Tree-Based\_Classification

Samuel A, Mikolaj W

April 17, 2020

PROBLEM 1 –– SATELLITE IMAGE DATA The goal here is to predict the type of ground cover from a satellite image broken up into pixels. Description from UCI Machine Learning database: The database consists of the multi-spectral values of pixels in 3x3 neighborhoods in a satellite image, and the classification associated with the central pixel in each neighborhood. The aim is to predict this classification, given the multi-spectral values. In the sample database, the class of a pixel is coded as a number.

The Landsat satellite data is one of the many sources of information available for a scene. The interpretation of a scene by integrating spatial data of diverse types and resolutions including multispectral and radar data, maps indicating topography, land use etc. is expected to assume significant importance with the onset of an era characterized by integrative approaches to remote sensing (for example, NASA’s Earth Observing System commencing this decade). Existing statistical methods are ill-equipped for handling such diverse data types. Note that this is not true for Landsat MSS data considered in isolation (as in this sample database). This data satisfies the important requirements of being numerical and at a single resolution, and standard maximum-likelihood classification performs very well. Consequently, for this data, it should be interesting to compare the performance of other methods against the statistical approach.

One frame of Landsat MSS imagery consists of four digital images of the same scene in different spectral bands. Two of these are in the visible region (corresponding approximately to green and red regions of the visible spectrum) and two are in the (near) infra-red. Each pixel is a 8-bit binary word, with 0 corresponding to black and 255 to white. The spatial resolution of a pixel is about 80m x 80m. Each image contains 2340 x 3380 such pixels.

The database is a (tiny) sub-area of a scene, consisting of 82 x 100 pixels. Each line of data corresponds to a 3x3 square neighborhood of pixels completely contained within the 82x100 sub-area. Each line contains the pixel values in the four spectral bands (converted to ASCII) of each of the 9 pixels in the 3x3 neighborhood and a number indicating the classification label of the central pixel. The number is a code for the following classes:

Number Class 1 red soil 2 cotton crop 3 grey soil 4 damp grey soil 5 soil with vegetation stubble 6 mixture class (all types present)  
7 very damp grey soil

Note: There are no examples with class 6 in this dataset.

The data is given in random order and certain lines of data have been removed so you cannot reconstruct the original image from this dataset.

In each line of data the four spectral values for the top-left pixel are given first followed by the four spectral values for the top-middle pixel and then those for the top-right pixel, and so on with the pixels read out in sequence left-to-right and top-to-bottom. Thus, the four spectral values for the central pixel are given by attributes 17,18,19 and 20.

1. Compare CART/RPART, bagged CART/RPART, Random Forest classification, and Boosted Trees in the classification of the test cases. Which method performs best for these data? Be sure to adjust the various tuning parameters to optimize the performance of these methods for this prediction problem. Show the model development process for each of these methods and report the final settings you used for any tuning parameters. (20 pts. – 5 pts. for each method)

Getting started:

setwd(getwd())  
SATimage = read.csv("SATimage.csv")  
SATimage = data.frame(class=as.factor(SATimage$class),SATimage[,1:36])  
  
  
set.seed(888)  
testcases = sample(1:dim(SATimage)[1],1000,replace=F)  
SATtest = SATimage[testcases,]  
SATtrain = SATimage[-testcases,]

Packages:

require(rpart)  
require(rpart.plot)  
require(ipred)

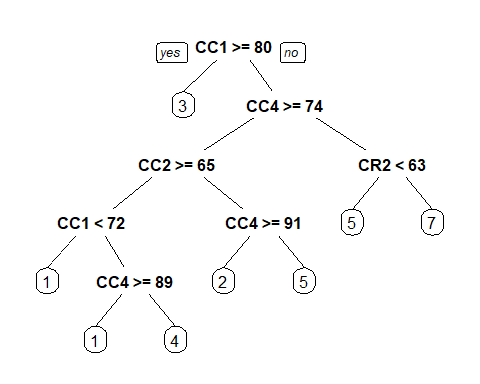
RPART basic Models:

Misclass Function:

misclass = function(fit,y) {  
temp <- table(fit,y)  
cat("Table of Misclassification\n")  
cat("(row = predicted, col = actual)\n")  
print(temp)  
cat("\n\n")  
numcor <- sum(diag(temp))  
numinc <- length(y) - numcor  
mcr <- numinc/length(y)  
cat(paste("Misclassification Rate = ",format(mcr,digits=3)))  
cat("\n")  
}

fitting a simple tree

mod.default = rpart(class~.,data=SATtrain)  
prp(mod.default)



phat = predict(mod.default,newdata=SATtest,type="prob")  
head(phat)

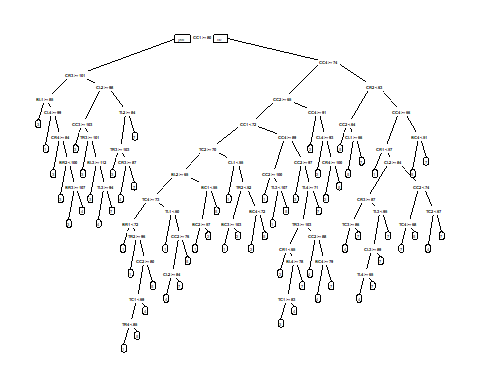
## 1 2 3 4 5 7  
## 1672 0.03687151 0.003351955 0.003351955 0.138547486 0.045810056 0.77206704  
## 4015 0.01869159 0.000000000 0.813084112 0.123831776 0.001168224 0.04322430  
## 3851 0.03687151 0.003351955 0.003351955 0.138547486 0.045810056 0.77206704  
## 1131 0.00000000 0.982248521 0.000000000 0.000000000 0.017751479 0.00000000  
## 1027 0.03687151 0.003351955 0.003351955 0.138547486 0.045810056 0.77206704  
## 1266 0.00000000 0.005181347 0.010362694 0.005181347 0.927461140 0.05181347

yhat = predict(mod.default,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 231 7 2 5 18 0  
## 2 0 88 0 0 2 0  
## 3 4 0 204 33 0 10  
## 4 1 2 15 23 0 18  
## 5 4 6 0 0 61 3  
## 7 12 2 1 31 21 196  
##   
##   
## Misclassification Rate = 0.197

Basic improvements

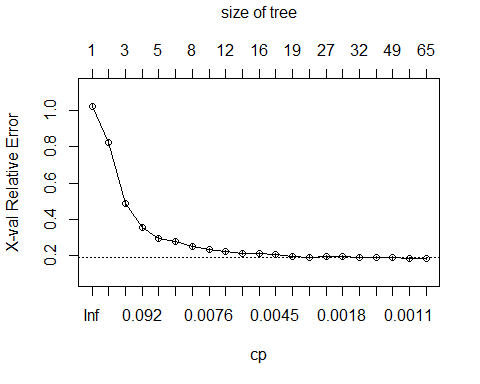
control = rpart.control(minsplit=5,minbucket=3,cp=.001)  
mod2 = rpart(class~.,data=SATtrain,control=control)  
prp(mod2)



yhat = predict(mod2,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 239 2 2 0 8 0  
## 2 0 99 0 0 5 0  
## 3 4 0 207 23 0 7  
## 4 2 1 10 49 0 19  
## 5 5 2 0 1 76 5  
## 7 2 1 3 19 13 196  
##   
##   
## Misclassification Rate = 0.134

plotcp(mod2)

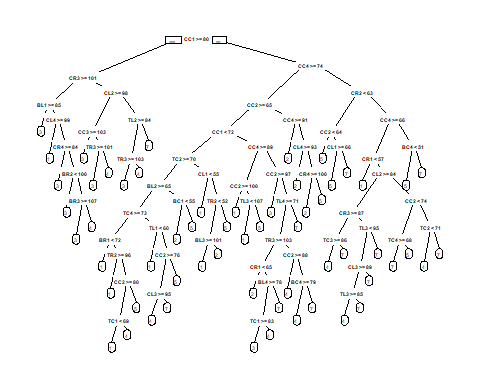


crpart.sscv = function(fit,y,data,B=25,p=.333) {  
n = length(y)  
cv <- rep(0,B)  
for (i in 1:B) {  
 ss <- floor(n\*p)  
 sam <- sample(1:n,ss)  
 temp <- data[-sam,]  
 fit2 <- rpart(formula(fit),data=temp,parms=fit$parms,control=fit$control)  
 ynew <- predict(fit2,newdata=data[sam,],type="class")  
 tab <- table(y[sam],ynew)  
 mc <- ss - sum(diag(tab))  
 cv[i] <- mc/ss  
 }  
 cv  
}  
results = crpart.sscv(mod2,SATimage$class,data=SATimage,B=50)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1274 0.1357 0.1419 0.1428 0.1491 0.1653

While this second model is an improvement, we should still see if we can’t push it further. Let’s walk it back and try a simpler model firdt with a greater requirement for splits and a larger bucket size.

