파라미터 등이 추가되었음.

파이썬 래퍼 XGBoost 하이퍼 파라미터 p.231*********

-일반 파라미터: 일반적으로 실행 시 스레드의 개수나 silent 모드 등의 선택을 위한 파라미터로서 디폴트 파라미터 값을 바꾸는 경우는 거의 없음

-부스터 파라미터: 대부분의 파라미터들이 속함. 트리 최적화, 부스팅, regularization 등과 관련 파라미터

-학습 태스크 파라미터: 학습 수행 시의 객체 함수, 평가를 위한 지표 등을 설정하는 파라미터

주요 일반 파라미터: booster, silent, nthread

주요 부스터 파라미터: eta, num_boost_rounds, min_child_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

뛰어난 알고리즘일수록 파라미터를 튜닝할 필요가 적고 공수 대비 성능 향상 효과가 높지 않은 경우가 많음.

파라미터를 튜닝하는 경우의 수는 여러 가지 상황에 따라 달라지는데, 피처의 수가 매우 많거나 피처 간 상관되는 정도가 많거나 데이터 세트에 따라 여러 가지 특성이 있을 수 있음.

과적합이 심각하다면?

- eta 값을 낮춘다(0.01~0.1). eta 값을 낮출 경우 num_round(or n_estimator)는 반대로 높여줘야 한다.

- max_depth 값을 낮춘다

- min_child_weight 값을 높인다

- gamma 값을 높인다

-subsample과 colsample_bytree를 조정하는 것도 트리가 너무 복잡하게 생성되는 것을 막아 과적합 문제에 도움이 될 수 있다.

파이썬 래퍼 XGBoost 적용 – 위스콘신 유방암 예측 p.234

plot_importance : xgboost에서 제공하는 피처 중요도 시각화 모듈

load_breast_cancer() : 위스콘신 유방암 데이터 세트. 다양한 피처들을 바탕으로 악성, 양성 종양 분류

파이썬 래퍼 XGBoost는 학습용과 테스트용 데이터 세트를 위해 별도의 객체인 DMatrix를 생성한다는 것이 큰 차이점 중 하나.

DMatrix : 넘파이 입력 파라미터를 받아서 만들어지는 XGBoost만의 전용 데이터 세트. data와 label이 주요 입력 파라미터. 회귀의 경우 label은 숫자형인 종속값 데이터 세트. 넘파이 외에도

libsvm txt 포맷 파일, xgboost 이진 버퍼 파일을 파라미터로 입력받아 변환할 수 있음.

판다스의 df로 데이터 인터페이스를 하기 위해서는 df.values를 이용해 넘파이로 일차 변환한 뒤에 DMatrix 변환을 적용해야 함.

XGBoost의 하이퍼 파라미터는 주로 딕셔너리 형태로 입력.

파이썬 래퍼 XGBoost는 하이퍼 파라미터를 xgboost 모듈의 train() 함수에 파라미터로 전달. 사이킷런의 경우는 Estimator 생성자를 하이퍼 파라미터로 전달하는 데 반해 차이가 있다.

조기 중단은 xgboost의 train() 함수에 early_stopping_rounds로 파라미터를 입력하여 설정. 조기 중단 수행 위서는 반드시 eval_set과 eval_metric이 함께 설정되어야 함. p.237

파이썬 래퍼 XGBoost는 train() 함수를 호출해 학습이 완료된 모델 객체를 반환하고, 이 모델 객체는 예측을 위해 predict() 메서드를 이용. xgboost의 predict()는 예측 결과를 추정할 수 있는 확률 값을 반환한다.

plot_importance() : xgboost에 내장되어 피처의 중요도를 막대그래프 형식으로 나타냄. f1 스코어 기반. 학습이 완료된 모델 객체 및 맷플롯립의 ax객체를 입력하면 됨. xgboost 넘파이 기반의 피처 데이터로 학습 시 피처명을 제대로 알 수가 없어 피처 순서별로 f자 뒤에 순서를 붙여서 X축에 피처들로 나열.

to_graphvis() : xgboost에서 결정 트리 트리 기반 규칙 구조를 보여주는 API. Graphviz 프로그램과 패키지가 설치되어 있어야 함. p.240

cv() : XGBoost에서 사이킷런의 GridSearchCV와 유사하게 데이터 세트에 대한 교차 검증 수행 후 최적 파라미터를 구할 수 있는 API. 반환값 df. -> 파라미터 설명 p.240

사이킷런 래퍼 XGBoost의 개요 및 적용

사이킷런의 기본 Estimator을 그대로 상속해 만들었기 때문에 다른 Estimator와 동일하게 fit()과

predict()만으로 학습과 예측이 가능하고, 사이킷런의 다른 유틸리티를 그대로 사용할 수 있음. 조기 중단 파라미터를 fit()에 입력하면 그대로 수행 가능.(성능 평가를 수행할 데이터 세트는 학습 데이터와 테스트 데이터 외의 별도의 데이터 세트여야 함) 파이썬 래퍼처럼 plot_importance() 그대로 적용 가능.

➔ XGBClassifier, XGBRegressor로 나뉨.

➔ 사이킷런에서 일반적으로 사용하는 하이퍼 파라미터와 호환성 유지를 위해 기존 xgboost 모듈에서 사용하던 네이티브 하이퍼 파라미터 몇 개를 변경함 p.241

- 조기 중단값을 너무 급격하게 줄이면 예측 성능이 저하될 우려가 큼.

