

RF for TBI

Ian Shen

12/7/2021

```
library(ranger)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##   filter, lag

## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union
```

```
library(sampling)
```

```
## Warning: package 'sampling' was built under R version 4.1.2
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##   combine
```

```
tbidata <- read.csv("tbi_flattened.csv")
# ciTBI and outcome are the same, we only need one.
tbidata <- subset(tbidata, select=-c(ciTBI))
tbidata$outcome <- as.factor(tbidata$outcome)

set.seed(2112)
# proportion of overall data that has TBI
p <- mean(tbidata$outcome==1)
# Create our validation set via stratified samples.
```

```

stratsample <- function(data,n){
  stratas <- strata(data, c("outcome"),size = c(round(n*(1-p)),round(n*p)), method = "srswor")
  return(getdata(data,stratas)[,1:94])
}

valid <- stratsample(tbidata,4739)
cvdata <- setdiff(tbidata,valid)

# Randomly generate our folds from the remainder of the data:
fold1 <- stratsample(cvdata,10000)
fold23 <- setdiff(cvdata,fold1)
fold2 <- stratsample(fold23,10000)
fold3<- setdiff(fold23,fold2)
fold12 <- rbind(fold1,fold2)
fold13 <- rbind(fold1,fold3)

```

Now that our folds are set up, we can tune our model's hyperparameters. From exploratory work with the forests, we stick with a forest size of 200 trees, as it maintains sufficient accuracy without large computational burdens. Thus, maximum depth would seem to be left as our main hyperparameter of interest.

