CHS788_Final_Project

Will Bliss & Alexia Spoon

5/11/2021

```
# Set seed
set.seed(05102021)

# Read in data
pitches <- read.csv("pitches.csv")
at.bat <- read.csv("atbats.csv")

# Merge files by at bat ID, remove NAs, and look at 2015
game.info <- merge(pitches, at.bat, "ab_id")
game.info <- game.info[!(game.info$pitch_type == ""), ]
game.info <- game.info[game.info$ab_id < 2016000000, ]

# Choose one pitcher
uniqv <- unique(game.info$pitcher_id)
pitcher <- uniqv[which.max(tabulate(match(game.info$pitcher_id, uniqv)))]
game.info.pitcher <- game.info[game.info$pitcher_id == pitcher, ]
game.info.pitcher$pitcher_id <- NULL

## Create a run differential feature (p_score - b_score)
game.info.pitcher$run_differential <- game.info.pitcher$p_score - game.info.pitcher$b_score

## Remove p_score and b_score features
game.info.pitcher <- game.info.pitcher[,-which(colnames(game.info.pitcher) == 'b_score')]
game.info.pitcher <- game.info.pitcher[,-which(colnames(game.info.pitcher) == 'p_score')]

# Change L to 0 and R to 1 (stand)
game.info.pitcher$stand <- ifelse(game.info.pitcher$stand == "R", 1, 0)

# Remove irrelevant features
game.info.pitcher$ab_id <- NULL
game.info.pitcher$px <- NULL
game.info.pitcher$pz <- NULL
game.info.pitcher$start_speed <- NULL
game.info.pitcher$end_speed <- NULL
game.info.pitcher$spin_rate <- NULL
game.info.pitcher$spin_dir <- NULL
game.info.pitcher$break_angle <- NULL
game.info.pitcher$break_length <- NULL
game.info.pitcher$break_y <- NULL
game.info.pitcher$ax <- NULL
game.info.pitcher$ay <- NULL
game.info.pitcher$az <- NULL
```

```
game.info.pitcher$sz_bot <- NULL
game.info.pitcher$sz_top <- NULL
game.info.pitcher$vx0 <- NULL
game.info.pitcher$vy0 <- NULL
game.info.pitcher$vz0 <- NULL
game.info.pitcher$x <- NULL
game.info.pitcher$x0 <- NULL
game.info.pitcher$y <- NULL
game.info.pitcher$y0 <- NULL
game.info.pitcher$z0 <- NULL
game.info.pitcher$pfx_x <- NULL
game.info.pitcher$pfx_z <- NULL
game.info.pitcher$nasty <- NULL
game.info.pitcher$code <- NULL
game.info.pitcher$zone <- NULL
game.info.pitcher$event_num <- NULL
game.info.pitcher$batter_id <- NULL
game.info.pitcher$event <- NULL
game.info.pitcher$g_id <- NULL
game.info.pitcher$o <- NULL
game.info.pitcher$p_throws <- NULL
game.info.pitcher$type <- NULL
game.info.pitcher$type_confidence <- NULL

# Check how many instances of each pitch_type
length(game.info.pitcher[game.info.pitcher$pitch_type == "CH",1])
```

```
## [1] 473
```

```
length(game.info.pitcher[game.info.pitcher$pitch_type == "FC",1])
```

```
## [1] 185
```

```
length(game.info.pitcher[game.info.pitcher$pitch_type == "FF",1])
```

```
## [1] 398
```

```
length(game.info.pitcher[game.info.pitcher$pitch_type == "FT",1])
```

```
## [1] 1754
```

```
length(game.info.pitcher[game.info.pitcher$pitch_type == "SL",1])
```

```
## [1] 676
```

```
# For future adaboost, fastballs 1 otherwise 0
game2 <- game.info.pitcher
game2$pitch_type <- replace(game2$pitch_type, game2$pitch_type=="CH", "O")
game2$pitch_type <- replace(game2$pitch_type, game2$pitch_type=="FC", "F")
game2$pitch_type <- replace(game2$pitch_type, game2$pitch_type=="FF", "F")
game2$pitch_type <- replace(game2$pitch_type, game2$pitch_type=="FT", "F")
game2$pitch_type <- replace(game2$pitch_type, game2$pitch_type=="SL", "O")

game2$top <- replace(game2$top, game2$top=="True", 1)
game2$top <- replace(game2$top, game2$top=="False", 0)
game2$top <- as.numeric(game2$top)
```

```r
library(MASS)

# LDA
game<-game2[sample(nrow(game2)),]
folds <- cut(seq(1,nrow(game)),breaks=10,labels=FALSE)
keep.Error <- rep(NA, 10)
for(i in seq(1,10,1)){
  testIndexes <- which(folds==i)
  testData <- game[testIndexes, ]
  trainData <- game[-testIndexes, ]
  lda.train <- lda(x = trainData[,-1], grouping = trainData[,1])
  keep.Error[i] <- mean(predict(lda.train,testData[,-1])$class != testData[,1])
}
keep.Error
```

