random-forest

JackMoorer

```r
train <- read.csv("/Users/jackmoorer/Stat154/Projects/Project/data/clean_train.csv", header = TRUE)
test <- read.csv("/Users/jackmoorer/Stat154/Projects/Project/data/clean_test.csv", header = TRUE)
```

```r
library(ggplot2)
library(randomForest)
```

```
## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(caret)
```

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

Train Set

```r
train_forest <- train[, -ncol(train)]
```

First I tried just the basic random forest to see the results

```r
set.seed(200)
rf <- randomForest(Over50k ~., data = train_forest, imporatance = TRUE)
```

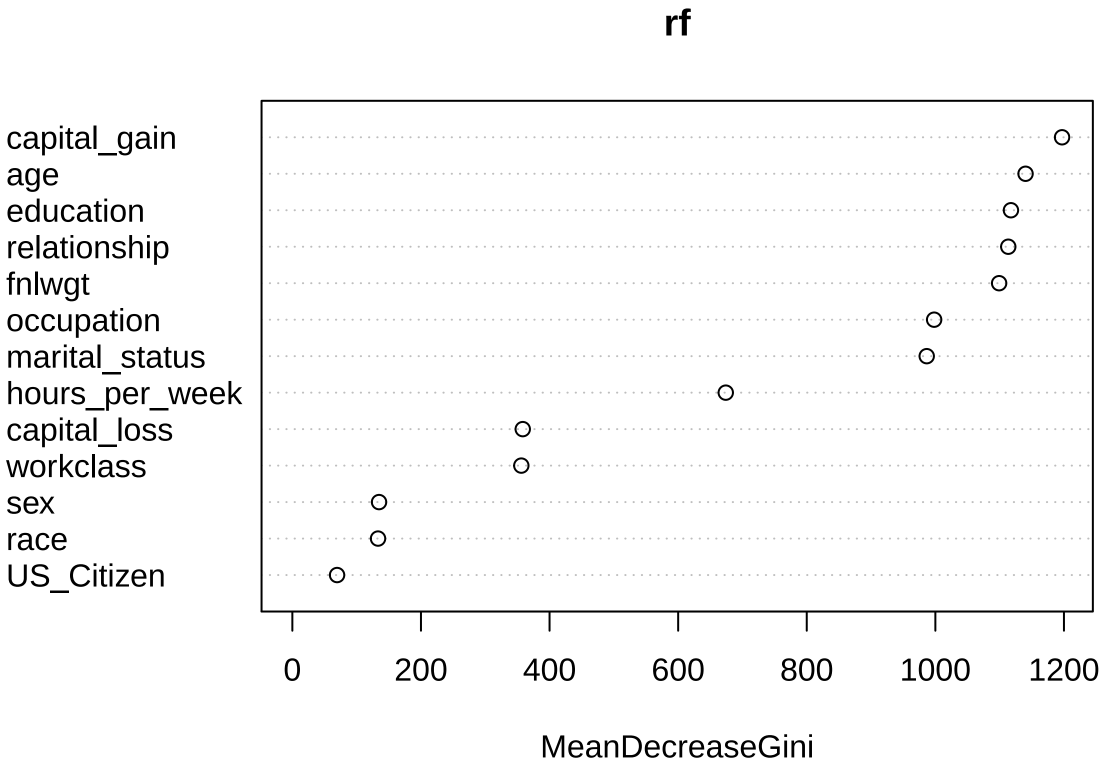
```r
rf$confusion
```

```
##        No  Yes class.error
## No  21177 1477   0.0651982
## Yes  2622 4886   0.3492275
```

```r
importance(rf)
```

```
##                MeanDecreaseGini
## age                  1140.26647
## workclass             356.09448
## fnlwgt               1099.15031
## education            1117.53836
## marital_status        986.55330
## occupation            998.11101
## relationship         1113.39050
## race                  133.24092
## sex                   134.81105
## capital_gain         1197.05878
## capital_loss          358.29773
## hours_per_week        674.04817
## US_Citizen             69.55423
```

```r
varImpPlot(rf)
```




```r
train_predictors <- train_forest[, -ncol(train_forest)]
```

