Classification for phone prices

XIAO MA, HAO ZHENG, YONGZI YU

05/09/2022

# library(reticulate)  
library(caret)  
library(tidyverse)  
library(ggpubr)  
library(doParallel)  
library(ranger)  
library(pROC)  
library(gbm)  
library(pdp)  
library(lime)  
library(cutpointr)

# read data  
df = read.csv("data/train.csv")  
  
# covert outcome to binary  
df$price\_range = as.factor(ifelse(df$price\_range >=2, "High", "Low"))  
  
# convert data format  
df = df %>%   
 mutate\_at(vars("blue", "dual\_sim", "four\_g", "three\_g", "touch\_screen", "wifi"),   
 ~factor(., levels = c(0, 1), labels = c("No", "Yes")))

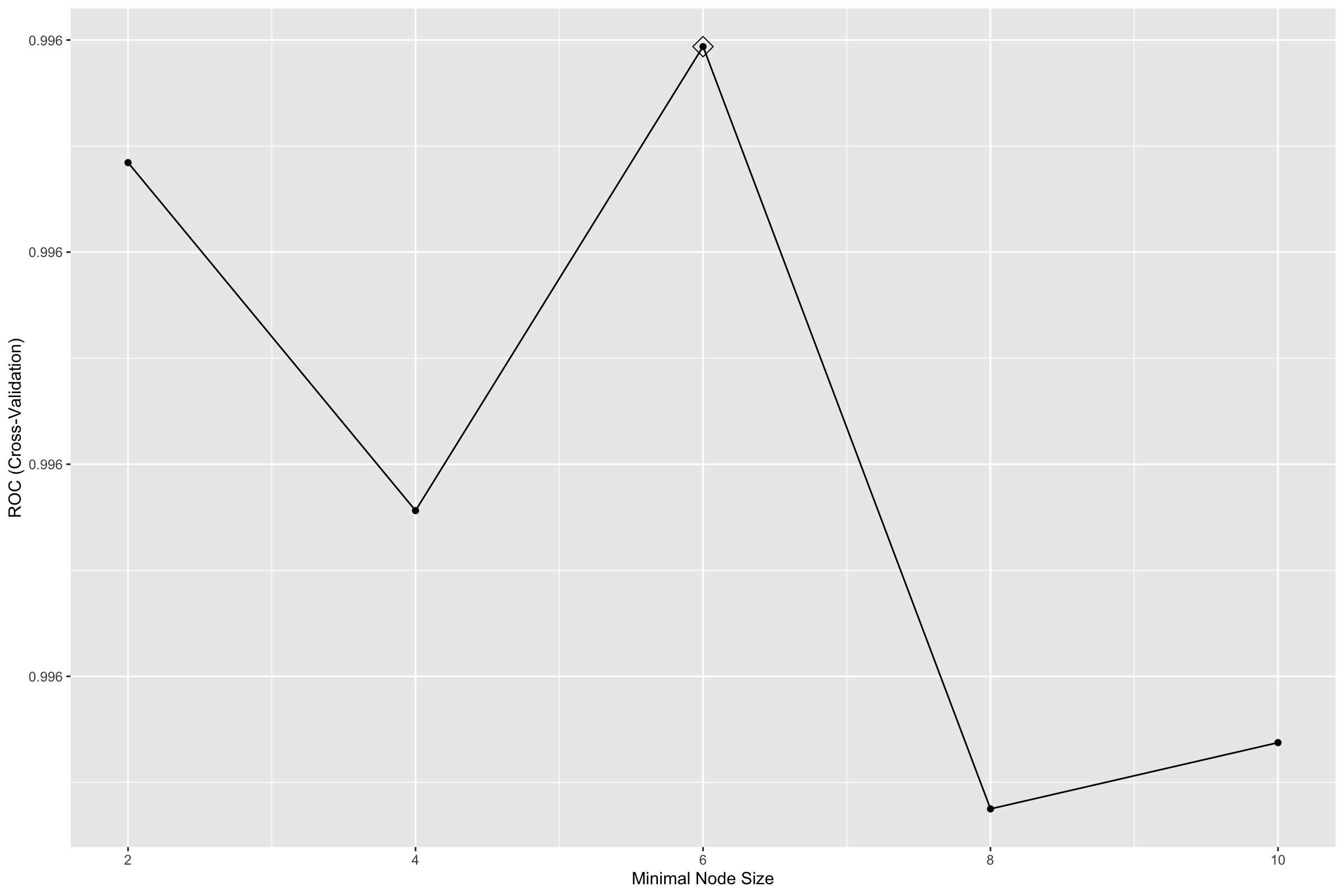
# split into training set  
set.seed(1)  
train\_index = createDataPartition(df$price\_range,p=0.8,list = F)  
train\_df = df[train\_index, ]  
test\_df = df[-train\_index, ]

