xgboost

Distributed(Deep) Machine Learning Community(DMLC) group 내 Tianqi Chen에 의한 연구 프로젝트로 시작. the Higgs Machine Learning Challenge 라는 Kaggle challenge에서 우승 solution으로 선정되어 알려졌고, 이후 xgboost를 이용하여 다른 Kaggle 경쟁에서도 다수 우승 (2015년동안 29개 Kaggle challenge 에서 17개의 solution 이 XGBoost를 사용)

https://m.blog.naver.com/PostView.nhn?blogId=tjdudwo93&logNo=221071886633&proxyReferer=https:%2F%2Fwww.google.com%2F

https://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/

xgboost: 앙상블 학습기법으로 모델을 생성하는 분류모델

[표 15.6] 배깅과 부스팅 알고리즘 비교

분류	배깅(Bagging)	부스팅(Boosting)
공통점	전체 데이터 셋으로부터 복원추출방식. n개의 학습데이터 셋을 생성하는 부트스트랩 방식 사용	
차이점	병렬학습방식: n개의 학습 데이터셋으로부터 n개의 트리 모델 생성	순차학습방식: 첫번째 학습 데이터 셋으로부터 트리모델 → 두번째 학습 데이터 셋으로부터 트리 모델 생성
특징	전체 데이터 셋을 균등하게 추출	분류하기 어려운 데이터 셋 생성
강점	과적합에 강함	높은 정확도
약점	특정 영역의 정확도 낮음	이상치(outlier)에 취약
모델	랜덤 포레스트	Xgboost

xgboost는 부스팅 방식을 기반으로 만들어진 모델
분류하기 어려운 특정 영역에 초점을 두고 정확도를 높이는 알고리즘
높은 정확도가 가장 큰 강점
배깅과 동일하게 복원 추출방식으로 첫 번째 학습 데이터 셋을 생성하는 방법은 동일
두 번째부터는 학습된 트리 모델의 결과를 바탕으로 정확도가 낮은 영역에 높은 가중치를 적용하여 해당 영역을 학습 데이터 셋으로 구성
즉, 기계학습이 안 되는 데이터셋을 집중적으로 학습하여 트리 모델의 정확도를 높이는 방식

실습 (다항 분류 xgboost 모델 생성)

xgboost패키지 내 xgboost()함수 이용

형식: xgboost(data = NULL, label = NULL, missing = NULL, params = list(), nrounds, verbose = 1, print.every.n = 1L, early.stop.round = NULL, maximize = NULL, ...)
where
objective: y변수가 이항("binary:logistic") 또는 다항("multi:softmax") 지정
max_depth: tree의 깊이 지정(tree-2인 경우 간단한 트리 생성)
nthread: 쳬녀사용 수 지정
nrounds: 반복 학습 수 지정
eta: 학습률 지정(기본값 0.3: 숫자가 낮을수록 모델의 복잡도가 높아지고, 컴퓨팅 파워가 많아짐)
verbose: 메시지 출력 여부. 0: 메시지 출력 안함, 1: 메시지 출력)

https://www.rdocumentation.org/packages/xgboost/versions/0.4-4/topics/xgboost

1단계: 패키지 설치

```
install.packages("xgboost")
library(xgboost)
```

xgboost 패키지

2단계: y변수 생성

```
iris_label <- ifelse(iris$Species == 'setosa', 0,
                     ifelse(iris$Species == 'versicolor', 1, 2))
table(iris_label)
iris$label <- iris_label
```

3단계: data set 생성

```
idx <- sample(nrow(iris), 0.7 * nrow(iris))
train <- iris[idx, ]
test <- iris[-idx, ]
```

4단계: matrix 객체 변환

```
train_mat <- as.matrix(train[-c(5:6)])
dim(train_mat)

train_lab <- train$label
length(train_lab)
```

x변수는 matrix 객체로 변환. y변수는 label을 이용하여 설정

5단계: xgb.DMatrix 객체 변환

```
dtrain <- xgb.DMatrix(data = train_mat, label = train_lab)
```

xgb.DMatrix()함수: 학습데이터 생성
https://www.rdocumentation.org/packages/xgboost/versions/1.3.2.1/topics/xgb.DMatrix

6단계: model생성 – xgboost matrix 객체 이용

```
xgb_model <- xgboost(data = dtrain, max_depth = 2, eta = 1,
                     nthread = 2, nrounds = 2,
                     objective = "multi:softmax",
                     num_class = 3,
                     verbose = 0)
xgb_model
```

xgboost()함수: 트리모델 생성
https://www.rdocumentation.org/packages/xgboost/versions/0.4-4/topics/xgboost

7단계: test set 생성

```
test_mat <- as.matrix(test[-c(5:6)])
dim(test_mat)
test_lab <- test$label
length(test_lab)
```

8단계: model prediction

```
pred_iris <- predict(xgb_model, test_mat)
pred_iris
```

predict()함수
https://www.rdocumentation.org/packages/xgboost/versions/1.3.2.1/topics/predict.xgb.Booster

9단계: confusion matrix

```
table(pred_iris, test_lab)
```

10단계: 모델 성능평가1 – Accuracy

```
(19 + 13 + 12) / length(test_lab)
```

분류정확도

11단계: model의 중요 변수(feature)와 영향력 보기

```
importance_matrix <- xgb.importance(colnames(train_mat),
                                    model = xgb_model)
importance_matrix
```

xgb.importance() 함수
https://www.rdocumentation.org/packages/xgboost/versions/0.6-4/topics/xgb.importance

12단계: 중요 변수 시각화

```
xgb.plot.importance(importance_matrix)
```

xgb.plot.importance() 함수
https://www.rdocumentation.org/packages/xgboost/versions/0.4-4/topics/xgb.plot.importance