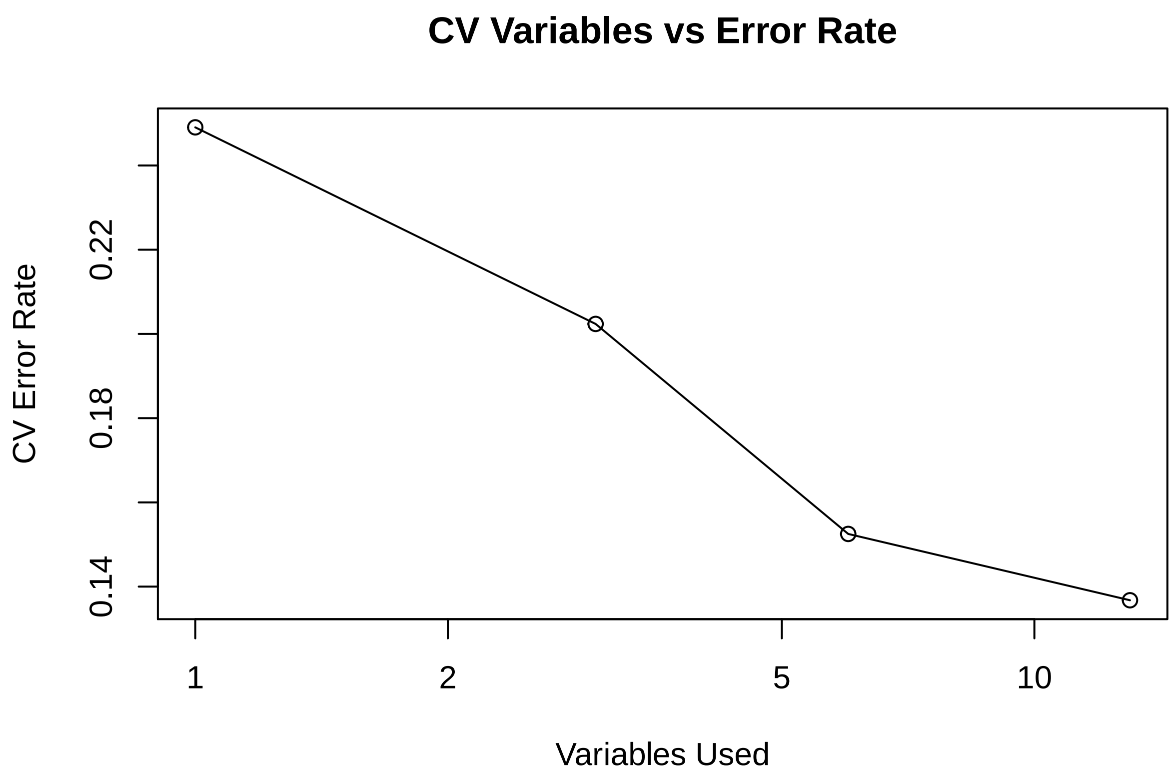
```r
train_response <- train_forest$Over50k
```

Next I did cross validation to tune the number of predictors used while building a tree

```r
set.seed(200)
rf_cv <- rfcv(train_predictors, train_response, cv.fold = 5)
```