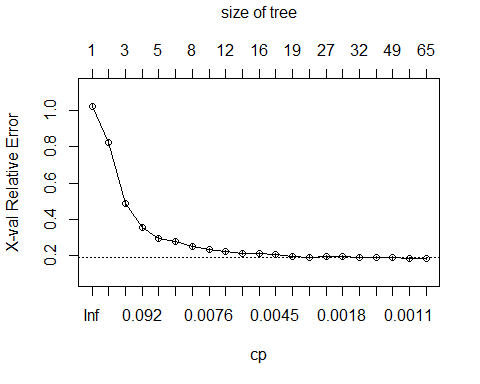
control = rpart.control(minsplit=7,minbucket=5,cp=.001)  
modx = rpart(class~.,data=SATtrain,control=control)  
prp(modx)



yhat = predict(modx,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 239 2 2 0 9 0  
## 2 1 99 0 0 4 0  
## 3 4 0 207 23 0 7  
## 4 2 1 10 47 1 18  
## 5 4 2 0 1 76 5  
## 7 2 1 3 21 12 197  
##   
##   
## Misclassification Rate = 0.135

plotcp(mod2)

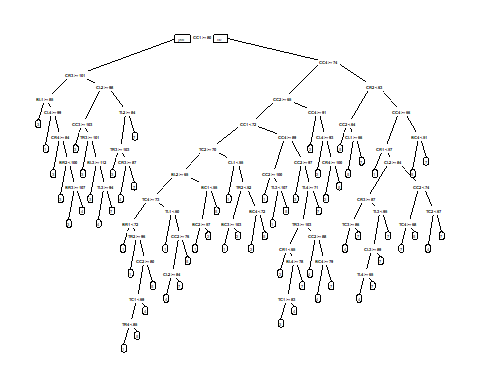


results = crpart.sscv(modx,SATimage$class,data=SATimage,B=50)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1192 0.1384 0.1446 0.1438 0.1491 0.1646

Very little changed, perhaps we need greater complexity. Perhaps by moving in the opposite direction with our metrics can find a better solution

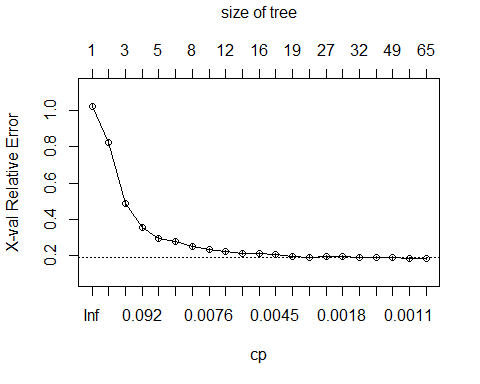
control = rpart.control(minsplit=3,minbucket=2,cp=.001)  
modx = rpart(class~.,data=SATtrain,control=control)  
prp(modx)



yhat = predict(modx,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 239 2 2 0 8 0  
## 2 0 99 0 0 5 0  
## 3 4 0 207 23 0 7  
## 4 2 1 10 49 0 19  
## 5 5 2 0 1 76 5  
## 7 2 1 3 19 13 196  
##   
##   
## Misclassification Rate = 0.134

plotcp(mod2)

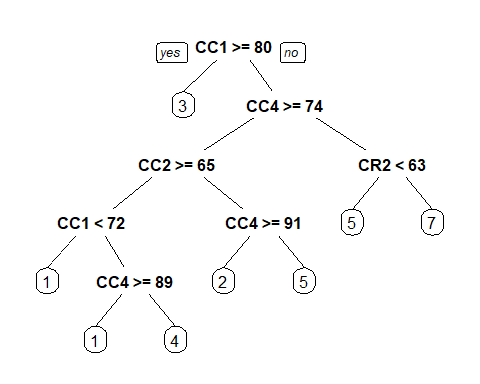


results = crpart.sscv(modx,SATimage$class,data=SATimage,B=50)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1287 0.1382 0.1426 0.1429 0.1475 0.1599

Adding in just a little more complexity, we have inched out a bit more on our performance on our training set whilst being just as good as mod2 on the test set. The only metric left to truly optimize may just be the complexity parameter, so we will try next, alough this currently is the best model we have for basic rpart.

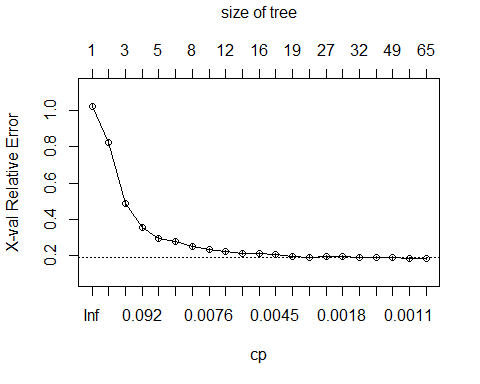
control = rpart.control(minsplit=3,minbucket=2,cp=.01)  
modx = rpart(class~.,data=SATtrain,control=control)  
prp(modx)



yhat = predict(modx,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 231 7 2 5 18 0  
## 2 0 88 0 0 2 0  
## 3 4 0 204 33 0 10  
## 4 1 2 15 23 0 18  
## 5 4 6 0 0 61 3  
## 7 12 2 1 31 21 196  
##   
##   
## Misclassification Rate = 0.197

plotcp(mod2)



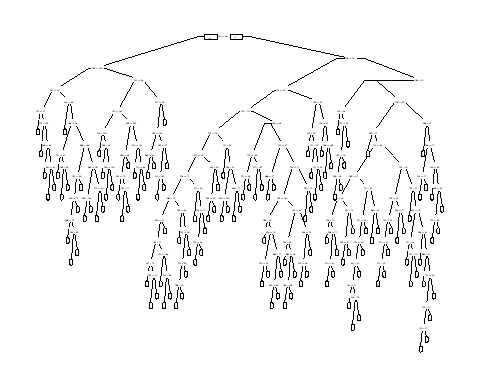
results = crpart.sscv(modx,SATimage$class,data=SATimage,B=50)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1680 0.1804 0.1873 0.1897 0.1965 0.2297

While clearly less is not more, perhaps more is more to at least a certain extant.

control = rpart.control(minsplit=3,minbucket=2,cp=.0005)  
modx = rpart(class~.,data=SATtrain,control=control)  
prp(modx)