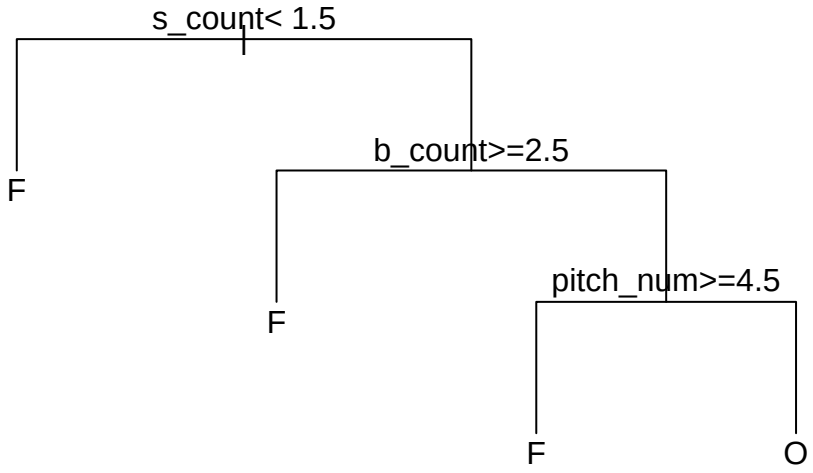
```
##  [1] 0.3696275 0.3438395 0.2844828 0.2922636 0.3764368 0.3381089 0.3275862
##  [8] 0.2951289 0.2931034 0.3295129
```

```r
mean(keep.Error)
```

```
## [1] 0.3250091
```

```r
library(rpart)

# Classification tree
folds <- cut(seq(1,nrow(game2)),breaks=10,labels=FALSE)
keep.Error <- rep(NA, 10)
for(i in seq(1,10,1)){
  testIndexes <- which(folds==i)
  testData <- game2[testIndexes, ]
  trainData <- game2[-testIndexes, ]
  bi.tr <- rpart(pitch_type ~., data = trainData, method="class")
  keep.Error[i] <- mean(predict(bi.tr,testData, type = "class") != testData[,1])
}
plot(bi.tr, uniform = TRUE, margin = 0.1)
text(bi.tr)
```




```r
keep.Error
```

```
##  [1] 0.2550143 0.3467049 0.2586207 0.3381089 0.2701149 0.2951289 0.3045977
```

```
##  [8] 0.3954155 0.3821839 0.3495702
```

```
mean(keep.Error)
```

```
## [1] 0.319546
```

