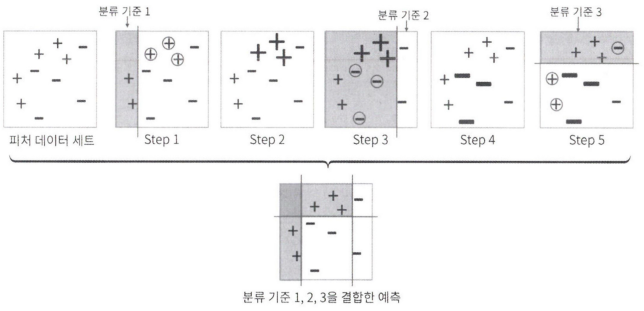
💻Week4_예습과제_이재린

태그	예습과제
주차	4주차

Ch2. 사이킷런으로 시작하는 머신러닝
▼ 05. GBM

GBM 개요 및 실습

- 부스팅 알고리즘 : 여러 개의 약한 학습기를 순차적으로 학습-예측하면서 잘못 예측한 데이터에 가중치를 부여를 통해 오류를 개선해 나가면서 학습하는 방법
- AdaBoost : 부스팅의 대표적인 구현, 오류 데이터에 가중치를 부여하면서 부스팅을 수행하는 알고리즘



- step1 : 약한 학습기 > 분류 기준 1로 +-로 분류(동그라미로 표시된 데이터는 + 데이터가 잘못 분류된 오류 데이터

- step2 : 오류 데이터에 가중치 부여 > 크기가 더 커진다

- step3,4 : 약한 학습기 > 분류 기준 2로 +-로 분류(동그라미로 표시된 데이터는 - 데이터가 잘못 분류된 오류 데이터 > 오류 데이터에 가중치 부여 > 크기가 더 커진다

- step5 : 약한 학습기 > 분류 기준 3로 +-로 분류(동그라미로 표시된 데이터는 - 데이터가 잘못 분류된 오류 데이터 > 오류 데이터에 가중치 부여 > 크기가 더 커진다 > 분류 기준으로 순차적인 오류 값에 대해 가중치 부여

- Last : 1+2+3 약한 학습기를 모두 결합한 결과 예측 > 정확도 증가

- GBM <> AdaBoost : 가중치 업데이트를 경사 하강법을 이용 > 오류 값(h(x))=실제값(y)-예측값(F(x))

  - 경사 하강법 : 오류식 h(x)를 최소화하는 방향성을 갖고 반복적으로 가중치 값을 업데이트하는 것

    - 분류 및 회귀도 가능
    - GradientBoostinClassfier 클래스 제공

      >> 기본 하이퍼 파라미터만으로도 랜덤 포레스트보다 더 나은 예측 성능 but 시간 오래 걸림

GBM 하이퍼 파라미터 및 튜닝

- loss: 경사 하강법에서 사용할 비용 함수를 지정합니다. 특별한 이유가 없으면 기본값인 'deviance'를 그대로 적용합니다.
- learning_rate: GBM이 학습을 진행할 때마다 적용하는 학습률입니다. Weak learner가 순차적으로 오류 값을 보정해 나가는 데 적용하는 계수입니다. 0~1 사이의 값을 지정할 수 있으며 기본값은 0.1입니다. 너무 작은 값을 적용하면 업데이트되는 값이 작아져서 최소 오류 값을 찾아 예측 성능이 높아질 가능성이 높습니다. 하지만 많은 weak learner는 순차적인 반복이 필요해서 수행 시간이 오래 걸리고, 또 너무 작게 설정하면 모든 weak learner의 반복이 완료돼도 최소 오류 값을 찾지 못할 수 있습니다. 반대로 큰 값을 적용하면 최소 오류 값을 찾지 못하고 그냥 지나쳐 버려 예측 성능이 떨어질 가능성이 높아지지만, 빠른 수행이 가능합니다.

  이러한 특성 때문에 learning_rate는 n_estimators와 상호 보완적으로 조합해 사용합니다. learning_rate를 작게 하고 n_estimators를 크게 하면 더 이상 성능이 좋아지지 않는 한계점까지는 예측 성능이 조금씩 좋아질 수 있습니다. 하지만 수행 시간이 너무 오래 걸리는 단점이 있으며, 예측 성능 역시 현격히 좋아지지는 않습니다.
- n_estimators: weak learner의 개수입니다. weak learner가 순차적으로 오류를 보정하므로 개수가 많을수록 예측 성능이 일정 수준까지는 좋아질 수 있습니다. 하지만 개수가 많을수록 수행 시간이 오래 걸립니다. 기본값은 100입니다.
- subsample: weak learner가 학습에 사용하는 데이터의 샘플링 비율입니다. 기본값은 1이며, 이는 전체 학습 데이터를 기반으로 학습한다는 의미입니다(0.5이면 학습 데이터의 50%). 과적합이 염려되는 경우 subsample을 1보다 작은 값으로 설정합니다.