The plot below shows the best number of predictors to use is all 13.

```r
with(rf_cv, plot(n.var, error.cv, log="x", type="o", lwd=1, main = "CV Variables vs Error Rate", xlab =
```




```
rf_cv$error.cv[[1]]
```

```
## [1] 0.1367615
```

```
num_var <- as.numeric(names(which.min(rf_cv$error.cv)))
print(num_var)
```

```
## [1] 13
```

```
#this takes too long
set.seed(200)
ntree <- c(50, 100, 500, 1000, 1500)
matrix <- matrix(rep(0, 4*5*3), ncol = 3, nrow = 20)
i <- 0
for (tree in ntree) {
  rf_cv <- rfcv(train_predictors, train_response, cv.fold = 3, ntree  = tree)
  for (j in 1:length(rf_cv$n.var)) {
    matrix[i + j, ] <- c(tree, rf_cv$n.var[j], rf_cv$error.cv[[j]])
  }
  i <- i + j
}
```

```
cv_df <- data.frame(matrix)
names(cv_df) <- c("ntree", "num_var", "error_rate")
err_rate <- cv_df$error_rate
row_num <- as.numeric(which.min(err_rate))
best_ntree <- cv_df$ntree[row_num]
best_num_var <- cv_df$num_var[row_num]
```

```
set.seed(200)
ntree <- c(50, 100, 500, 1000, 1500)
cv_list <- as.list(rep(0, 5))
```

```r
names(cv_list) <- ntree
i <- 1
for (tree in ntree) {
  tune_param <- tuneRF(train_predictors, train_response, ntreeTry = tree, trace = FALSE, plot = FALSE)
  cv_list[[i]] <- tune_param
  i <- i + 1
}
```

```
## 0.003301108 0.05
## -0.06366423 0.05
## -0.011597 0.05
## -0.05146171 0.05
## -0.0004873294 0.05
## -0.05165692 0.05
## 0.008464329 0.05
## -0.04353083 0.05
## 0.002679006 0.05
## -0.04627375 0.05
```

```r
cv_list
```

```
## $`50`
##       mtry  OOBError
## 2.OOB    2 0.1401432
## 3.OOB    3 0.1406074
## 6.OOB    6 0.1495590
##
## $`100`
##       mtry  OOBError
## 2.OOB    2 0.1388171
## 3.OOB    3 0.1372256
## 6.OOB    6 0.1442875
##
## $`500`
##       mtry  OOBError
## 2.OOB    2 0.1361316
## 3.OOB    3 0.1360652
## 6.OOB    6 0.1430940
##
## $`1000`
##       mtry  OOBError
## 2.OOB    2 0.1359326
## 3.OOB    3 0.1370930
## 6.OOB    6 0.1430608
##
## $`1500`
##       mtry  OOBError
## 2.OOB    2 0.1357669
## 3.OOB    3 0.1361316
## 6.OOB    6 0.1424309
```

