Homework 7: Ensemble Learning

Vietnam Test Scores

Parker Whitehead

10/27/2021

Importing the data from .csv Link To Dataset: https://www.kaggle.com/ngvietlg/vietnam-national-highschool-exam-score-2018 (https://www.kaggle.com/ngvietlg/vietnam-national-highschool-exam-score-2018)

```
df <- read.csv("Vietnam_Scores_2018.csv", header = TRUE)
```

Data Cleaning

```
#Data Contains unneeded cols, such as name, id, etc.
df <- df[,c(3,4,6,7,8,9,10)]
#Data contains numerous students without scores in specific categories.
#To combat this, students that don't take all exams are omitted.
df <- na.omit(df)
#A new factor called "passed math" is added. This will be our target.
df$passedMath <- FALSE
df$passedMath[df$Math>=5] <- TRUE
df$passedMath <- factor(df$passedMath,
                        levels=c(FALSE,TRUE),
                        labels=c('Failed Math','Passed Math'))
#The original math column is removed.
df <- df[,c(2:8)]
```

Train/Test creation

```
set.seed(1234)
i = sample(1:nrow(df),nrow(df)*.75,replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
rf <- randomForest(passedMath~., data=train, importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = passedMath ~ ., data = train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 21.73%
## Confusion matrix:
##             Failed Math Passed Math class.error
## Failed Math       14260        1967   0.1212177
## Passed Math        3536        5562   0.3886568
```

```
pred_rf <- predict(rf, newdata=test)
acc <- length(which(pred_rf==test$passedMath)) / length(test$passedMath)
print(paste("acc: ",acc))
```

```
## [1] "acc:  0.787017294479981"
```

Boosting

```
library(adabag)
```

```
## Loading required package: rpart
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
## Loading required package: foreach
```

```
## Loading required package: doParallel
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
bst <- boosting(passedMath~., data=train, boos=TRUE, mfinal=20, coeflearn="Breiman")
summary(bst)
```

```
##            Length Class   Mode
## formula         3  formula call
## trees          20  -none-  list
## weights        20  -none-  numeric
## votes       50650  -none-  numeric
## prob        50650  -none-  numeric
## class       25325  -none-  character
## importance      6  -none-  numeric
## terms           3  terms   call
## call            6  -none-  call
```

```
pred_bst <- predict(bst, newdata=test, type="response")
acc <- length(which(pred_bst$class==test$passedMath)) / length(test$passedMath)
print(paste("acc: ",acc))
```

```
## [1] "acc:  0.791873963515755"
```

AdaBoost

```
library(fastAdaboost)
set.seed(1234)
adab <- adaboost(passedMath~., train, 10)
summary(adab)
```

```
##                   Length Class   Mode
## formula             3    formula call
## trees              10    -none-  list
## weights            10    -none-  numeric
## classnames          2    -none-  character
## dependent_variable  1    -none-  character
## call                4    -none-  call
```

```
pred_adab <- predict(adab, newdata=test, type="response")
acc <- length(which(pred_adab$class==test$passedMath)) / length(test$passedMath)
print(paste("acc: ",acc))
```

```
## [1] "acc:  0.74153044302298"
```

XGBoost

```
library(xgboost)
train_label <- ifelse(train$passedMath=="Passed Math", 1, 0)
train_matrix <- data.matrix(train[,c(1:6)])

test_label <- ifelse(test$passedMath=="Passed Math", 1, 0)
test_matrix <- data.matrix(test[,c(1:6)])

xgb <- xgboost(data=train_matrix, label=train_label, nrounds=200, objective="binary:logistic", v
erbose=FALSE)
```

```
## [13:43:32] WARNING: amalgamation/../src/learner.cc:1095: Starting in XGBoost 1.3.0, the defau
lt evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logl
oss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
probs_xgb <- predict(xgb, test_matrix)
pred_xgb <- ifelse(probs_xgb>0.5, 1, 0)

acc <- mean(pred_xgb==test_label)
print(paste("acc: ",acc))
```

```
## [1] "acc:  0.786661928452973"
```

Comparing Accuracies to Runtime:

Random Forest got very good acc (roughly 78%) but was by far the slowest of all algorithms used. This is most likely due to the many trees RF must create in order to make an accurate model.

Boosting had the best results out of all of the ensemble algorithms tried, but was also fairly slow, though not as slow as Random Forest.

Adaboost, although faster than both Random Forest and Boosting, was the worst performing by a decent margin, being only roughly 74% accurate.

XGBoost was by far the most impressive. It absolutely blew every other algorithm away with its speed, and was just below Boosting in accuracy.