However, due to the unbalanced nature of our outputs (around 0.7% of children have a TBI, the remaining 99.3% do not), simply fitting a random forest classifier would predict non-TBI every single time, and be more than 99% accurate. Thus we need to add a second hyperparameter: that of weight attached to positive results. This will be our main way of tweaking the balance between sensitivity and specificity; the greater the weight of outcome 1, the more false negatives are penalized and the greater the sensitivity of the model. Given TBI's rarity, we use weights ranging from 100 to 1000.

```

pred_rf_sens <- function(train,test,trees=200,depth=5,weight=1){
  # Gives the sensitivity of a random forest prediction on the TBI data. Increasing the weight of posit
  fit <- ranger(outcome ~ .-Race-Gender-AgeinYears, data = train, num.trees=trees, max.depth = depth,
  pred <- ranger:::predict.ranger(fit,subset(test,select=c(-Race,-Gender,-AgeinYears,-outcome)))
  return(sum(pred$predictions==test$outcome & test$outcome==1)/sum(test$outcome==1))
}

pred_rf_spec <- function(train,test,trees=200,depth=5,weight=1){
  # As before, but with specificity this time.
  fit <- ranger(outcome ~ .-Race-Gender-AgeinYears, data = train, num.trees=trees, max.depth = depth,
  pred <- ranger:::predict.ranger(fit,subset(test,select=c(-Race,-Gender,-AgeinYears,-outcome)))
  return(sum(pred$predictions==test$outcome & test$outcome==0)/sum(test$outcome==0))
}

# Function to generate and plot ROC curves for a given max depth.
rocplot <- function(depth){
  sens <- c()
  spec <- c()
  for(i in 0:10){
    # Take sensitivity and specificity for each fold, then average.
    sens1 <- pred_rf_sens(train=fold23,test=fold1,depth=depth,weight=i*100)
    sens2 <- pred_rf_sens(train=fold13,test=fold2,depth=depth,weight=i*100)
    sens3 <- pred_rf_sens(train=fold12,test=fold3,depth=depth,weight=i*100)
    sens <- c(sens,mean(c(sens1,sens2,sens3)))
    spec1 <- pred_rf_spec(train=fold23,test=fold1,depth=depth,weight=i*100)
    spec2 <- pred_rf_spec(train=fold13,test=fold2,depth=depth,weight=i*100)
    spec3 <- pred_rf_spec(train=fold12,test=fold3,depth=depth,weight=i*100)
  }
}

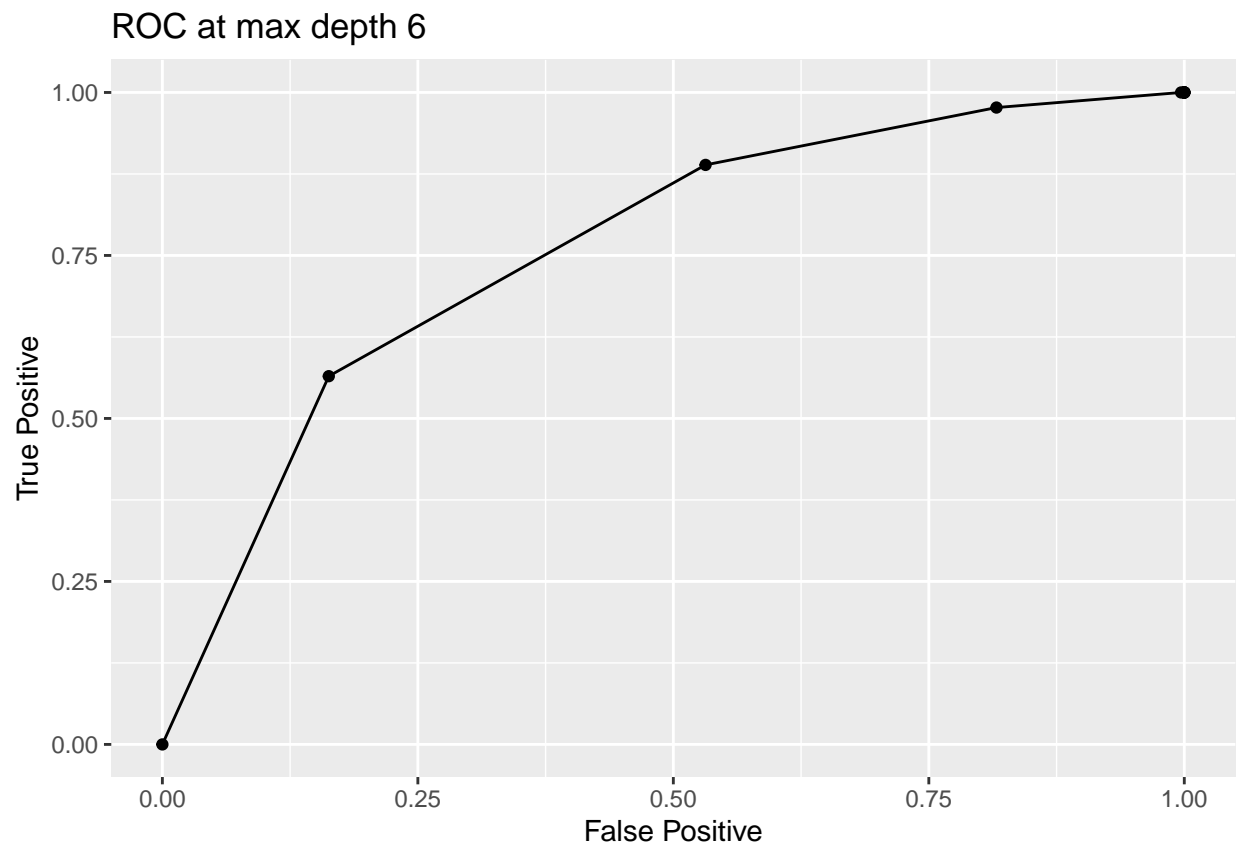
```