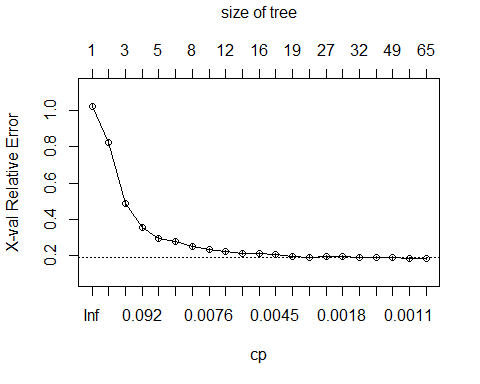
## Warning: labs do not fit even at cex 0.15, there may be some overplotting



yhat = predict(modx,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 239 2 2 0 7 0  
## 2 0 100 0 0 4 0  
## 3 4 0 202 20 0 4  
## 4 2 1 11 50 1 19  
## 5 5 2 0 1 76 7  
## 7 2 0 7 21 14 197  
##   
##   
## Misclassification Rate = 0.136

plotcp(mod2)

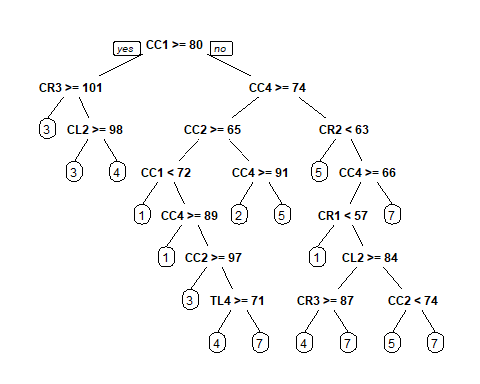


results = crpart.sscv(modx,SATimage$class,data=SATimage,B=50)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1321 0.1382 0.1436 0.1456 0.1518 0.1714

That was certainly a mistake. Perhaps we can find a better middle ground?

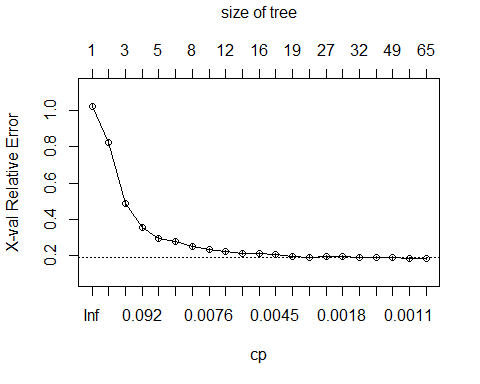
control = rpart.control(minsplit=3,minbucket=2,cp=.005)  
modx = rpart(class~.,data=SATtrain,control=control)  
prp(modx)



yhat = predict(modx,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 243 7 2 5 18 0  
## 2 0 88 0 0 2 0  
## 3 5 0 209 28 0 7  
## 4 0 2 10 42 0 26  
## 5 4 7 0 0 71 6  
## 7 0 1 1 17 11 188  
##   
##   
## Misclassification Rate = 0.159

plotcp(mod2)

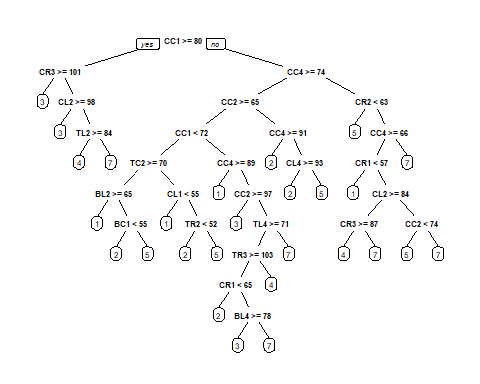


results = crpart.sscv(modx,SATimage$class,data=SATimage,B=50)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1423 0.1553 0.1596 0.1616 0.1684 0.1917

Still a massive loss compared to others, one final adjustment will be made to attemtp an improvment.

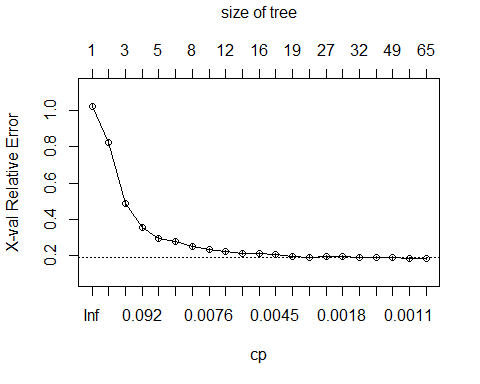
control = rpart.control(minsplit=3,minbucket=2,cp=.002)  
modx = rpart(class~.,data=SATtrain,control=control)  
prp(modx)



yhat = predict(modx,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 241 2 2 4 11 0  
## 2 1 97 0 0 5 0  
## 3 5 0 211 28 0 8  
## 4 0 0 6 40 0 19  
## 5 5 5 0 1 75 6  
## 7 0 1 3 19 11 194  
##   
##   
## Misclassification Rate = 0.142

plotcp(mod2)



results = crpart.sscv(modx,SATimage$class,data=SATimage,B=50)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1301 0.1438 0.1491 0.1492 0.1568 0.1626

Still no improvement, it seems more and more likely that for any other further improvement we will have to change our whole methodology. Still, for how simply it is to implement and understand, a missclassification rate of about 13% isn’t bad.

Final Model:

# control = rpart.control(minsplit=3,minbucket=2,cp=.001)  
# modx = rpart(class~.,data=SATtrain,control=control)

Metrics:

For Training (Mean): 0.1413

For Test: 0.134

RPART bagging Models:

Fit a basic model:

sat.bag = bagging(class~.,data=SATtrain,coob=T)  
sat.bag

##   
## Bagging classification trees with 25 bootstrap replications   
##   
## Call: bagging.data.frame(formula = class ~ ., data = SATtrain, coob = T)  
##   
## Out-of-bag estimate of misclassification error: 0.1121

check it:

phat = predict(sat.bag,newdata=SATtest,type="prob")  
head(phat)

## 1 2 3 4 5 7  
## [1,] 0 0 0.00 0.24 0 0.76  
## [2,] 0 0 0.92 0.08 0 0.00  
## [3,] 1 0 0.00 0.00 0 0.00  
## [4,] 0 1 0.00 0.00 0 0.00  
## [5,] 0 0 0.00 0.08 0 0.92  
## [6,] 0 0 0.00 0.00 1 0.00

yhat = predict(sat.bag,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 243 0 1 1 5 0  
## 2 0 104 0 0 1 0  
## 3 3 0 215 21 0 6  
## 4 0 1 4 51 0 14  
## 5 6 0 0 1 85 4  
## 7 0 0 2 18 11 203  
##   
##   
## Misclassification Rate = 0.099

An improvement already, lets push it further.

control = rpart.control(minsplit=5,minbucket=3,cp=0,xval=0)  
sat.bag2 = bagging(class~.,data=SATtrain,nbagg=100,coob=T,control=control)  
sat.bag2

##   
## Bagging classification trees with 100 bootstrap replications   
##   
## Call: bagging.data.frame(formula = class ~ ., data = SATtrain, nbagg = 100,   
## coob = T, control = control)  
##   
## Out-of-bag estimate of misclassification error: 0.1028

yhat = predict(sat.bag2,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 244 0 1 1 5 0  
## 2 0 101 0 0 1 0  
## 3 3 0 217 22 0 5  
## 4 0 0 2 53 0 13  
## 5 5 2 0 0 85 4  
## 7 0 2 2 16 11 205  
##   
##   
## Misclassification Rate = 0.095

Split-Sample function

bagg.sscv = function(fit,y,data,B=25,nbagg=100,p=.333) {  
n = length(y)  
cv <- rep(0,B)  
for (i in 1:B) {  
 ss <- floor(n\*p)  
 sam <- sample(1:n,ss)  
 temp <- data[-sam,]  
 fit2 <- bagging(formula(fit),data=temp,control=fit$control,coob=F)  
 ynew <- predict(fit2,newdata=data[sam,],type="class")  
 tab <- table(y[sam],ynew)  
 mc <- ss - sum(diag(tab))  
 cv[i] <- mc/ss  
 }  
 cv  
}