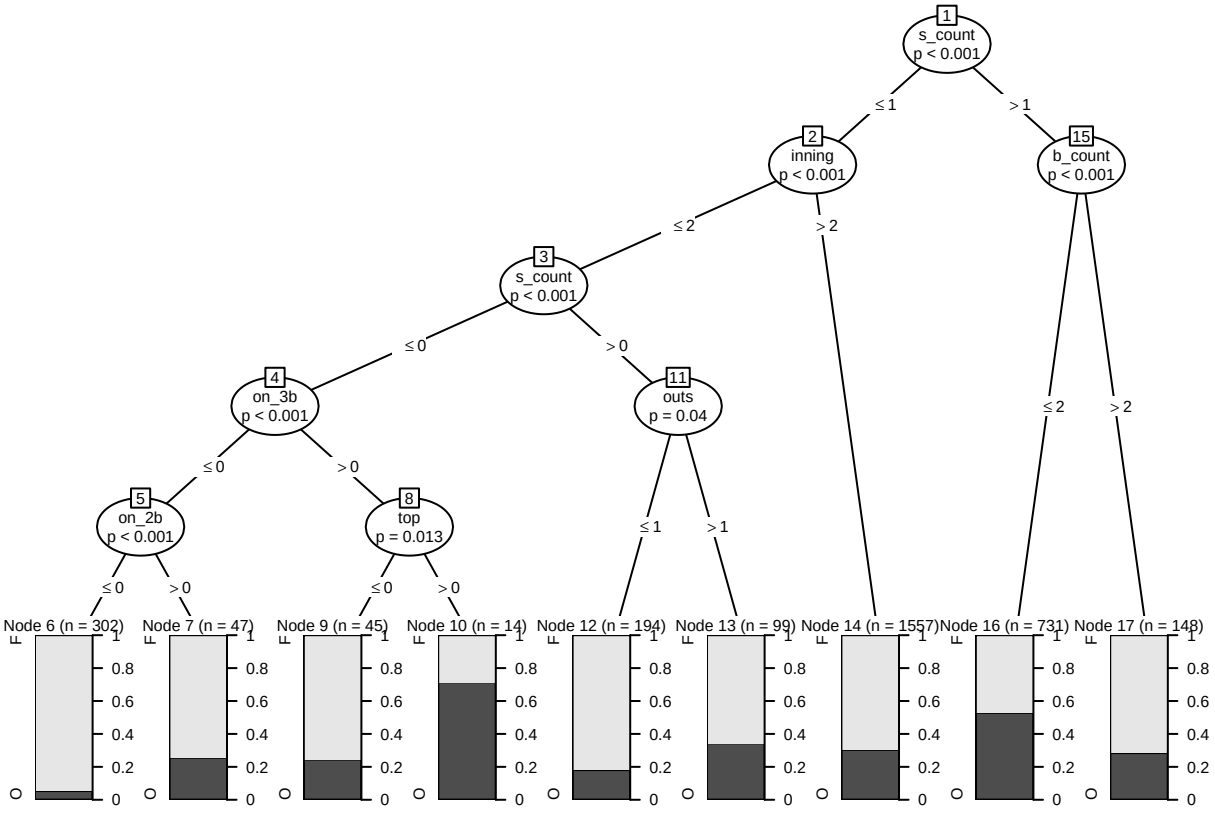
```
library(partykit)
```

```
## Loading required package: grid
```

```
## Loading required package: libcoin
```

```
## Loading required package: mvtnorm
```

```
# Conditional Inference Tree
folds <- cut(seq(1,nrow(game2)),breaks=10,labels=FALSE)
keep.Error <- rep(NA, 10)
for(i in seq(1,10,1)){
  testIndexes <- which(folds==i)
  testData <- game2[testIndexes, ]
  trainData <- game2[-testIndexes, ]
  bi.ct <- ctree(as.factor(pitch_type) ~., data = trainData)
  keep.Error[i] <- mean(predict(bi.ct,testData) != testData[,1])
}
plot(bi.ct, gp = gpar(fontsize = 6))
```




```
keep.Error
```

```
##  [1] 0.2636103 0.3495702 0.2701149 0.3381089 0.2643678 0.2951289 0.3045977
##  [8] 0.3954155 0.3419540 0.3151862
```

```r
mean(keep.Error)
```

```
## [1] 0.3138055
```

```r
# Random Forest
library(randomForest)
```

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

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

```r
game_oob <- 1
game.rf.optimal <- randomForest(as.factor(pitch_type) ~ ., data = game2)
for (i in c(500, 1000)) {
  for (j in seq(1,11,1)) {
    game.rf <- randomForest(as.factor(pitch_type) ~ ., data = game2, ntree = i, mtry = j)
    if(game.rf$err.rate[i,1] < game_oob){
      game_oob <- game.rf$err.rate[i,1]
      game.rf.optimal <- game.rf
    }
    cat(c("ntree = ", i, " and mtry = ", j, '\n',
          round(game.rf$err.rate[i, 1]*100, digits = 2), "%", '\n'), sep = '')
  }
}
```

```
## ntree = 500 and mtry = 1
## 32.99%
## ntree = 500 and mtry = 2
## 32.73%
## ntree = 500 and mtry = 3
## 33.39%
## ntree = 500 and mtry = 4
## 34.39%
## ntree = 500 and mtry = 5
## 34.88%
## ntree = 500 and mtry = 6
## 35.54%
## ntree = 500 and mtry = 7
## 36.35%
## ntree = 500 and mtry = 8
## 35.77%
## ntree = 500 and mtry = 9
## 36.43%
## ntree = 500 and mtry = 10
## 36.46%
## ntree = 500 and mtry = 11
## 36.63%
## ntree = 1000 and mtry = 1
## 32.96%
## ntree = 1000 and mtry = 2
## 32.64%
## ntree = 1000 and mtry = 3
## 33.08%
## ntree = 1000 and mtry = 4
## 34.17%
## ntree = 1000 and mtry = 5
```

```
## 34.62%
## ntree = 1000 and mtry = 6
## 35.37%
## ntree = 1000 and mtry = 7
## 36.12%
## ntree = 1000 and mtry = 8
## 36.23%
## ntree = 1000 and mtry = 9
## 36.4%
## ntree = 1000 and mtry = 10
## 36.83%
## ntree = 1000 and mtry = 11
## 36.63%
```

```
cat(c("Combination with lowest OOB error: ", '\n',
      "mtry = ", game.rf.optimal$mtry, " and ntree = ", game.rf.optimal$ntree, '\n',
      "OOB = ", round(game_oob*100, digits = 2), "%"), sep = '')
```

```
## Combination with lowest OOB error:
## mtry = 2 and ntree = 1000
## OOB = 32.64%
```