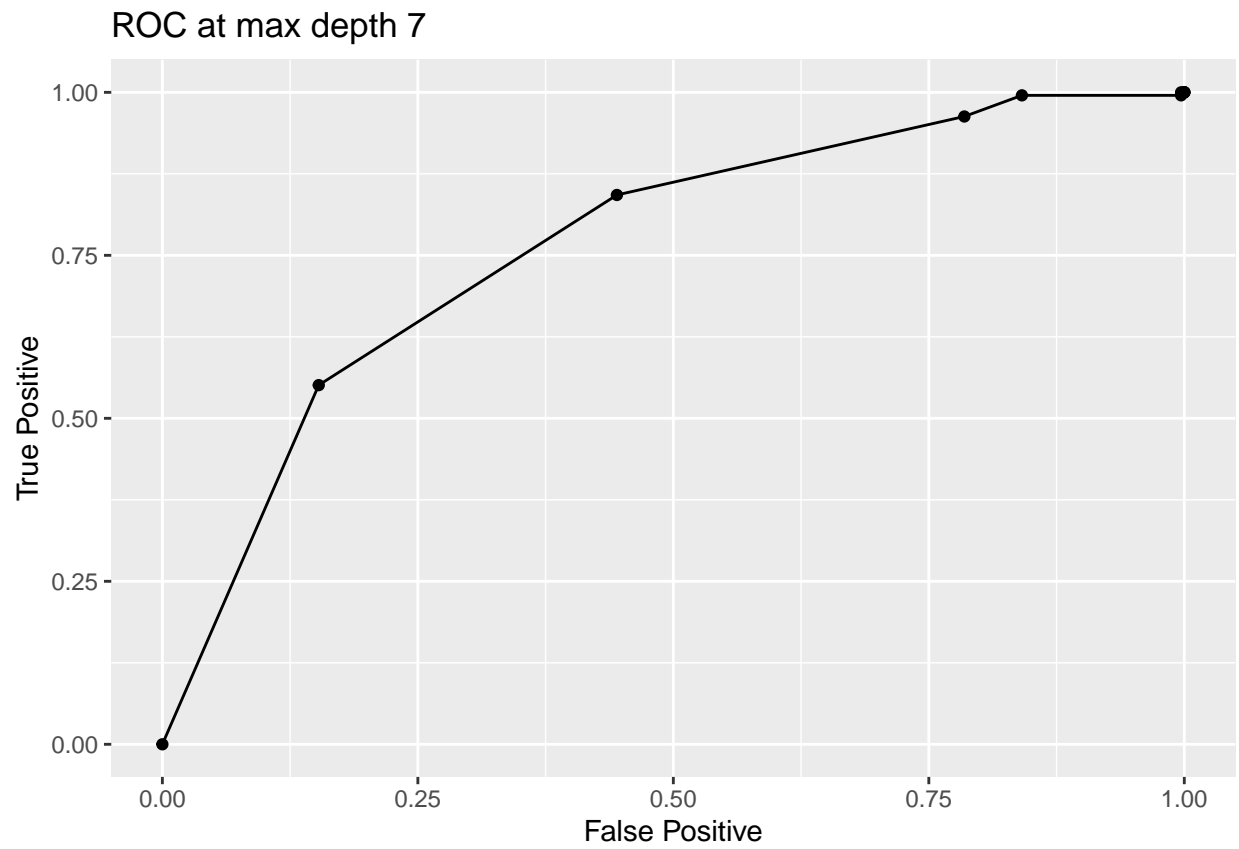
```

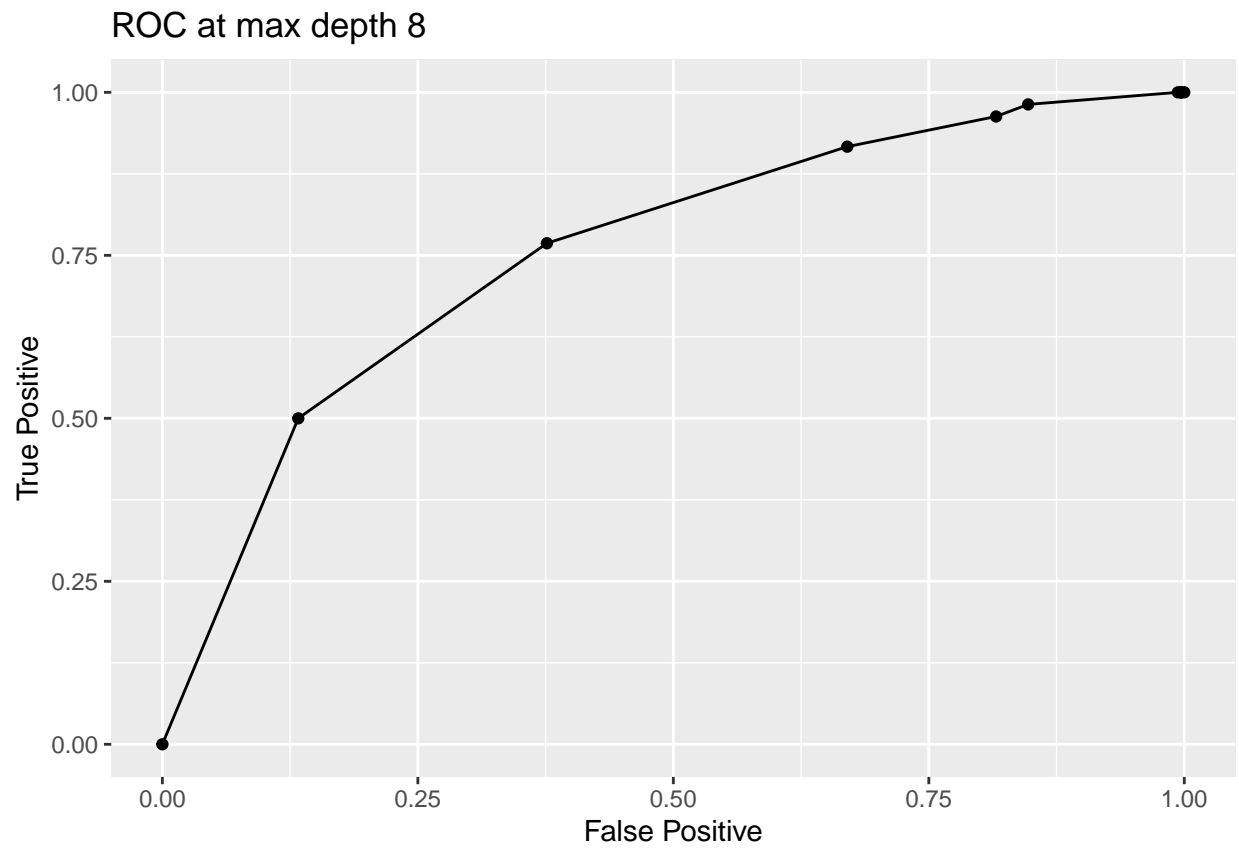
    spec <- c(spec, mean(c(spec1, spec2, spec3)))
  }
  # Create ROC plot
  