Using it:

results = bagg.sscv(mod2,SATimage$class,data=SATimage,B=5,nbagg=25)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1084 0.1131 0.1206 0.1210 0.1253 0.1375

While this is better, it doesn’t exactly blow us away in terms of succeeding over the normal bagging model. Lets see if we can optimize more.

control = rpart.control(minsplit=3,minbucket=2,cp=0,xval=0)  
sat.bag2 = bagging(class~.,data=SATtrain,nbagg=100,coob=T,control=control)  
sat.bag2

##   
## Bagging classification trees with 100 bootstrap replications   
##   
## Call: bagging.data.frame(formula = class ~ ., data = SATtrain, nbagg = 100,   
## coob = T, control = control)  
##   
## Out-of-bag estimate of misclassification error: 0.1001

yhat = predict(sat.bag2,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 244 0 1 1 5 0  
## 2 0 103 0 0 1 0  
## 3 3 0 217 22 0 4  
## 4 0 1 3 50 0 11  
## 5 5 0 0 0 86 3  
## 7 0 1 1 19 10 209  
##   
##   
## Misclassification Rate = 0.091

results = bagg.sscv(mod2,SATimage$class,data=SATimage,B=5,nbagg=25)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1098 0.1131 0.1138 0.1152 0.1152 0.1240

Using our bucket and split settings from the previous model made for better predictions on the test set, but worse on split-sample mean. While we personally would prefer this model, we are still hesistant to call it better. Maybe the complexity parameter can forward it.

control = rpart.control(minsplit=3,minbucket=2,cp=0.001,xval=0)  
sat.bag2 = bagging(class~.,data=SATtrain,nbagg=100,coob=T,control=control)  
sat.bag2

##   
## Bagging classification trees with 100 bootstrap replications   
##   
## Call: bagging.data.frame(formula = class ~ ., data = SATtrain, nbagg = 100,   
## coob = T, control = control)  
##   
## Out-of-bag estimate of misclassification error: 0.1138

yhat = predict(sat.bag2,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 245 0 2 2 6 0  
## 2 0 101 0 0 2 0  
## 3 3 0 213 23 0 6  
## 4 0 0 6 48 0 14  
## 5 4 2 0 0 83 3  
## 7 0 2 1 19 11 204  
##   
##   
## Misclassification Rate = 0.106

results = bagg.sscv(mod2,SATimage$class,data=SATimage,B=5,nbagg=25)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1125 0.1186 0.1226 0.1207 0.1247 0.1253

A slight decrease, maybe the bagging settings themselves we provide more.

control = rpart.control(minsplit=3,minbucket=2,cp=0,xval=0)  
sat.bag2 = bagging(class~.,data=SATtrain,nbagg=150,coob=T,control=control)  
sat.bag2

##   
## Bagging classification trees with 150 bootstrap replications   
##   
## Call: bagging.data.frame(formula = class ~ ., data = SATtrain, nbagg = 150,   
## coob = T, control = control)  
##   
## Out-of-bag estimate of misclassification error: 0.0999

yhat = predict(sat.bag2,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 246 0 1 1 5 0  
## 2 0 102 0 0 1 0  
## 3 3 0 214 21 0 5  
## 4 0 0 4 51 0 13  
## 5 3 1 0 0 86 3  
## 7 0 2 3 19 10 206  
##   
##   
## Misclassification Rate = 0.095

results = bagg.sscv(mod2,SATimage$class,data=SATimage,B=5,nbagg=150)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.09417 0.10501 0.10705 0.11111 0.12263 0.12669

More bagging brought in a slight decrease in both rates whilst avoiding overfitting. But can we go farther?

control = rpart.control(minsplit=3,minbucket=2,cp=0,xval=0)  
sat.bag2 = bagging(class~.,data=SATtrain,nbagg=200,coob=T,control=control)  
sat.bag2

##   
## Bagging classification trees with 200 bootstrap replications   
##   
## Call: bagging.data.frame(formula = class ~ ., data = SATtrain, nbagg = 200,   
## coob = T, control = control)  
##   
## Out-of-bag estimate of misclassification error: 0.1016

yhat = predict(sat.bag2,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 243 0 1 1 5 0  
## 2 0 102 0 0 1 0  
## 3 3 0 214 22 0 4  
## 4 0 0 5 52 0 11  
## 5 6 1 0 0 86 4  
## 7 0 2 2 17 10 208  
##   
##   
## Misclassification Rate = 0.095

results = bagg.sscv(mod2,SATimage$class,data=SATimage,B=5,nbagg=200)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1077 0.1131 0.1131 0.1153 0.1159 0.1267

A minor improvement, although with metric like these we begin to fear overfitting. What was gained probably was not worth what was spent here. Lets try one more slight alteration before moving on to more advanced methods

control = rpart.control(minsplit=4,minbucket=4,cp=0,xval=0)  
sat.bag2 = bagging(class~.,data=SATtrain,nbagg=175,coob=T,control=control)  
sat.bag2

##   
## Bagging classification trees with 175 bootstrap replications   
##   
## Call: bagging.data.frame(formula = class ~ ., data = SATtrain, nbagg = 175,   
## coob = T, control = control)  
##   
## Out-of-bag estimate of misclassification error: 0.1031

yhat = predict(sat.bag2,newdata=SATtest,type="class")  
misclass(yhat,SATtest$class)

## Table of Misclassification  
## (row = predicted, col = actual)  
## y  
## fit 1 2 3 4 5 7  
## 1 244 0 1 1 5 0  
## 2 0 101 0 0 1 0  
## 3 3 0 215 22 0 4  
## 4 0 0 4 51 0 11  
## 5 5 2 0 0 86 4  
## 7 0 2 2 18 10 208  
##   
##   
## Misclassification Rate = 0.095

results = bagg.sscv(mod2,SATimage$class,data=SATimage,B=5,nbagg=175)  
summary(results)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.09688 0.10705 0.11653 0.11423 0.11721 0.13347

While not inherently bad, it seems like we are too the point that no amount of adjusting is gonna gain us anything major in either category with out simply just overfitting the data. Instead, we should opt to instead use a more advanced method of classification, although looking at these models still has its merits.

Final Model Settings:

# control = rpart.control(minsplit=3,minbucket=2,cp=0,xval=0)  
# sat.bag2 = bagging(class~.,data=SATtrain,nbagg=150,coob=T,control=control)

Metrics:

Test Set: 0.098

Training Set (Mean): 0.1198

Though bagging was a clear improvment, random forest will probably out do it.

Random Forest:

Packages:

require(randomForest)

Fitting a basic model:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=1,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 1, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 1

OOB estimate of error rate: 9.99%

Confusion matrix: 1 2 3 4 5 7 class.error 1 801 1 15 0 3 0 0.02317073 2 2 362 1 2 4 3 0.03208556 3 5 0 714 15 0 5 0.03382950 4 4 3 65 183 3 65 0.43343653 5 30 1 1 6 298 32 0.19021739 7 0 0 13 48 16 734 0.09494451

Checking it:

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 0 2 1 7 0 2 0 102 0 0 2 0 3 2 0 212 19 1 6 4 0 0 5 52 0 6 5 3 1 0 0 83 3 7 0 2 3 20 9 212

Misclassification Rate = 0.092

We already can see an improvment over our best bag model, although some internal split sample should still be considered.