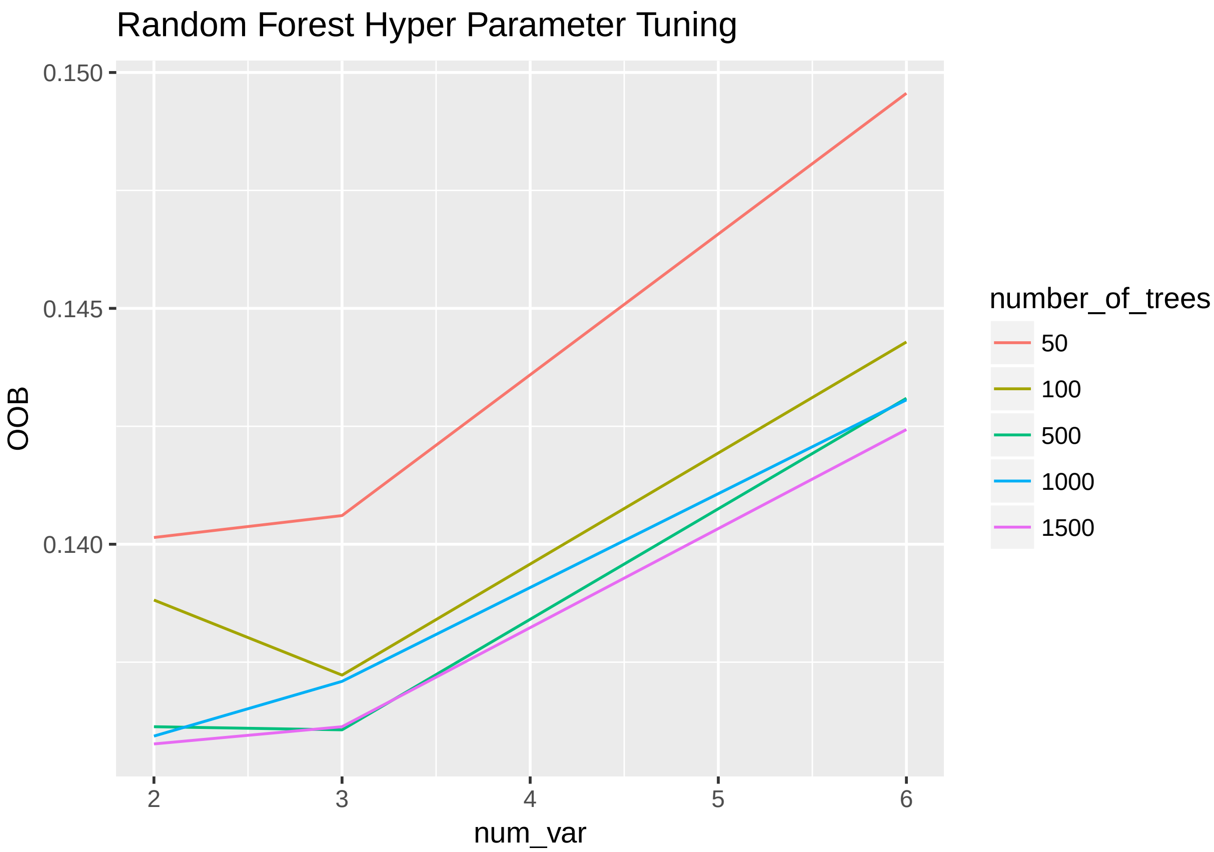
```r
library(stringr)
matrix <- matrix(rep(0, 3*5*3), nrow = 15, ncol = 3)
index <- 0
for (i in 1:5) {
```

```r
  cur_cv <- cv_list[i]
  errs <- cv_list[[i]][,2]
  for (j in 1:length(errs)){
    cur_err <- cv_list[[i]][,2][[j]]
    val <- str_sub(names(cv_list[[i]][,2][j]), 1, 1)
    val <- as.numeric(val)
    matrix[index + j, ] <- c(ntree[i], val, cur_err)
  }
  index <- index + j
}
cv_df <- data.frame(matrix)
names(cv_df) <- c("ntree", "num_var", "OOB")
cv_df
```

```
##    ntree num_var       OOB
## 1     50       2 0.1401432
## 2     50       3 0.1406074
## 3     50       6 0.1495590
## 4    100       2 0.1388171
## 5    100       3 0.1372256
## 6    100       6 0.1442875
## 7    500       2 0.1361316
## 8    500       3 0.1360652
## 9    500       6 0.1430940
## 10  1000       2 0.1359326
## 11  1000       3 0.1370930
## 12  1000       6 0.1430608
## 13  1500       2 0.1357669
## 14  1500       3 0.1361316
## 15  1500       6 0.1424309
```

```r
library(ggplot2)
cv_df$number_of_trees <- factor(cv_df$ntree)
```

```r
ggplot(data = cv_df, aes(x = num_var, y = OOB, color = number_of_trees)) + geom_line() + ggtitle("Randon
```




```
oobs <- cv_df$OOB
row_num <- as.numeric(which.min(oobs))
num_tree <- cv_df[row_num, 1]
num_var <- cv_df[row_num, 2]
```

I am going to run the random forest using this parameter:

```
set.seed(100)
random_forest <- randomForest(Over50k ~., data = train_forest, ntree = num_tree, mtry = num_var, importa
```

```
print(random_forest)
```