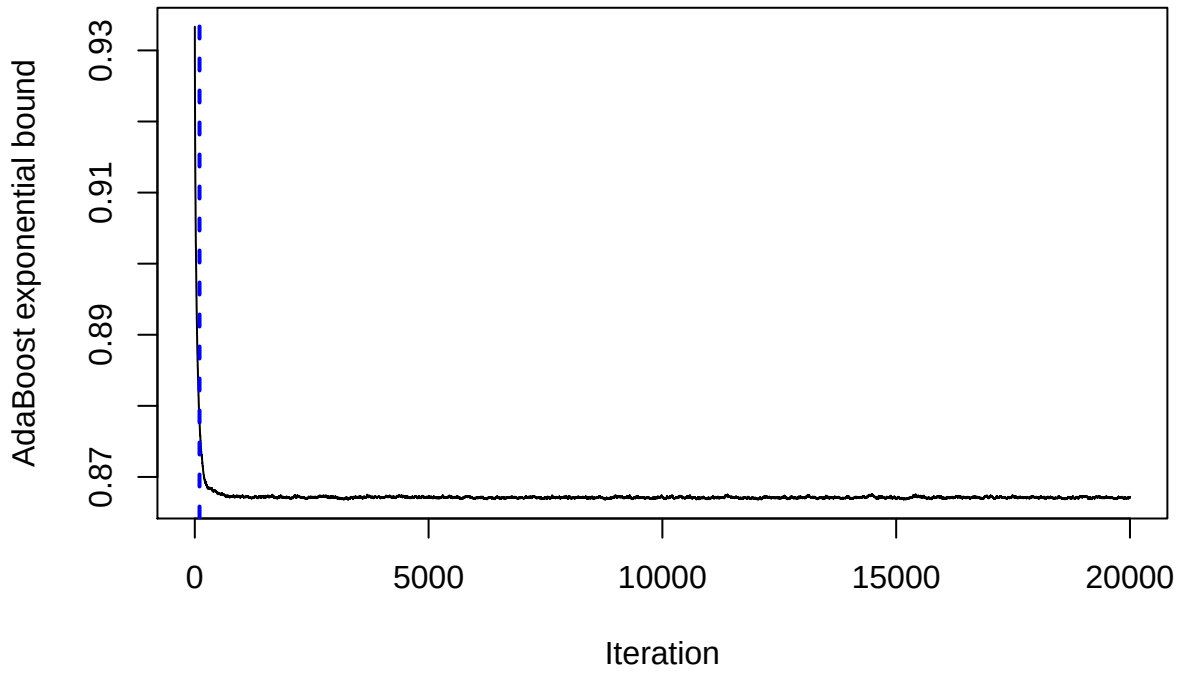
```
game3 <- game2
game3$pitch_type <- replace(game3$pitch_type, game3$pitch_type=="O", 0)
game3$pitch_type <- replace(game3$pitch_type, game3$pitch_type=="F", 1)

# Adaboost
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
indices <- sample(1:nrow(game3), 2*nrow(game3)/3)
game_train <- game3[indices,]
game_test <- game3[-indices,]
spam_adaboost <- gbm(pitch_type ~., data=game_train,
                     distribution="adaboost",
                     n.trees=20000)
optimal_trees <- gbm.perf(spam_adaboost, method = "OOB")
```

```
## OOB generally underestimates the optimal number of iterations although predictive performance is reas
```




```
optimal_trees[1]
```

```
## [1] 101
```

```
mean(ifelse(
  predict.gbm(spam_adaboost, newdata=game_test, n.trees=optimal_trees[1]) > 0,
  1, 0) != game_test[,1])
```

```
## [1] 0.3149742
```