Split-Sample function for Random Forest

crf.sscv = function(fit,y,data,B=25,p=.333,mtry=fit$mtry,ntree=fit$ntree) {  
n = length(y)  
cv <- rep(0,B)  
 for (i in 1:B) {  
ss <- floor(n\*p)  
sam <- sample(1:n,ss)  
temp <- data[-sam,]  
fit2 <- randomForest(formula(fit),data=temp,mtry=mtry,ntree=ntree)  
ynew <- predict(fit2,newdata=data[sam,],type="class")  
tab <- table(y[sam],ynew)  
mc <- ss - sum(diag(tab))  
cv[i] <- mc/ss  
}  
 cv  
}

Using it:

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

The improvment holds up here as well, however there are still many features we can try to adjust to pull the most out of this kind of model. Well start with finding the preffered value for mtry, or the number of variables randomly sampled at each split.

mtry = 2:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=2,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 2, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 2

OOB estimate of error rate: 9.29%

Confusion matrix: 1 2 3 4 5 7 class.error 1 804 1 13 0 2 0 0.01951220 2 1 365 0 2 3 3 0.02406417 3 3 2 717 10 0 7 0.02976996 4 4 4 64 192 2 57 0.40557276 5 26 2 1 3 302 34 0.17934783 7 0 1 15 48 11 736 0.09247842

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 0 2 2 5 0 2 0 102 0 0 2 0 3 3 0 212 19 0 6 4 0 0 5 52 0 6 5 2 1 0 0 86 4 7 0 2 3 19 9 211

Misclassification Rate = 0.09

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08311 0.09449 0.09974 0.09928 0.10324 0.11636

mtry = 3:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=3,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 3, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 3

OOB estimate of error rate: 9.11%

Confusion matrix: 1 2 3 4 5 7 class.error 1 796 2 17 0 5 0 0.02926829 2 0 367 0 2 4 1 0.01871658 3 4 1 716 12 0 6 0.03112314 4 3 3 64 192 2 59 0.40557276 5 24 2 1 3 310 28 0.15760870 7 0 1 12 44 13 741 0.08631319

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 0 2 2 5 0 2 0 102 0 0 2 0 3 3 0 212 19 0 5 4 0 0 5 54 0 6 5 2 1 0 0 86 4 7 0 2 3 17 9 212

Misclassification Rate = 0.087

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08399 0.09274 0.09711 0.09788 0.10324 0.11549

mtry = 4:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=4,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 4, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 4

OOB estimate of error rate: 9%

Confusion matrix: 1 2 3 4 5 7 class.error 1 801 2 13 0 4 0 0.02317073 2 0 364 1 2 4 3 0.02673797 3 3 1 716 12 0 7 0.03112314 4 3 3 66 198 2 51 0.38699690 5 24 2 0 6 311 25 0.15489130 7 0 1 13 49 12 736 0.09247842

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 0 2 2 5 0 2 0 102 0 0 2 0 3 3 0 212 19 0 5 4 0 0 5 53 0 6 5 2 1 0 0 87 3 7 0 2 3 18 8 213

Misclassification Rate = 0.086

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08049 0.09099 0.09624 0.09655 0.10149 0.11724

mtry = 5:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=5,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 5, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 5

OOB estimate of error rate: 9.05%

Confusion matrix: 1 2 3 4 5 7 class.error 1 799 1 15 0 5 0 0.02560976 2 1 364 1 2 4 2 0.02673797 3 5 1 713 12 1 7 0.03518268 4 2 3 64 195 2 57 0.39628483 5 20 2 0 5 314 27 0.14673913 7 0 1 12 43 16 739 0.08877928

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 246 0 2 2 5 0 2 0 103 0 0 2 0 3 3 0 212 18 0 5 4 0 0 5 54 0 6 5 3 0 0 0 85 4 7 0 2 3 18 10 212

Misclassification Rate = 0.088

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08224 0.09536 0.10061 0.09963 0.10324 0.11811

mtry = 6:

sat.rf = randomForest(class~.,data=SATtrain,mtry=6,importance=T)  
sat.rf

##   
## Call:  
## randomForest(formula = class ~ ., data = SATtrain, mtry = 6, importance = T)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 6  
##   
## OOB estimate of error rate: 8.7%  
## Confusion matrix:  
## 1 2 3 4 5 7 class.error  
## 1 802 2 13 0 3 0 0.02195122  
## 2 0 367 0 1 4 2 0.01871658  
## 3 4 1 714 14 0 6 0.03382950  
## 4 3 2 65 195 2 56 0.39628483  
## 5 18 2 1 3 314 30 0.14673913  
## 7 0 1 15 38 13 744 0.08261406

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 6, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 6

OOB estimate of error rate: 8.85%

Confusion matrix: 1 2 3 4 5 7 class.error 1 799 2 15 0 4 0 0.02560976 2 0 366 0 2 4 2 0.02139037 3 3 1 714 14 0 7 0.03382950 4 4 3 63 198 2 53 0.38699690 5 21 2 0 4 316 25 0.14130435 7 0 1 12 42 18 738 0.09001233

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 0 1 2 5 0 2 0 102 0 0 2 0 3 3 0 215 19 0 5 4 0 0 5 52 0 6 5 2 1 0 0 86 4 7 0 2 1 19 9 212

Misclassification Rate = 0.086

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08049 0.09274 0.09536 0.09596 0.10149 0.11374

We can see that out of the gate we are seeing improvement as we consider more variables, knowing this, we will try and zero in on that target.

mtry = 7:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=7,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 7, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 7

OOB estimate of error rate: 8.68%

Confusion matrix: 1 2 3 4 5 7 class.error 1 800 2 14 0 4 0 0.02439024 2 1 366 0 2 3 2 0.02139037 3 2 1 714 15 0 7 0.03382950 4 4 2 64 193 2 58 0.40247678 5 18 3 0 4 318 25 0.13586957 7 0 1 14 37 13 746 0.08014797

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 246 0 1 2 5 0 2 0 104 0 0 2 0 3 3 0 214 19 0 5 4 0 0 5 54 0 6 5 3 0 0 0 85 3 7 0 1 2 17 10 213

Misclassification Rate = 0.084

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08661 0.09274 0.09974 0.09876 0.10411 0.11199

mtry = 10:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=10,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 10, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 10

OOB estimate of error rate: 8.68%

Confusion matrix: 1 2 3 4 5 7 class.error 1 799 2 14 0 5 0 0.02560976 2 1 363 1 1 5 3 0.02941176 3 2 1 718 11 0 7 0.02841678 4 4 2 64 202 2 49 0.37461300 5 16 2 1 3 319 27 0.13315217 7 0 1 12 45 17 736 0.09247842

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 246 0 1 2 5 0 2 0 103 0 0 2 0 3 3 0 214 19 0 5 4 0 0 5 53 0 7 5 3 0 0 0 85 4 7 0 2 2 18 10 211

Misclassification Rate = 0.088

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.07612 0.09099 0.09711 0.09606 0.10236 0.11111

mtry = 15:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=15,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 15, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 15

OOB estimate of error rate: 8.94%

Confusion matrix: 1 2 3 4 5 7 class.error 1 797 1 13 0 9 0 0.02804878 2 1 362 0 2 7 2 0.03208556 3 4 1 716 12 1 5 0.03112314 4 5 2 68 193 1 54 0.40247678 5 15 2 2 2 324 23 0.11956522 7 0 1 13 46 15 736 0.09247842

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 246 0 1 2 5 0 2 0 103 0 0 2 0 3 3 0 215 19 0 4 4 0 0 4 51 0 8 5 3 0 0 0 85 4 7 0 2 2 20 10 211

Misclassification Rate = 0.089

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08136 0.09099 0.09624 0.09683 0.10149 0.11899

mtry = 20:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=20,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 20, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 20

OOB estimate of error rate: 9%

Confusion matrix: 1 2 3 4 5 7 class.error 1 793 1 14 0 12 0 0.03292683 2 1 364 1 1 5 2 0.02673797 3 4 1 714 13 0 7 0.03382950 4 5 1 65 193 1 58 0.40247678 5 15 3 2 2 323 23 0.12228261 7 0 1 15 42 14 739 0.08877928

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 245 0 1 2 5 0 2 0 103 0 0 1 0 3 3 0 215 19 0 4 4 0 0 4 51 0 10 5 4 0 0 0 86 3 7 0 2 2 20 10 210