print(ggplot(mapping=aes(x=1-spec, y=sens)) + geom_line() + geom_point() + xlab("False Positive") + ylab("True Positive"))
}

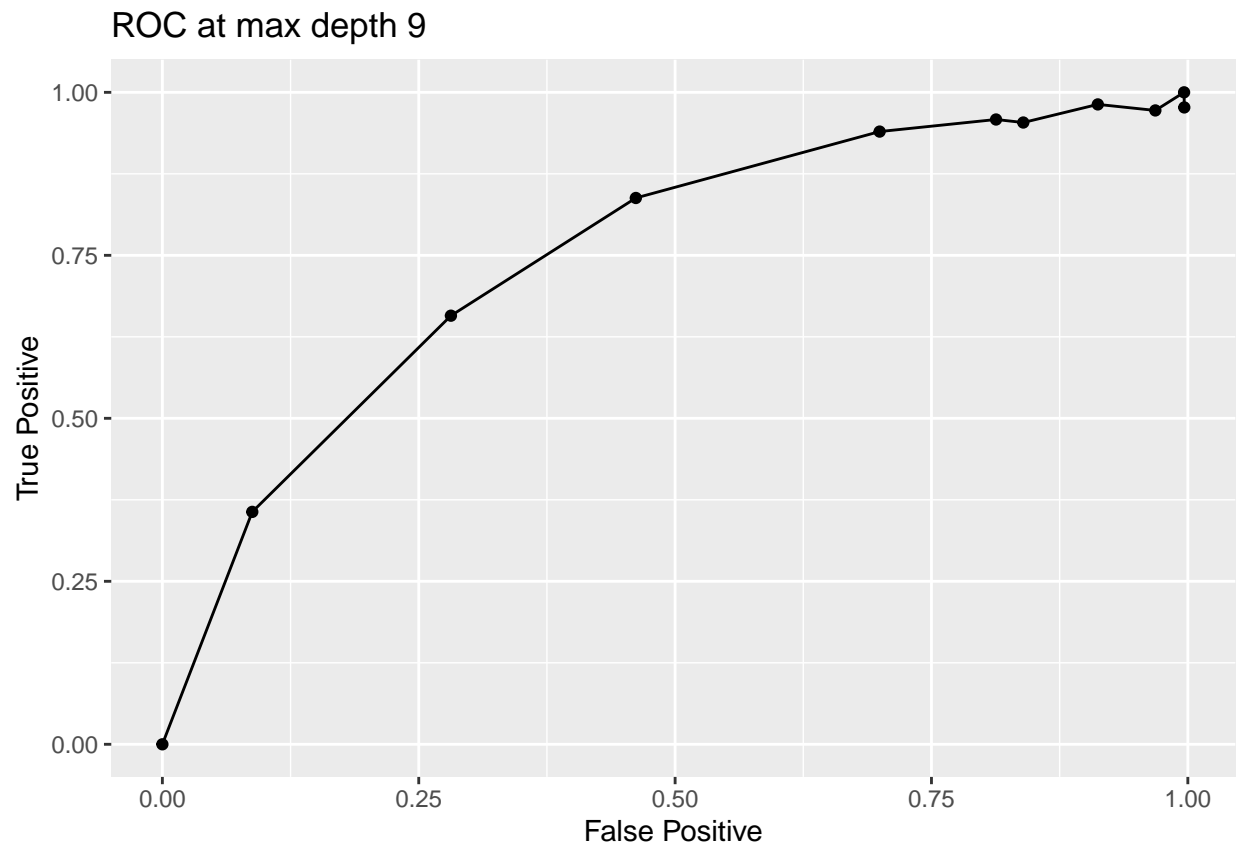
for(i in 6:11){
  rocplot(i)
}