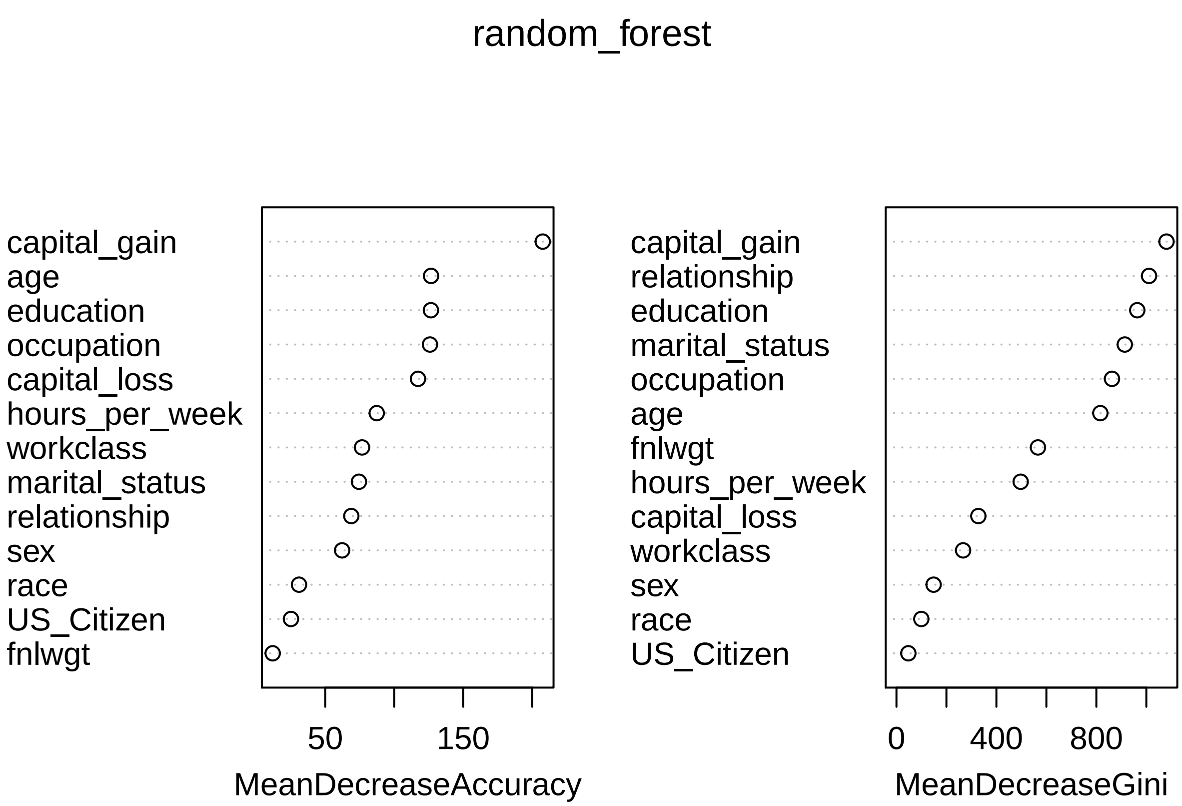
```
##
## Call:
##  randomForest(formula = Over50k ~ ., data = train_forest, ntree = num_tree,      mtry = num_var, impo
##                Type of random forest: classification
##                      Number of trees: 1500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 13.63%
## Confusion matrix:
##        No  Yes class.error
## No  21308 1346  0.05941556
## Yes  2764 4744  0.36814065
```

```
importance(random_forest)
```

```
##                        No         Yes MeanDecreaseAccuracy
## age              2.916181 117.7812947            126.71208
```

```
## workclass        67.458005  27.2146935             76.69084
## fnlwgt           13.726050   0.5755618             11.91015
## education        82.102928  96.6243220            126.61319
## marital_status   75.476360  44.6633592             74.44127
## occupation       70.493598  96.1170909            125.93797
## relationship     45.553450  66.4228965             68.89658
## race             19.108144  20.2750338             31.00323
## sex              44.129040  10.9439606             62.22469
## capital_gain    172.812802 216.5955580            207.66669
## capital_loss     88.886756 115.6337429            117.23932
## hours_per_week   18.625739  84.8279608             87.33325
## US_Citizen       27.569738   1.2870478             25.15558
##                 MeanDecreaseGini
## age                    815.99695
## workclass              266.76014
## fnlwgt                 566.25831
## education              963.76462
## marital_status         913.81883
## occupation             862.54477
## relationship          1010.80850
## race                    99.57915
## sex                    148.65378
## capital_gain          1080.78006
## capital_loss           327.79417
## hours_per_week         497.17865
## US_Citizen              47.64475
```

```
varImpPlot(random_forest)
```



```r
rf_pred <- predict(random_forest, train_forest[, -ncol(train_forest)])
```

```r
real_train <- train_forest$Over50k
err_rate <- mean(rf_pred != real_train)
err_rate
```

```
## [1] 0.08938399
```