Misclassification Rate = 0.09

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08136 0.09274 0.09886 0.09753 0.10411 0.10849

mtry = 25:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=25,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 25, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 25

OOB estimate of error rate: 9.08%

Confusion matrix: 1 2 3 4 5 7 class.error 1 797 1 13 0 9 0 0.02804878 2 1 363 1 2 5 2 0.02941176 3 5 1 712 14 0 7 0.03653586 4 8 1 66 189 3 56 0.41486068 5 16 4 1 3 322 22 0.12500000 7 0 1 16 38 16 740 0.08754624

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 243 0 1 2 5 0 2 0 102 0 0 1 0 3 3 0 215 19 0 4 4 0 0 5 50 0 9 5 6 1 0 0 86 4 7 0 2 1 21 10 210

Misclassification Rate = 0.094

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.07612 0.09186 0.09711 0.09890 0.10761 0.12161

mtry = 30:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=30,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 30, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 30

OOB estimate of error rate: 9.2%

Confusion matrix: 1 2 3 4 5 7 class.error 1 800 1 10 0 9 0 0.02439024 2 1 362 1 0 8 2 0.03208556 3 7 1 712 13 0 6 0.03653586 4 8 2 67 187 1 58 0.42105263 5 19 4 0 5 317 23 0.13858696 7 0 1 17 37 15 741 0.08631319

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 242 0 1 1 5 0 2 0 102 0 0 1 0 3 3 0 214 20 0 4 4 0 0 4 50 0 11 5 7 1 0 0 86 4 7 0 2 3 21 10 208

Misclassification Rate = 0.098

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08311 0.09711 0.10324 0.10268 0.11024 0.12248

mtry = 36:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=36,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 36, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 36

OOB estimate of error rate: 9.46%

Confusion matrix: 1 2 3 4 5 7 class.error 1 799 1 10 0 10 0 0.02560976 2 2 360 1 1 7 3 0.03743316 3 6 0 710 13 0 10 0.03924222 4 6 1 66 183 3 64 0.43343653 5 17 4 1 3 321 22 0.12771739 7 0 0 17 42 15 737 0.09124538

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 242 0 1 1 5 0 2 0 102 0 0 1 0 3 3 0 214 21 0 5 4 0 1 4 51 0 10 5 7 1 0 0 86 4 7 0 1 3 19 10 208

Misclassification Rate = 0.097

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.07962 0.09449 0.10236 0.10096 0.10849 0.11811

Our answer probably is between 7 and 10

mtry = 8:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=8,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 8, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 8

OOB estimate of error rate: 8.82%

Confusion matrix: 1 2 3 4 5 7 class.error 1 797 2 16 0 5 0 0.02804878 2 1 364 1 1 4 3 0.02673797 3 2 1 717 12 0 7 0.02976996 4 3 2 67 196 2 53 0.39318885 5 19 2 0 3 320 24 0.13043478 7 0 1 14 43 15 738 0.09001233

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 246 0 1 2 5 0 2 0 102 0 0 2 0 3 3 0 215 19 0 5 4 0 0 5 53 0 6 5 3 1 0 0 85 4 7 0 2 1 18 10 212

Misclassification Rate = 0.087

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08311 0.09011 0.09449 0.09718 0.10411 0.11461

mtry = 9:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=9,importance=T)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 9, importance = T) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 9

OOB estimate of error rate: 8.94%

Confusion matrix: 1 2 3 4 5 7 class.error 1 797 2 16 0 5 0 0.02804878 2 1 363 1 2 5 2 0.02941176 3 3 1 716 13 0 6 0.03112314 4 5 3 66 191 2 56 0.40866873 5 12 2 1 3 323 27 0.12228261 7 0 1 12 45 15 738 0.09001233

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 0 1 2 5 0 2 0 103 0 0 2 0 3 3 0 214 19 0 4 4 0 0 5 52 0 6 5 2 0 0 0 85 4 7 0 2 2 19 10 213

Misclassification Rate = 0.086

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08224 0.09274 0.10061 0.09935 0.10586 0.11286

Funnily enough, it turns out that 7 was actually the optimal value for mtry.

While the best model yet by far, we still can try to play around with a larger nodesize to bring that missclassification rate down further.

# sat.rf = randomForest(class~.,data=SATtrain,mtry=7,importance=T, nodesize = 2)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 7, importance = T, nodesize = 2) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 7

OOB estimate of error rate: 9%

Confusion matrix: 1 2 3 4 5 7 class.error 1 800 2 14 0 4 0 0.02439024 2 1 364 1 2 4 2 0.02673797 3 2 1 718 11 0 7 0.02841678 4 5 3 65 193 2 55 0.40247678 5 22 2 0 3 317 24 0.13858696 7 0 0 14 47 16 734 0.09494451

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 246 0 2 2 5 0 2 0 103 0 0 2 0 3 3 0 214 18 0 5 4 0 0 5 54 0 8 5 3 0 0 0 85 3 7 0 2 1 18 10 211

Misclassification Rate = 0.087

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08224 0.09274 0.09624 0.09760 0.10236 0.11811

We got ever so slightly worse on the test set, more nodesize likely isn’t going to help us. Next we will try to see if more trees will provide us a gain or only overfit.

# sat.rf = randomForest(class~.,data=SATtrain,mtry=7,importance=T, nodesize = 1, ntree = 750)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 7, importance = T, nodesize = 1, ntree = 750) Type of random forest: classification Number of trees: 750 No. of variables tried at each split: 7

OOB estimate of error rate: 8.88%

Confusion matrix: 1 2 3 4 5 7 class.error 1 801 2 12 0 5 0 0.02317073 2 0 365 0 1 7 1 0.02406417 3 3 1 716 11 0 8 0.03112314 4 4 2 63 194 2 58 0.39938080 5 16 2 1 5 319 25 0.13315217 7 0 1 17 43 15 735 0.09371147

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 0 1 2 5 0 2 0 103 0 0 2 0 3 3 0 214 19 0 5 4 0 0 5 52 0 6 5 2 0 0 0 85 3 7 0 2 2 19 10 213

Misclassification Rate = 0.086

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08399 0.09099 0.09711 0.09662 0.10061 0.11286

More trees actually made a slightly worse model, likely due to more overfitting. Maybe if we cut it down a bit we can see something. Although this may not help due to randomforests needing a lot of trees.

# sat.rf = randomForest(class~.,data=SATtrain,mtry=7,importance=T, nodesize = 1, ntree = 400)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 7, importance = T, nodesize = 1, ntree = 400) Type of random forest: classification Number of trees: 400 No. of variables tried at each split: 7

OOB estimate of error rate: 8.79%

Confusion matrix: 1 2 3 4 5 7 class.error 1 798 2 14 0 6 0 0.02682927 2 0 363 0 2 5 4 0.02941176 3 3 1 717 11 0 7 0.02976996 4 3 3 65 195 3 54 0.39628483 5 15 2 1 2 321 27 0.12771739 7 0 1 13 46 12 739 0.08877928

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 0 1 2 5 0 2 0 103 0 0 2 0 3 3 0 215 19 0 5 4 0 0 5 53 0 8 5 2 0 0 0 85 3 7 0 2 1 18 10 211

Misclassification Rate = 0.086

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08399 0.09186 0.09886 0.09708 0.10149 0.10849

Reducing the nodesize again only caused it to diminish slightly. What if we set a limit to maximal node size?

