Ensembler rcd180001

Joshua Durana

2022-10-23

Source: https://www.kaggle.com/datasets/danofer/law-school-admissions-bar-passage (https://www.kaggle.com/datasets/danofer/law-school-admissions-bar-passage)

Load and Clean Data

```
BARdata <- read.csv("Data/BarData.csv", header = TRUE, na.strings = c("", "NA"))

#Remove unneccessary factors
BARdata <- subset(BARdata, select = c(lsat, bar_passed, ugpa, fulltime, fam_inc, tier))

#Remove NAs
BARdata <- na.omit(BARdata)

#Sets columns into factors
setToFactors <- c("bar_passed", "fulltime", "fam_inc", "tier")
BARdata[setToFactors] <- lapply(BARdata[setToFactors], as.factor)
```

Split Data

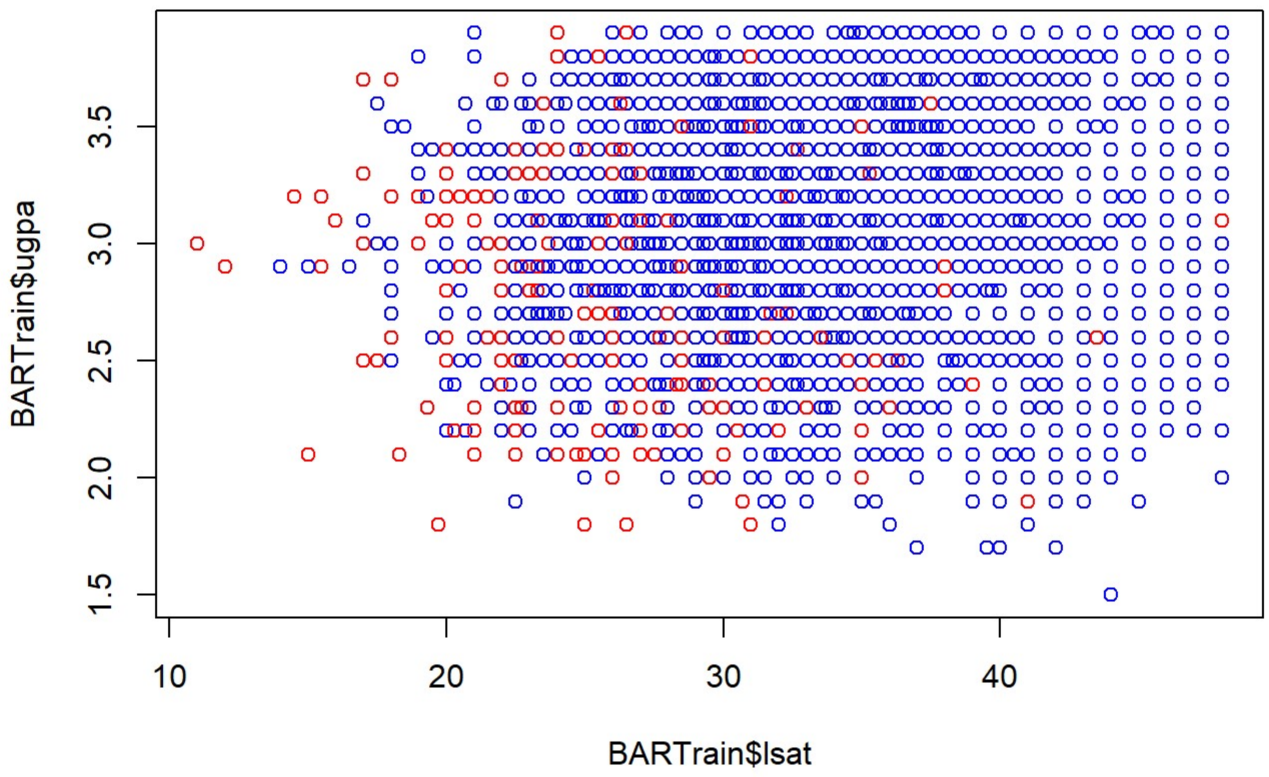
```
set.seed(1022)

i <- sample(nrow(BARdata), .80*nrow(BARdata), replace = FALSE)
BARTrain <- BARdata[i,]
BARTest <- BARdata[-i,]
```

Data Exploration

```
summary(BARTrain)
```

```
##       lsat       bar_passed        ugpa       fulltime  fam_inc  tier   
##  Min.   :11.00   FALSE:  903   Min.   :1.500   1:16297   1: 371   1: 485  
##  1st Qu.:33.00   TRUE :16696   1st Qu.:3.000   2: 1302   2:1775   2:1327  
##  Median :37.00                 Median :3.200             3:6267   3:6343  
##  Mean   :36.78                 Mean   :3.216             4:7758   4:4778  
##  3rd Qu.:41.00                 3rd Qu.:3.500             5:1428   5:3047  
##  Max.   :48.00                 Max.   :3.900                      6:1619  
```