- GridSearchCV를 이용해 하이퍼 파라미터 최적화 > n_estimators를 100, 500으로 learning_rate를 0.05, 0.1로만 계약, 교차 검증 세트도 2개로만

▼ 06. XGBoost

XGBoost 개요

- XGBoost란? 트리 기반의 앙상블 학습에서 가장 각광받고 있는 알고리즘 > 병렬 학습 가능
- 장점

  - 뛰어난 예측 성능

  - GBM 대비 빠른 수행 시간 (병렬 학습 가능)

  - 과적합 규제 : XGBoost 자체에 과적합 규제 기능

  - 나무 가지치기 : max_depth로 분할 깊이를 조정하기도 하지만, 긍정 이득이 없는 분할을 가지치기해서 분할 수를 줄인다

  - 자체 내장된 교차검증 : 평가 데이터 세트와 학습 데이터 세트에 대한 교차 검증을 수행해 최적화되면 반복을 조기에 중단할 수 있음.

  - 결손값 자체 처리
- XGBoost > 사이킷런과 연동할 수 있는 Wrapper class 제공함 > fit()과 predict() 활용 가능

XGBoost 설치하기

1. 맥북 기준 터미널로 XGBoost 설치하기

파이썬 wrapper XGBoost 하이퍼 파라미터

1. 일반 파라미터 : 스레드의 개수나 silent 모드 등의 선택을 위한 파라미터, 디폴트 값 거의 유지
- booster
- silent : 디폴트는 0, 출력 메세지를 나타내고 싶지 않으면 1로
- nthread : CPU 실행 스레드 개수 조정

2. 부스터 파라미터 : 트리 최적화, 부스팅, 정규화 등

- eta [default=0.3, alias: learning_rate]: GBM의 학습률(learning rate)과 같은 파라미터입니다. 0에서 1 사이의 값을 지정하며 부스팅 스텝을 반복적으로 수행할 때 업데이트되는 학습률 값. 파이썬 래퍼 기반의 xgboost를 이용할 경우 디폴트는 0.3. 사이킷런 래퍼 클래스를 이용할 경우 eta는 learning_rate 파라미터로 대체되며, 디폴트는 0.1입니다. 보통은 0.01 ~ 0.2 사이의 값을 선호합니다.
- num_boost_rounds: GBM의 n_estimators와 같은 파라미터입니다.
- min_child_weight[default=1]: 트리에서 추가적으로 가지를 나눌지를 결정하기 위해 필요한 데이터들의 weight 총합. min_child_weight이 클수록 분할을 자제합니다. 과적합을 조절하기 위해 사용됩니다.
- gamma [default=0, alias: min_split_loss]: 트리의 리프 노드를 추가적으로 나눌지를 결정할 최소 손실 감소 값입니다. 해당 값보다 큰 손실(loss)이 감소된 경우에 리프 노드를 분리합니다. 값이 클수록 과적합 감소 효과가 있습니다.
- max_depth[default=6]: 트리 기반 알고리즘의 max_depth와 같습니다. 0을 지정하면 깊이에 제한이 없습니다. Max_depth가 높으면 특정 피처 조건에 특화되어 룰 조건이 만들어지므로 과적합 가능성이 높아지며 보통은 3~10 사이의 값을 적용합니다.
- sub_sample[default=1]: GBM의 subsample과 동일합니다. 트리가 커져서 과적합되는 것을 제어하기 위해 데이터를 샘플링하는 비율을 지정합니다. sub_sample=0.5로 지정하면 전체 데이터의 절반을 트리를 생성하는 데 사용합니다. 0에서 1사이의 값이 가능하나 일반적으로 0.5 ~ 1 사이의 값을 사용합니다.

- colsample_bytree[default=1]: GBM의 max_features와 유사합니다. 트리 생성에 필요한 피처(칼럼)를 임의로 샘플링하는 데 사용됩니다. 매우 많은 피처가 있는 경우 과적합을 조정하는 데 적용합니다.
- lambda [default=1, alias: reg_lambda]: L2 Regularization 적용 값입니다. 피처 개수가 많을 경우 적용을 검토하며 값이 클수록 과적합 감소 효과가 있습니다.
- alpha [default=0, alias: reg_alpha]: L1 Regularization 적용 값입니다. 피처 개수가 많을 경우 적용을 검토하며 값이 클수록 과적합 감소 효과가 있습니다.
- scale_pos_weight [default=1]: 특정 값으로 치우친 비대칭한 클래스로 구성된 데이터 세트의 균형을 유지하기 위한 파라미터입니다.