Maxnodes = 10

# sat.rf = randomForest(class~.,data=SATtrain,mtry=7,importance=T, nodesize = 1, ntree = 500, maxnodes = 10)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 7, importance = T, nodesize = 1, ntree = 500, maxnodes = 10) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 7

OOB estimate of error rate: 20.2%

Confusion matrix: 1 2 3 4 5 7 class.error 1 803 0 13 0 1 3 0.02073171 2 40 330 0 0 1 3 0.11764706 3 44 0 690 2 0 3 0.06630582 4 98 0 80 46 0 99 0.85758514 5 130 7 0 0 182 49 0.50543478 7 81 0 19 18 3 690 0.14919852

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 12 13 31 31 23 2 0 91 0 0 3 0 3 3 0 209 24 0 3 4 0 0 0 12 0 4 5 2 0 0 0 49 1 7 0 2 0 25 19 196

Misclassification Rate = 0.196

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.07787 0.08924 0.09361 0.09480 0.09886 0.11286

Maxnodes = 50

# sat.rf = randomForest(class~.,data=SATtrain,mtry=7,importance=T, nodesize = 1, ntree = 500, maxnodes = 50)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 7, importance = T, nodesize = 1, ntree = 500, maxnodes = 50) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 7

OOB estimate of error rate: 12.31%

Confusion matrix: 1 2 3 4 5 7 class.error 1 802 1 15 0 2 0 0.02195122 2 5 353 1 5 5 5 0.05614973 3 6 0 713 13 1 6 0.03518268 4 15 1 68 171 0 68 0.47058824 5 59 1 0 3 260 45 0.29347826 7 4 0 18 71 5 713 0.12083847

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 0 2 3 12 1 2 0 102 0 0 2 0 3 3 0 213 20 0 6 4 0 0 5 47 0 16 5 2 1 0 0 75 2 7 0 2 2 22 13 202

Misclassification Rate = 0.114

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08136 0.08749 0.09536 0.09613 0.10411 0.11374

Maxnodes = 100

# sat.rf = randomForest(class~.,data=SATtrain,mtry=7,importance=T, nodesize = 1, ntree = 500, maxnodes = 100)  
# sat.rf

Call: randomForest(formula = class ~ ., data = SATtrain, mtry = 7, importance = T, nodesize = 1, ntree = 500, maxnodes = 100) Type of random forest: classification Number of trees: 500 No. of variables tried at each split: 7

OOB estimate of error rate: 11.21%

Confusion matrix: 1 2 3 4 5 7 class.error 1 800 1 15 0 4 0 0.02439024 2 1 359 2 3 5 4 0.04010695 3 4 1 713 12 1 8 0.03518268 4 11 2 68 172 1 69 0.46749226 5 45 2 1 3 281 36 0.23641304 7 1 0 17 60 8 725 0.10604192

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 0 2 2 9 0 2 0 102 0 0 2 0 3 3 0 213 19 0 5 4 0 0 5 49 0 12 5 2 1 0 0 79 3 7 0 2 2 22 12 207

Misclassification Rate = 0.103

# results = crf.sscv(sat.rf,SATtrain$class,data=SATtrain)  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.07612 0.09099 0.09711 0.09599 0.10149 0.11199

Trying to limit the algorithm manually begins to become counterintuitive, and beyond and endless brute force method, it is likely we will see only marginal improvement of dubious quality.

In the end, a realtively defualt model with only a specified mty of 7 variables was used to great effect over our previous models.

Final Model:

# sat.rf = randomForest(class~.,data=SATtrain,mtry=7,importance=T, nodesize = 1, ntree = 500, maxnodes = 100)

Metrics:

Test Set: 0.084

Training Set (Mean): 0.098

Finally, we will build a boosted tree for classification.

Boosted Tree:

Packages:

require(adabag)

Basic Model:

# sat.boost = boosting(class~.,data=SATtrain,mfinal=200)  
# summary(sat.boost)

Length Class Mode

formula 3 formula call  
trees 200 -none- list  
weights 200 -none- numeric  
votes 20610 -none- numeric  
prob 20610 -none- numeric  
class 3435 -none- character importance 36 -none- numeric  
terms 3 terms call  
call 4 -none- call

# misclass(sat.boost$class,SATtrain$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 812 0 3 2 8 0 2 1 373 0 0 0 0 3 7 0 728 53 0 6 4 0 0 4 238 1 17 5 0 1 0 0 343 6 7 0 0 4 30 16 782

Misclassification Rate = 0.0463

# yhat = predict(sat.boost,newdata=SATtest)  
# summary(yhat)

Length Class Mode

formula 3 formula call  
votes 6000 -none- numeric  
prob 6000 -none- numeric  
class 1000 -none- character confusion 36 table numeric  
error 1 -none- numeric

#yhat$error

[1] 0.108

Split sample function for boosting

boost.sscv = function(fit,y,data,p=.333,B=25,control=rpart.control())   
{  
n = length(y)  
cv <- rep(0,B)  
for (i in 1:B) {  
ss <- floor(n\*p)  
sam <- sample(1:n,ss,replace=F)  
temp <- data[-sam,]  
fit2 <- boosting(formula(fit),data=temp,control=control)  
ypred <- predict(fit2,newdata=data[sam,])  
tab = ypred$confusion  
mc <- ss - sum(diag(tab))  
cv[i] <- mc/ss  
 }  
cv  
}  
  
# results = boost.sscv(sat.boost,SATtrain$class,data=SATtrain,p=0.25,B=25)  
  
# summary(results)

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.08042 0.09091 0.09907 0.09925 0.10839 0.11888

Due to impracticality associated with time to run split sample, in the future we will be electing to not use it for this example (using it here to more than 30 minutes).

While on the training set we blew everything else out of the water with a missclassification rate of 4.63%, when we got onto the test set if performed supbar, indicating that we overfit the data. We can now try to roll that back

Starting out we will try and find the best weighting formula. Keep in mind that Breiman is the defualt

Trying Freund’s

# sat.boost = boosting(class~.,data=SATtrain,mfinal=200,coeflearn = "Freund" )  
# summary(sat.boost)

Length Class Mode

formula 3 formula call  
trees 200 -none- list  
weights 200 -none- numeric  
votes 20610 -none- numeric  
prob 20610 -none- numeric  
class 3435 -none- character importance 36 -none- numeric  
terms 3 terms call  
call 5 -none- call

# misclass(sat.boost$class,SATtrain$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 819 0 0 0 2 0 2 0 374 0 0 0 0 3 1 0 739 29 0 2 4 0 0 0 282 0 5 5 0 0 0 0 363 1 7 0 0 0 12 3 803

Misclassification Rate = 0.016

# yhat = predict(sat.boost,newdata=SATtest)  
# yhat$error

[1] 0.095

Freunds actually did seem to help.

Trying Zhu’s

# sat.boost = boosting(class~.,data=SATtrain,mfinal=200,coeflearn = "Zhu" )  
# summary(sat.boost)

Length Class Mode

formula 3 formula call  
trees 200 -none- list  
weights 200 -none- numeric  
votes 20610 -none- numeric  
prob 20610 -none- numeric  
class 3435 -none- character importance 36 -none- numeric  
terms 3 terms call  
call 5 -none- call

# yhat = predict(sat.rf,newdata=SATtest)  
# misclass(yhat,SATtest$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 247 0 2 2 9 0 2 0 102 0 0 2 0 3 3 0 213 19 0 5 4 0 0 5 49 0 12 5 2 1 0 0 79 3 7 0 2 2 22 12 207

Misclassification Rate = 0.103

# yhat = predict(sat.boost,newdata=SATtest)  
# yhat$error

[1] 0.118

Zhu’s did not work out as well.

Next, we will try to reduce mfinal to overcome that overfitting problem.