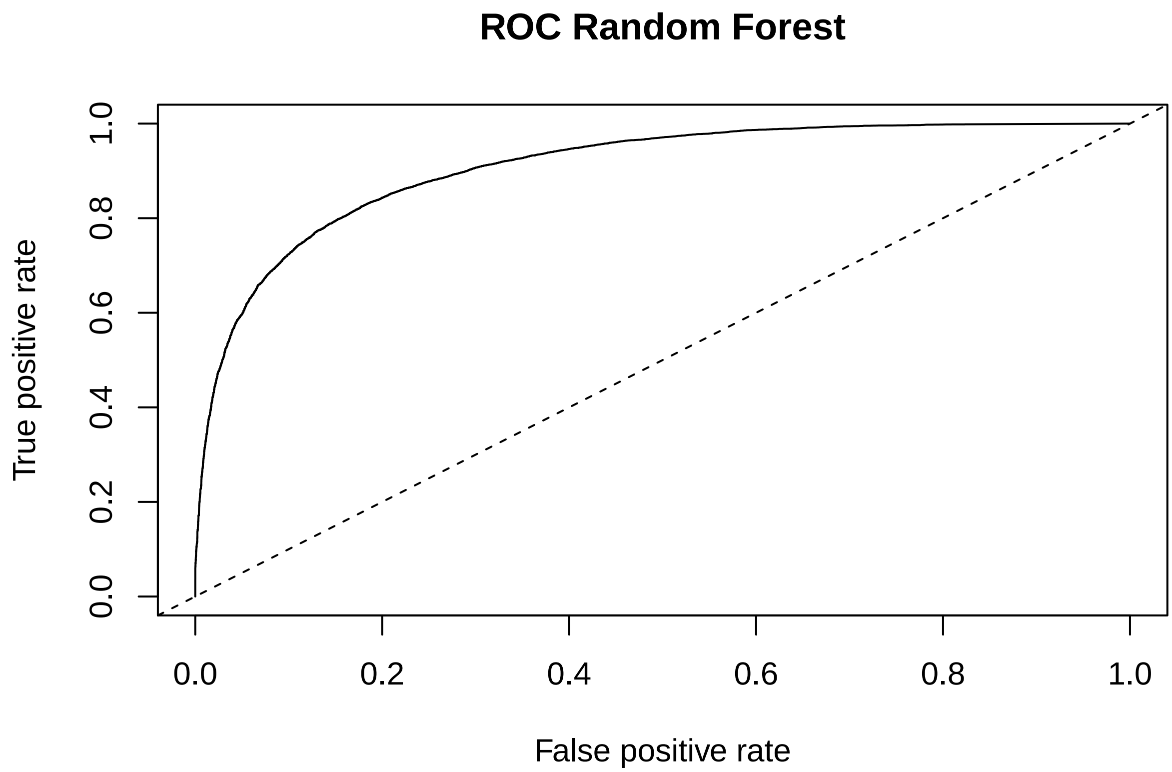
```r
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
probs <- as.vector(random_forest$votes[,2])
```

```r
prediction <- prediction(probs, real_train)
performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
```

```r
plot(performance, main="ROC Random Forest")
abline(a=0, b=1, lty = 2)
```




```r
auc <- performance(prediction, measure="auc")@y.values[[1]]
auc
```

```
## [1] 0.9071512
```

```r
print(random_forest$confusion)
```

```
##        No  Yes class.error
## No  21308 1346  0.05941556
## Yes  2764 4744  0.36814065
```