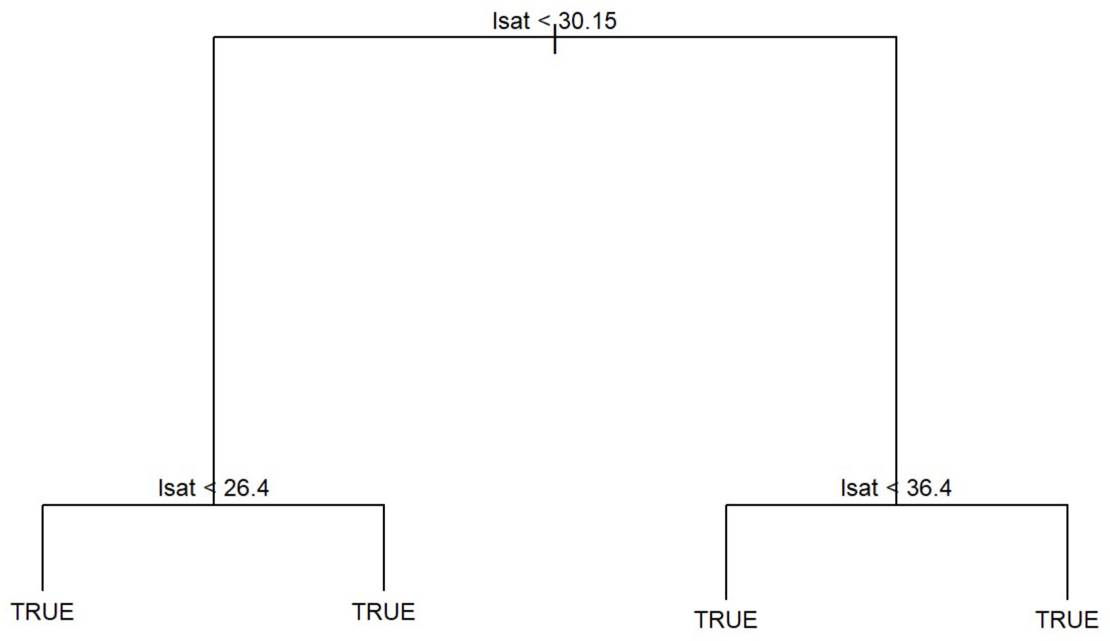
```
plot(BARTrain$lsat, BARTrain$ugpa, col = c("red", "blue") [unclass(BARTrain$bar_passed)])
```



The passed first time and failed instances seemed to have a good separation, but the passed 2nd time seems to be well mixed between both factors.

Decision Tree

```r
library(tree)


#Tree
BARTree <- tree(bar_passed~., data = BARTrain)

plot(BARTree)
text(BARTree, cex = .75, pretty = 0)
```



```
#Accuracy and MCC
BARTreePred <- predict(BARTree, newdata = BARTest, type = "class")
print(paste("accuracy = ", mean(BARTreePred == BARTest$bar_passed)))
```

```
## [1] "accuracy =  0.947727272727273"
```

The decision tree seems to have overfitted and can be pruned. But, the model seemed to be very accurate.

Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```r
library(mltools)

#Random Forest
BARrf <- randomForest(bar_passed~., data = BARTrain, importance = TRUE)
BARrf
```

```
##
## Call:
##  randomForest(formula = bar_passed ~ ., data = BARTrain, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 5.3%
## Confusion matrix:
##       FALSE  TRUE class.error
## FALSE    59   844 0.934662237
## TRUE     89 16607 0.005330618
```

```r
#Predict
BARrfPred <- predict(BARrf, newData = BARTest, type = "response")

#Metrics
print(paste("accuracy = ", mean(BARrfPred == BARTest$bar_passed)))
```

```
## [1] "accuracy =  0.940223876356611"
```

```r
print(paste("MCC = ", mcc(factor(BARrfPred), BARTest$bar_passed)))
```

```
## [1] "MCC =  0.000735781079377906"
```

It has a lower accuracy than decision tree. Most likely due to using a subset of the data for each tree and the low amount of passed_2nd_time instances causing some inaccuracy. The MCC metric is pretty close to 0, which means that test and random forest seems to agree and disagree pretty randomly.

adabag

```r
library(adabag)
```

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

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

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

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

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

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

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

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

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

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

```
#Boosting
BARada <- boosting(bar_passed~., data = BARTrain, boos = TRUE)
summary(BARada)
```

```
##            Length Class   Mode
## formula        3  formula call
## trees        100  -none-  list
## weights      100  -none-  numeric
## votes      35198  -none-  numeric
## prob       35198  -none-  numeric
## class      17599  -none-  character
## importance     5  -none-  numeric
## terms          3  terms   call
## call           4  -none-  call
```

```
#Metrics
adaPred <- predict(BARada, newdata = BARTest, type = "response")
adaAcc <- mean(adaPred$class == BARTest$bar_passed)
adaMcc <- mcc(factor(adaPred$class), BARTest$bar_passed)

print(paste("Accuracy = ", adaAcc))
```

```
## [1] "Accuracy =  0.946363636363636"
```

```
print(paste("MCC = ", adaMcc))
```

```
## [1] "MCC =  0.122457232906578"
```

The model seems to be very accurate most likely due to using the entire training set unlike random forest. The MCC number is 15% which is higher than random forest, so the model and test data seem to slightly agree.

XGBoost