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

- objective: 최솟값을 가져야할 손실 함수를 정의합니다. XGBoost는 많은 유형의 손실함수를 사용할 수 있습니다. 주로 사용되는 손실함수는 이진 분류인지 다중 분류인지에 따라 달라집니다.
- binary:logistic: 이진 분류일 때 적용합니다.
- multi:softmax: 다중 분류일 때 적용합니다. 손실함수가 multi:softmax일 경우에는 레이블 클래스의 개수인 num_class 파라미터를 지정해야 합니다.
- multi:softprob: multi:softmax와 유사하나 개별 레이블 클래스의 해당되는 예측 확률을 반환합니다.
- eval_metric: 검증에 사용되는 함수를 정의합니다. 기본값은 회귀인 경우는 rmse, 분류일 경우에는 error입니다. 다음은 eval_metric의 값 유형입니다.
  - 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

- 과적합 문제가 심하다면?

- eta 값을 낮춘다 (0.01~0.1) > num_round(or n_estimators) 는 반대로 높여준다

- max_depth 값을 낮춘다

- min_child_weight 높인다

- gamma 값을 높인다

- subsample, colsample_bytree 값 조정

파이썬 wrapper XGBoost 적용 - 위스콘신 유방암 예측

* XGBoost는 자체적으로 교차 검증, 성능 평가, 치퍼 중요도 등의 시각화 기능 존재함, 조기 중단 기능 존재 > num_rounds로 지정한 부스팅 반복 횟수에 도달하지 않더라도 예측 오류가 개선되지 않으면 반복 수행을 중지

* XGBoost는 병렬 처리&조기 중단 > 빠른 수행시간 처리 가능

* 위스콘신 유방암 데이터 세트 ? 종양의 크기, 모양 등 다양한 속성값을 기반으로 악성 종양인지 양성 종양인지 분류한 데이터 세트

part1.

1)위스콘신 유방암 데이터 세트를 로딩하고 피처의 중요도를 시각화

>타깃 레이블 값의 종류 : 악성 0(212개), 양성 1(357개)

2)데이터 세트 분할

3)학습 데이터 세트와 테스트 데이터 세트를 DMatrix로 변환 : df를 DataFrame.values를 이용해 넘파이로 일차 변환한 뒤 변환을 적용

4)딕셔너리 형태로 하이퍼 파라미터를 설정

- max_depth(트리 최대 깊이)는 3.
- 학습률 eta는 0.1(XGBClassifier를 사용할 경우 eta가 아니라 learning_rate입니다).
- 예제 데이터가 0 또는 1 이진 분류이므로 목적함수(objective)는 이진 로지스틱(binary:logistic).
- 오류 함수의 평가 성능 지표는 logloss.
- num_rounds(부스팅 반복 횟수)는 400회.

5)하이퍼 파라미터로 모델 학습

- 조기 중단 : train() 함수에 early_stopping_rounds 파라미터를 입력하여 설정 > 이를 위해서 eval_set과 eval_metric이 함께 설정돼야함 > 반복마다 두 지정된 평가 지표로 예측 오류를 측정
- 결과 : train-error와 eval-logloss가 지속적으로 감소

6)train() 함수를 호출해 학습이 완료된 모델 객체 반환 > 테스트 데이터 세트에 예측을 수행 by predict() > 0과 1이 아닌 확률값으로 나오므로 0.5 기준으로 0과 1로 표시하도록

part2. 시각화 기능 수행

1. 막대그래프

- plot_importance() API로 피처의 중요도를 바로 막대그래프 형식으로 나타내기 <> 사이킷런의 feature_importances_
- ax 객체만 입력하면 됨
- 피처명 알기 위해 피처 순서별로 f자를 붙여 X축에 피처들로 나열

2. 규칙 트리 구조

- to_graphiz() API를 이용
- xgboost.to_graph-viz() 내에 파라미터로 학습이 완료된 모델 객체와 Graphviz가 참조할 파일명 입력

- params (dict): 부스터 파라미터
- dtrain (DMatrix): 학습 데이터
- num_boost_round (int): 부스팅 반복 횟수
- nfold (int): CV 폴드 개수.
- stratified (bool): CV 수행 시 층화 표본 추출(stratified sampling) 수행 여부
- metrics (string or list of strings): CV 수행 시 모니터링할 성능 평가 지표
- early_stopping_rounds (int): 조기 중단을 활성화시킴. 반복 횟수 지정.

>>반환값은 DataFrame 형태

사이킷런 래퍼 XGBoost의 개요 및 적용

- XGBoost 래퍼 클래스 : 사이킷런의 프레임워크와 연동하기 위해 개발 > XGBClassifier, XGBRegressor
- 기존 하이퍼 파라미터의 변경 : eta > learing_rate, sub_sample > subsample, lambda > reg_lambda, alpha > reg_alpha
- n_estimators vs num_boost_round : 동일한 파라미터 but 사이킷런 래퍼 클래스 vs XGBoost API
- 사이킷런 래퍼 XGBoost에서도 조기 중단을 수행 가능 > 조기 중단 파라미터를 fit()에 입력 > 반복 횟수를 정의early_stopping_rounds, 조기 중단 평가 지표 eval_metric, 성능 평가 수행 데이터 세트 eval_set

*성능 평가 수행 데이터 세트는 학습 데이터와 별개의 데이터 세트

.

.