# sat.boost = boosting(class~.,data=SATtrain,mfinal=100,coeflearn = "Freund" )  
# summary(sat.boost)

Length Class Mode

formula 3 formula call  
trees 100 -none- list  
weights 100 -none- numeric  
votes 20610 -none- numeric  
prob 20610 -none- numeric  
class 3435 -none- character importance 36 -none- numeric  
terms 3 terms call  
call 5 -none- call

# misclass(sat.boost$class,SATtrain$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 819 0 0 0 2 0 2 0 374 0 0 0 0 3 1 0 739 34 0 5 4 0 0 0 275 0 8 5 0 0 0 0 361 3 7 0 0 0 14 5 795

Misclassification Rate = 0.021

Misclassification Rate = 0.103

# yhat = predict(sat.boost,newdata=SATtest)  
# yhat$error

[1] 0.092

While overfitting was reduced, it still did not bring it in line enough to match random forest. We will have to try and use additional rpart controls to solve that. However, while we are here, we might as well try splitting the difference.

# sat.boost = boosting(class~.,data=SATtrain,mfinal=150,coeflearn = "Freund" )  
# summary(sat.boost)

Length Class Mode

formula 3 formula call  
trees 150 -none- list  
weights 150 -none- numeric  
votes 20610 -none- numeric  
prob 20610 -none- numeric  
class 3435 -none- character importance 36 -none- numeric  
terms 3 terms call  
call 5 -none- call

# misclass(sat.boost$class,SATtrain$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 819 0 0 0 2 0 2 0 374 0 0 0 0 3 1 0 739 34 0 5 4 0 0 0 275 0 8 5 0 0 0 0 361 3 7 0 0 0 14 5 795

Misclassification Rate = 0.021

# yhat = predict(sat.boost,newdata=SATtest)  
# yhat$error

[1] 0.097

100 iterations still seems to be the best.

We will now start playing with bin and bucket settings. Since they worked well for previous trees, we might as well start with what was optimal for the more basic designs and work our from there (3 split, 2 bucket).

# sat.boost = boosting(class~.,data=SATtrain,mfinal=150,coeflearn = "Freund", control=rpart.control(minsplit=3,minbucket=2))  
  
# summary(sat.boost)

Length Class Mode

formula 3 formula call  
trees 150 -none- list  
weights 150 -none- numeric  
votes 20610 -none- numeric  
prob 20610 -none- numeric  
class 3435 -none- character importance 36 -none- numeric  
terms 3 terms call  
call 6 -none- call

# misclass(sat.boost$class,SATtrain$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 819 0 0 0 3 0 2 0 374 0 0 0 0 3 1 0 739 28 0 3 4 0 0 0 280 0 6 5 0 0 0 0 363 1 7 0 0 0 15 2 801

Misclassification Rate = 0.0172

# yhat = predict(sat.boost,newdata=SATtest)  
# yhat$error

[1] 0.099

Our training rate, got better and our test rate got worse, so we definetly are going to want to modify that a bit. Perhaps the solution is utilizing the complexity parameter.

# sat.boost = boosting(class~.,data=SATtrain,mfinal=150,coeflearn = "Freund", control=rpart.control(minsplit=3,minbucket=2, cp=0.01))  
#   
# summary(sat.boost)

Length Class Mode

formula 3 formula call  
trees 150 -none- list  
weights 150 -none- numeric  
votes 20610 -none- numeric  
prob 20610 -none- numeric  
class 3435 -none- character importance 36 -none- numeric  
terms 3 terms call  
call 6 -none- call

# misclass(sat.boost$class,SATtrain$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 819 0 0 0 2 0 2 0 374 0 0 0 0 3 1 0 738 31 0 1 4 0 0 1 278 0 6 5 0 0 0 0 364 2 7 0 0 0 14 2 802

Misclassification Rate = 0.0175

# yhat = predict(sat.boost,newdata=SATtest)  
# yhat$error

[1] 0.105

While an ever so slight improvment on overfitting, it appears that more tinkering with the bin and split is where are solution must lie.

# sat.boost = boosting(class~.,data=SATtrain,mfinal=100,coeflearn = "Freund", control=rpart.control(minsplit=5,minbucket=4))  
#   
# summary(sat.boost)

Length Class Mode

formula 3 formula call  
trees 100 -none- list  
weights 100 -none- numeric  
votes 20610 -none- numeric  
prob 20610 -none- numeric  
class 3435 -none- character importance 36 -none- numeric  
terms 3 terms call  
call 6 -none- call

# misclass(sat.boost$class,SATtrain$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 814 0 1 0 8 0 2 0 374 0 0 0 0 3 6 0 735 38 0 8 4 0 0 3 262 0 12 5 0 0 0 0 355 3 7 0 0 0 23 5 788

Misclassification Rate = 0.0311

# yhat = predict(sat.boost,newdata=SATtest)  
# yhat$error

[1] 0.105

It seems like minor alterations are not producing results. Re-evaluating the problem with a new mindset, we make some extreme changes

# sat.boost = boosting(class~.,data=SATtrain,mfinal=100,coeflearn = "Freund", control=rpart.control(minsplit=12,minbucket=6))  
#   
# summary(sat.boost)

Length Class Mode

formula 3 formula call  
trees 100 -none- list  
weights 100 -none- numeric  
votes 20610 -none- numeric  
prob 20610 -none- numeric  
class 3435 -none- character importance 36 -none- numeric  
terms 3 terms call  
call 6 -none- call

# misclass(sat.boost$class,SATtrain$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 815 0 0 0 4 0 2 0 374 0 0 0 0 3 4 0 736 39 0 6 4 0 0 3 266 0 12 5 1 0 0 0 360 4 7 0 0 0 18 4 789

Misclassification Rate = 0.0277

# yhat = predict(sat.boost,newdata=SATtest)  
# yhat$error

[1] 0.096

A step in the right direction, we finally have something that truly beats basic bagging. Let’s try go a little farther with modifying anything until an improvement is made.

# sat.boost = boosting(class~.,data=SATtrain,mfinal=100,coeflearn = "Freund", control=rpart.control(minsplit=12,minbucket=6, maxdepth = 8, xval = 20))  
   
# summary(sat.boost)

Length Class Mode

formula 3 formula call  
trees 100 -none- list  
weights 100 -none- numeric  
votes 20610 -none- numeric  
prob 20610 -none- numeric  
class 3435 -none- character importance 36 -none- numeric  
terms 3 terms call  
call 6 -none- call

# misclass(sat.boost$class,SATtrain$class)

Table of Misclassification (row = predicted, col = actual) y fit 1 2 3 4 5 7 1 818 0 1 0 1 0 2 0 374 0 0 0 0 3 2 0 735 28 0 4 4 0 0 3 275 0 9 5 0 0 0 0 364 1 7 0 0 0 20 3 797

Misclassification Rate = 0.021

# yhat = predict(sat.boost,newdata=SATtest)  
# yhat$error

[1] 0.1

Unfortunatley, after several iterations not shown here for lack of anything being gained from them, we still good not make any improvement beyond the model with 0.096 on the test set. Tring everything from reducing maxdepth, to abnormal iteration counts, to internal corss-validation, complexiy parameter adjustments, even reavaluting weighting formulae. nothing changed every pushed the model in a direction we would like with any amount not the result of random chance. Whilst better than bagging, we have to give to the random forest that it was supierior in tackling this problem.

Final model:

# sat.boost = boosting(class~.,data=SATtrain,mfinal=100,coeflearn = "Freund", control=rpart.control(minsplit=12,minbucket=6))  
#   
# summary(sat.boost)

Metrics:

Test Set: 0.0277

Training Set (Mean): 0.096

While certainly the best on the training set, boosting fundamentally overfits to much for this case with multiple predictors. Though versions could be made that fit the training data less, the performed worse on the test set than bagging by a notable margin. Still, for some problems, particularly binary ones, it likely will perform much better as simpler trees can be used for each individual iteration.

|  |  |  |
| --- | --- | --- |
| **Modeling Method** | **Training Data Error** | **Test Data Error** |
| Classification Tree | 0.1413 | 0.134 |
| Bagged Classification Trees | 0.1198 | 0.098 |
| Random Forest | 0.098 | 0.084 |
| Boosted Classification Trees | 0.0277 | 0.096 |