```
library(xgboost)

#Convert to one-hot and matrix
trainLabel <- ifelse(BARTrain$bar_passed=="TRUE", 1, 0)
trainMatrix <- data.matrix(BARTrain[,c(-2)])

#XGBoost
BARXGBoost <- xgboost(data = trainMatrix, label = trainLabel, nrounds = 100)
```

```
## [1]  train-rmse:0.379628
## [2]  train-rmse:0.303179
## [3]  train-rmse:0.256923
## [4]  train-rmse:0.230496
## [5]  train-rmse:0.215973
## [6]  train-rmse:0.207959
## [7]  train-rmse:0.203567
## [8]  train-rmse:0.201043
## [9]  train-rmse:0.199628
## [10] train-rmse:0.198718
## [11] train-rmse:0.197844
## [12] train-rmse:0.197086
## [13] train-rmse:0.196972
## [14] train-rmse:0.196771
## [15] train-rmse:0.196716
## [16] train-rmse:0.196673
## [17] train-rmse:0.196188
## [18] train-rmse:0.195766
## [19] train-rmse:0.195695
## [20] train-rmse:0.195557
## [21] train-rmse:0.194804
## [22] train-rmse:0.194441
## [23] train-rmse:0.194268
## [24] train-rmse:0.193777
## [25] train-rmse:0.193481
## [26] train-rmse:0.192825
## [27] train-rmse:0.192491
## [28] train-rmse:0.191684
## [29] train-rmse:0.191365
## [30] train-rmse:0.191104
## [31] train-rmse:0.190897
## [32] train-rmse:0.190589
## [33] train-rmse:0.190409
## [34] train-rmse:0.190185
## [35] train-rmse:0.189759
## [36] train-rmse:0.189543
## [37] train-rmse:0.189311
## [38] train-rmse:0.189016
## [39] train-rmse:0.188785
## [40] train-rmse:0.188407
## [41] train-rmse:0.188290
## [42] train-rmse:0.188120
## [43] train-rmse:0.188049
## [44] train-rmse:0.188013
## [45] train-rmse:0.187972
## [46] train-rmse:0.187440
## [47] train-rmse:0.187165
## [48] train-rmse:0.186898
## [49] train-rmse:0.186484
## [50] train-rmse:0.186411
## [51] train-rmse:0.186342
## [52] train-rmse:0.186285
## [53] train-rmse:0.186250
## [54] train-rmse:0.186179
## [55] train-rmse:0.185781
## [56] train-rmse:0.185616
## [57] train-rmse:0.185376
## [58] train-rmse:0.185259
## [59] train-rmse:0.185036
## [60] train-rmse:0.184693
## [61] train-rmse:0.184422
## [62] train-rmse:0.184137
## [63] train-rmse:0.183960
## [64] train-rmse:0.183813
## [65] train-rmse:0.183607
## [66] train-rmse:0.183454
## [67] train-rmse:0.183345
## [68] train-rmse:0.183154
## [69] train-rmse:0.183090
## [70] train-rmse:0.183080
## [71] train-rmse:0.183071
## [72] train-rmse:0.182891
## [73] train-rmse:0.182830
## [74] train-rmse:0.182770
## [75] train-rmse:0.182439
## [76] train-rmse:0.182028
## [77] train-rmse:0.181788
## [78] train-rmse:0.181623
## [79] train-rmse:0.181474
## [80] train-rmse:0.181338
## [81] train-rmse:0.181298
## [82] train-rmse:0.181256
## [83] train-rmse:0.181216
## [84] train-rmse:0.180827
## [85] train-rmse:0.180528
## [86] train-rmse:0.180253
## [87] train-rmse:0.180212
## [88] train-rmse:0.180028
## [89] train-rmse:0.179797
## [90] train-rmse:0.179591
## [91] train-rmse:0.179362
## [92] train-rmse:0.179247
## [93] train-rmse:0.179167
## [94] train-rmse:0.178916
## [95] train-rmse:0.178842
## [96] train-rmse:0.178729
## [97] train-rmse:0.178645
## [98] train-rmse:0.178609
## [99] train-rmse:0.178319
## [100]    train-rmse:0.178008
```

```
#Convert to one-hot and matrix
testLabel <- ifelse(BARTest$bar_passed=="TRUE", 1, 0)
testMatrix <- data.matrix(BARTest[,c(-2)])

#Prediction
xPred <- predict(BARXGBoost, testMatrix, type = "response")
xPredOut <- ifelse(xPred>.5, 1, 0)

#Mean and MCC
print(paste("Accuracy = ", mean(xPredOut == testLabel)))
```

```
## [1] "Accuracy =  0.941818181818182"
```

```
print(paste("MCC = ", mcc(xPredOut, testLabel)))
```

```
## [1] "MCC =  0.116102324658213"
```

The accuracy is high, but adaboost is slightly more accurate. Adaboost’s mcc is also slighty better than xgboost. But, it’s much more faster. So, I would rather use xgboost for bigger datasets.