Test Set

```r
test_forest <- test[, -ncol(test)]
```

```r
test_forest_preds <- test_forest[, -ncol(test_forest)]
real_test <- test_forest$Over50k
```

```r
rf_test_preds <- predict(random_forest, test_forest_preds)
err_rate <- mean(rf_test_preds != real_test)
err_rate
```

```
## [1] 0.1394422
```

```r
confusionMatrix <- table(rf_test_preds, real_test)
confusionMatrix
```

```
##              real_test
## rf_test_preds    No   Yes
##           No  10638  1378
##           Yes   722  2322
```

```r
sensitivity <- confusionMatrix[2, 2]/(confusionMatrix[2, 2] + confusionMatrix[1, 2])
sensitivity
```

```
## [1] 0.6275676
```

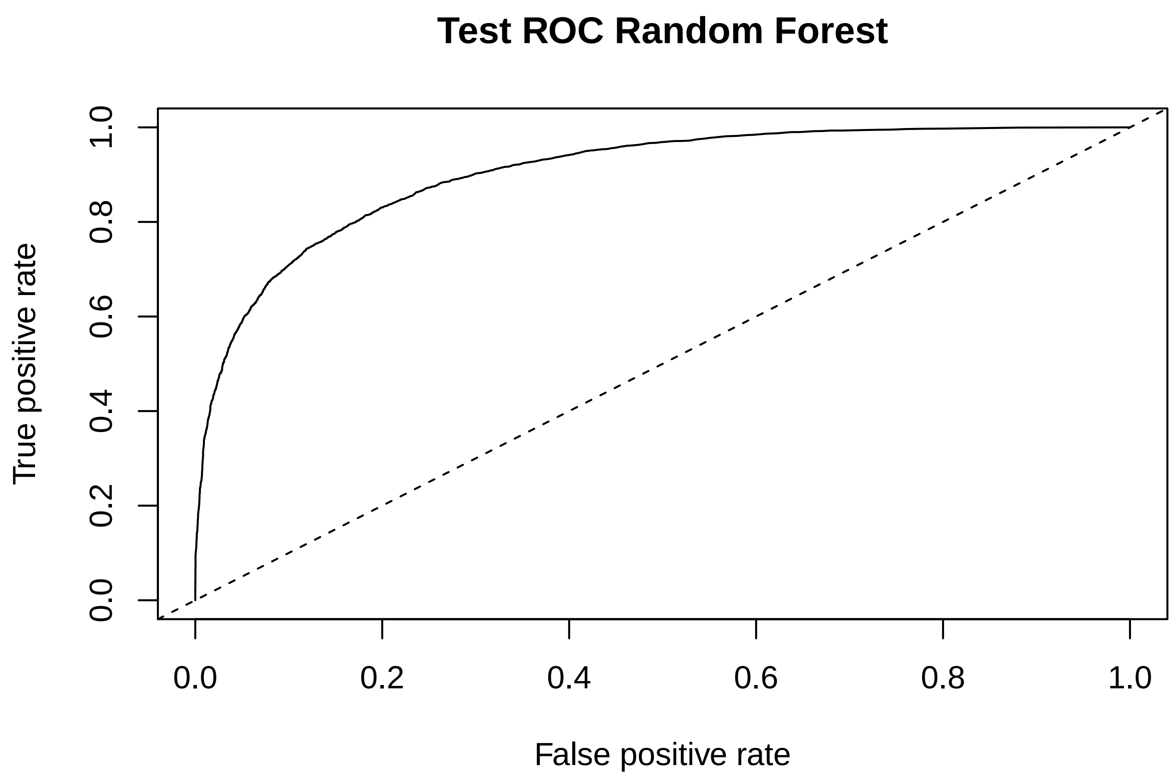
```r
specificity <- confusionMatrix[1, 1]/(confusionMatrix[1, 1] + confusionMatrix[2, 1])
specificity
```

```
## [1] 0.9364437
```

```r
test_probs <- predict(random_forest, test_forest_preds, type = "prob")
```

```r
test_prediction <- prediction(test_probs[,2], real_test)
test_performance <- performance(test_prediction,  measure = "tpr", x.measure = "fpr")
```

```r
plot(test_performance, main="Test ROC Random Forest")
abline(a=0, b=1, lty=2)
```




```
auc <- performance(test_prediction, measure="auc")@y.values[[1]]
auc
```

```
## [1] 0.9031879
```