7. LightGBM

LightGBM : XGBoost와 함께 부스팅 계열 알고리즘에서 가장 각광 받음. XGBoost보다 학습에 걸리는 시간이 훨씬 적으나 예측 성능은 별다른 차이가 없음. 오히려 기능 상 다양성은 더 많음. 단점은 적은 데이터 세트(보통 10000 이하)에 적용할 경우 과적합이 발생하기 쉽다는 것. 일반 GBM 계열의 트리 분할 방법과 다르게 리프 중심 트리 분할 방식 사용. 최대 손실 값을 가지는 리프 노드를 지속적으로 분할하면서 비대칭적인 규칙 트리 생성. 뛰어난 예측 성능 및 컴퓨팅 기능 제공하며 최근에는 GPU도 지원. p.245

lightgbm : LighGBM의 파이썬 패키지명

사이킷런 래퍼 LightGBM은 LGBMClassifier 클래스와 LGBMRegressor

이제 여기부터 XGBoost와 LightGBM 모두 사이킷런 래퍼 클래스만 사용할 것

LightGBM 설치

LightGBM 하이퍼 파라미터 p.247

LightGBM 하이퍼 파라미터는 XGBoost와 많은 부분이 유사. LightGBM은 Xgboost와 다르게 리프 노드가 계속 분할되면서 트리의 깊이가 깊어지므로 이러한 트리 특성에 맞는 하이퍼 파라미터 설정이 필요하다는 것 주의.

lightgbm의 주요 파라미터 p.247-248

- learning_rate, max_depth, min_data_in_leaf, num_leaves, boosting, bagging_fraction, feature_fraction, lambda_l2, lambda_l1

Learning Task 파라미터 p.248

- objective

하이퍼 파라미터 튜닝 방안

num_leaves의 개수를 중심으로 min_child_samples(min_data_in_leaf), max_depth를 함께 조정하면서 모델의 복잡도를 줄이는 것이 기본 튜닝 방안 p.249

learning_rate를 작게 하면서 n_estimators를 크게 하는 것은 부스팅 계열 튜닝에서 가장 기본적인 튜닝 방안이므로 이를 적용하는 것도 좋음.

그 외에도 reg_lambda, reg_alpha와 같은 regularization을 적용하거나 사용할 피처의 개수의 샘플링 레코드 개수를 줄이기 위해 colsamplt_bytree, subsample 파라미터를 적용할 수도 있음.

파이썬 래퍼 LightGBM과 사이킷런 래퍼 XGBoost, LightGBM 하이퍼 파라미터 비교 p.250

사이킷런 래퍼 LightGBM 클래스와 사이킷런 래퍼 XGBoost는 많은 하이퍼 파라미터가 동일.

LightGBM 적용 – 위스콘신 유방암 예측

LGBMClassifier의 fit()에 조기 중단 관련 파라미터 설정하면 조기 중단 가능.

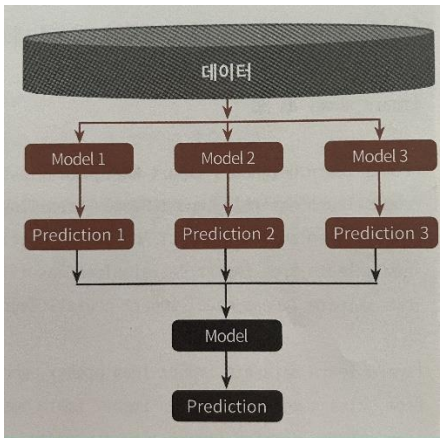
LightGBM의 파이썬 패키지인 lightgbm도 xgboost와 동일하게 피처 중요도 시각화 내장 API plot_importance() 제공. 사이킷런 래퍼 클래스도 입력 가능. 넘파이로 피처 데이터를 학습할 경우 피처명을 알 수 없기에 Column_뒤에 순서대로 숫자를 붙여서 X축에 나여러

10. 스태킹 앙상블

스태킹 : 여러 개별 알고리즘을 결합해 예측 결과를 도출한다는 점에서 배깅, 부스팅과 공통점을 가지지만 개별 알고리즘으로 예측한 데이터를 기반으로 다시 예측을 수행한다는 것이 차이점. 개별 알고리즘의 예측 결과 데이터 세트를 최종적인 메타 데이터 세트로 만들어 별도의 ML 알고리즘으로 최종 학습 수행. 개별적인 기반 모델과 최종 메타 모델, 필요. p.279

핵심은 여러 개별 모델의 예측 데이터를 각각 스태킹 형태로 결합해 최종 메타 모델의 학습용 피처 데이터 세트를 만드는 것.

- 메타 모델: 개별 모델의 예측된 데이터 세트를 기반으로 하여 학습 하고 예측하는 방식



기본 스태킹 모델

transpose() : 행과 열 위치를 바꾼 ndarray로 변환.

CV 세트 기반의 스태킹

CV 세트 기반의 스태킹 모델은 과적합을 개선하기 위한 교차 검증 기반으로 예측된 결과 데이터 세트를 이용. p.282

개별 모델에서 메타 모델인 2차 모델에서 사용될 학습용 데이터와 테스트용 데이터를 교차 검증을 통해 생성하는 것이 핵심.

********** 원리 설명 p.283-285