```

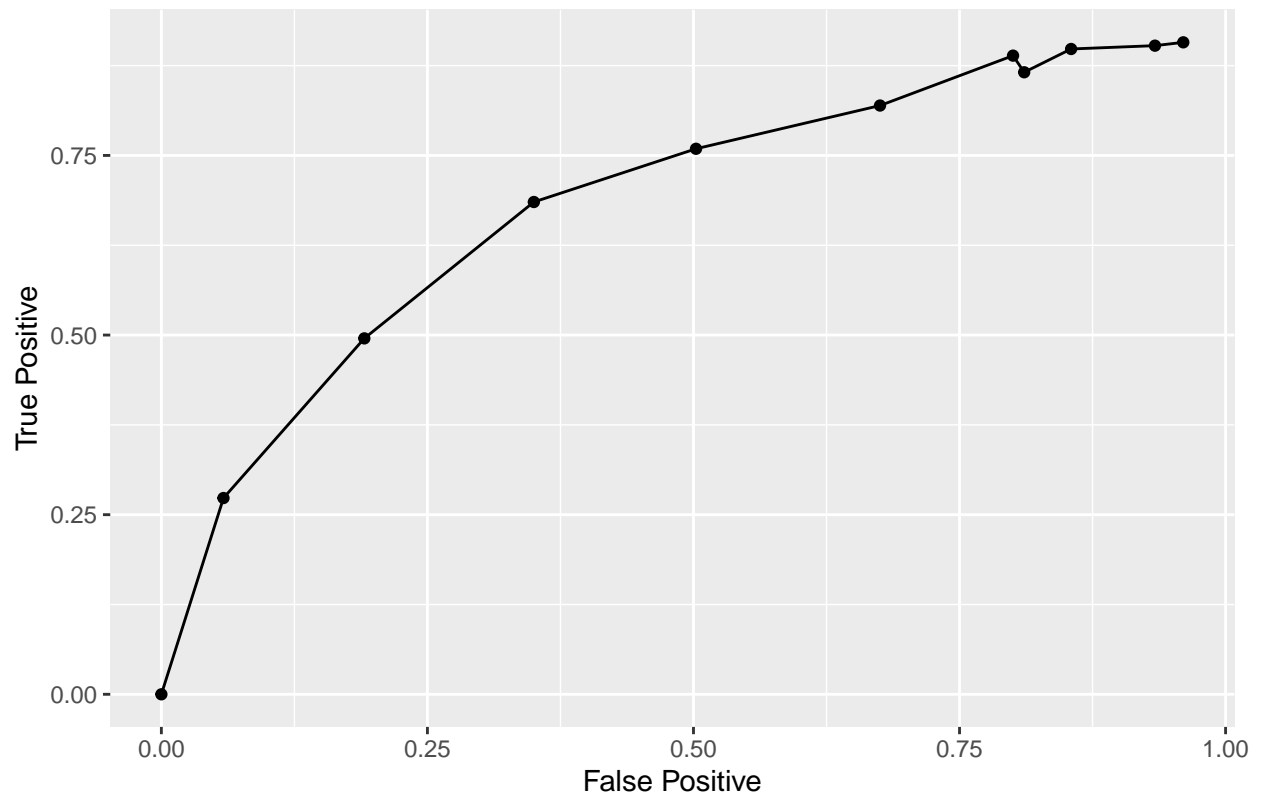




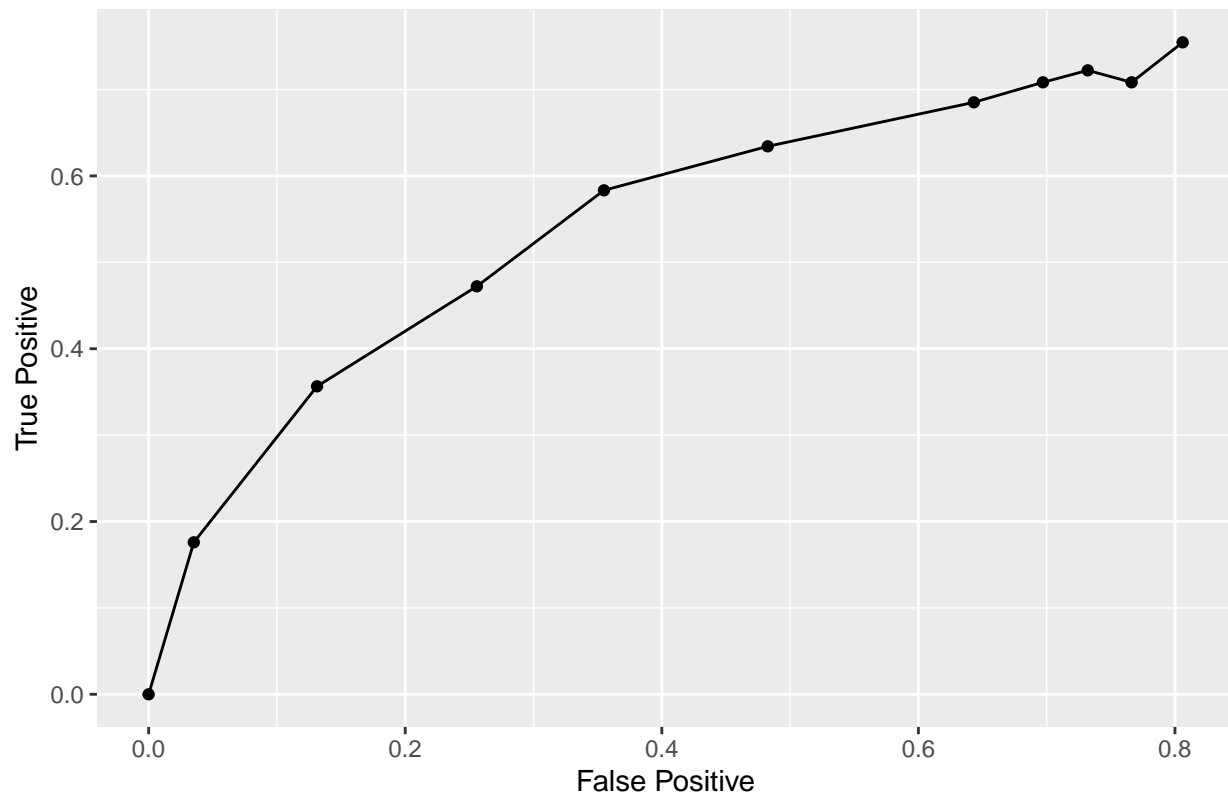




ROC at max depth 10



ROC at max depth 11



Overall, random forest in general does not do as well as other classifiers, with any decent level of sensitivity generally coming alongside a significant amount of false positives. Overall it seems like the best model is depth 8 with weight 200, with a sensitivity above 0.75 and a specificity around 0.65. Let's see how it does on the validation set, and what variables are important:

```
pred_rf_sens(cvdata,valid,depth=8,weight=200)
```

```
## [1] 0.8235294
```

```
pred_rf_spec(cvdata,valid,depth=8,weight=200)
```

```
## [1] 0.7088204
```

```
rf_final <- ranger(outcome ~ .-Race-Gender-AgeinYears, data = tbidata, num.trees=200, max.depth = 8, cla
imp <- importance(rf_final)
head(sort(imp,decreasing=T))
```

```
## SFxBasHem GCSVerbal ClavPar AMSSlow ActNorm AMSSleep
## 754.1699 448.4821 276.0896 269.7590 242.3190 240.6444
```

As it turns out, the model's sensitivity goes up to a respectable 85% when applied to the validation set, and its specificity increases to a decent 74%; probably good given that we would rather give an unnecessary test than miss a TBI.

The highest importance factors are the basilar hemorrhage, which makes sense given its high prevalence in other classifiers, and GCS verbal score, which also somewhat make sense, though it may be somewhat acting as a proxy for age. It is followed by parietal(scalp) trauma, acting normal, and an altered mental state of slowness- all things we would expect to have an effect given domain knowledge. Unfortunately, random forests do not give coefficients in the traditional sense, so we cannot exactly determine the direction each predictor's association with TBI goes, but we can infer most of those from domain knowledge and common sense.