# user parallel to accelarate   
cl <- makePSOCKcluster(4)  
registerDoParallel(cl)

## Bagging

ctrl <- trainControl(method = "cv", classProbs = TRUE,   
 summaryFunction = twoClassSummary)   
  
bagging.grid <- expand.grid(mtry = 20,   
 splitrule = "gini",  
 min.node.size = seq(from = 2, to = 10, by = 2))  
set.seed(1)   
bagging.fit <- train(price\_range ~ . ,   
 df,   
 subset = train\_index,   
 tuneGrid = bagging.grid,   
 method = "ranger",  
 metric = "ROC",  
 trControl = ctrl)

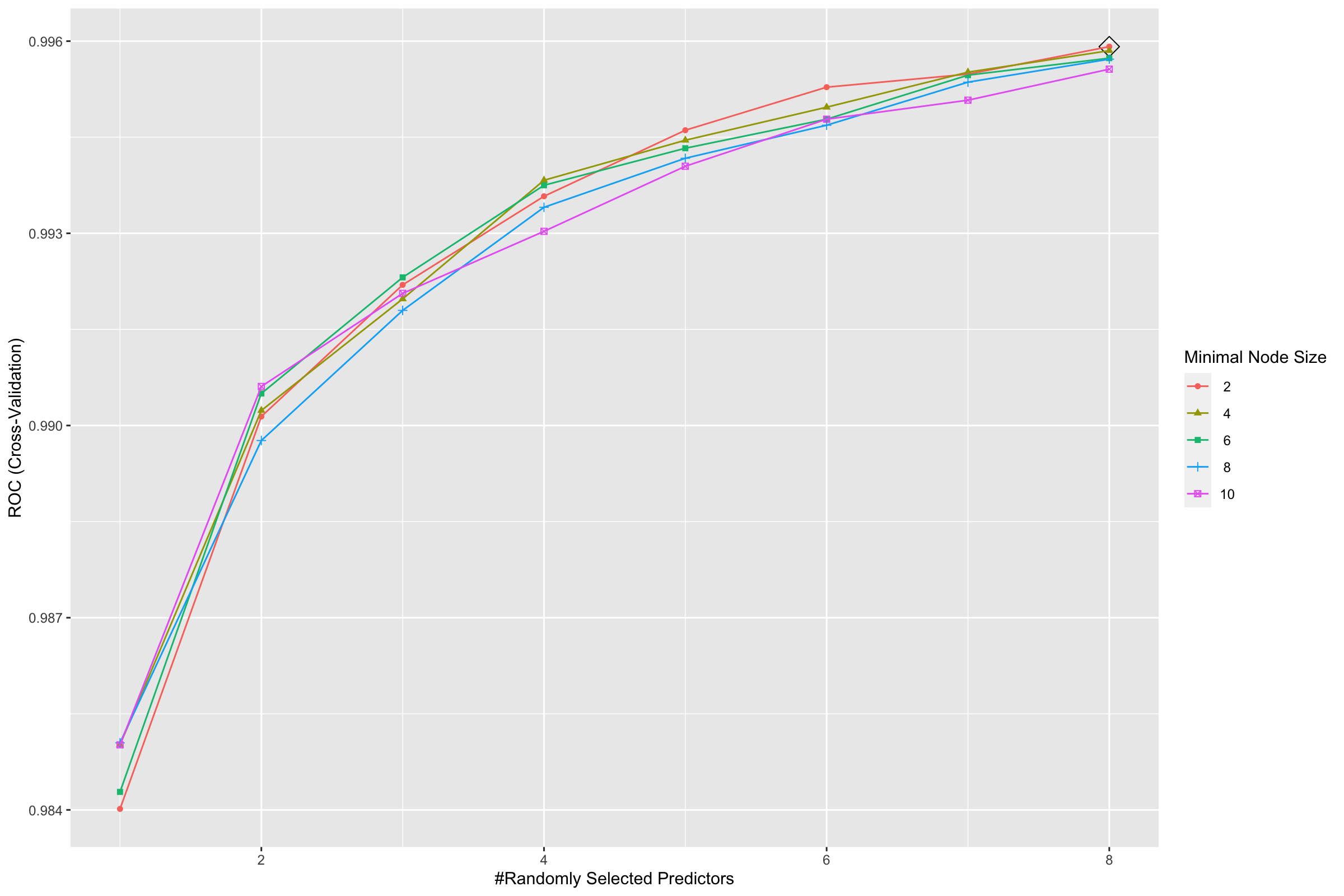
ggplot(bagging.fit, highlight = TRUE)



## Random forest

ctrl <- trainControl(method = "cv", classProbs = TRUE,   
 summaryFunction = twoClassSummary)   
  
rf.grid <- expand.grid(mtry = 1:8,   
 splitrule = "gini",  
 min.node.size = seq(from = 2, to = 10, by = 2))  
set.seed(1)   
rf.fit <- train(price\_range ~ . ,   
 df,   
 subset = train\_index,   
 tuneGrid = rf.grid,   
 method = "ranger",  
 metric = "ROC",  
 trControl = ctrl)

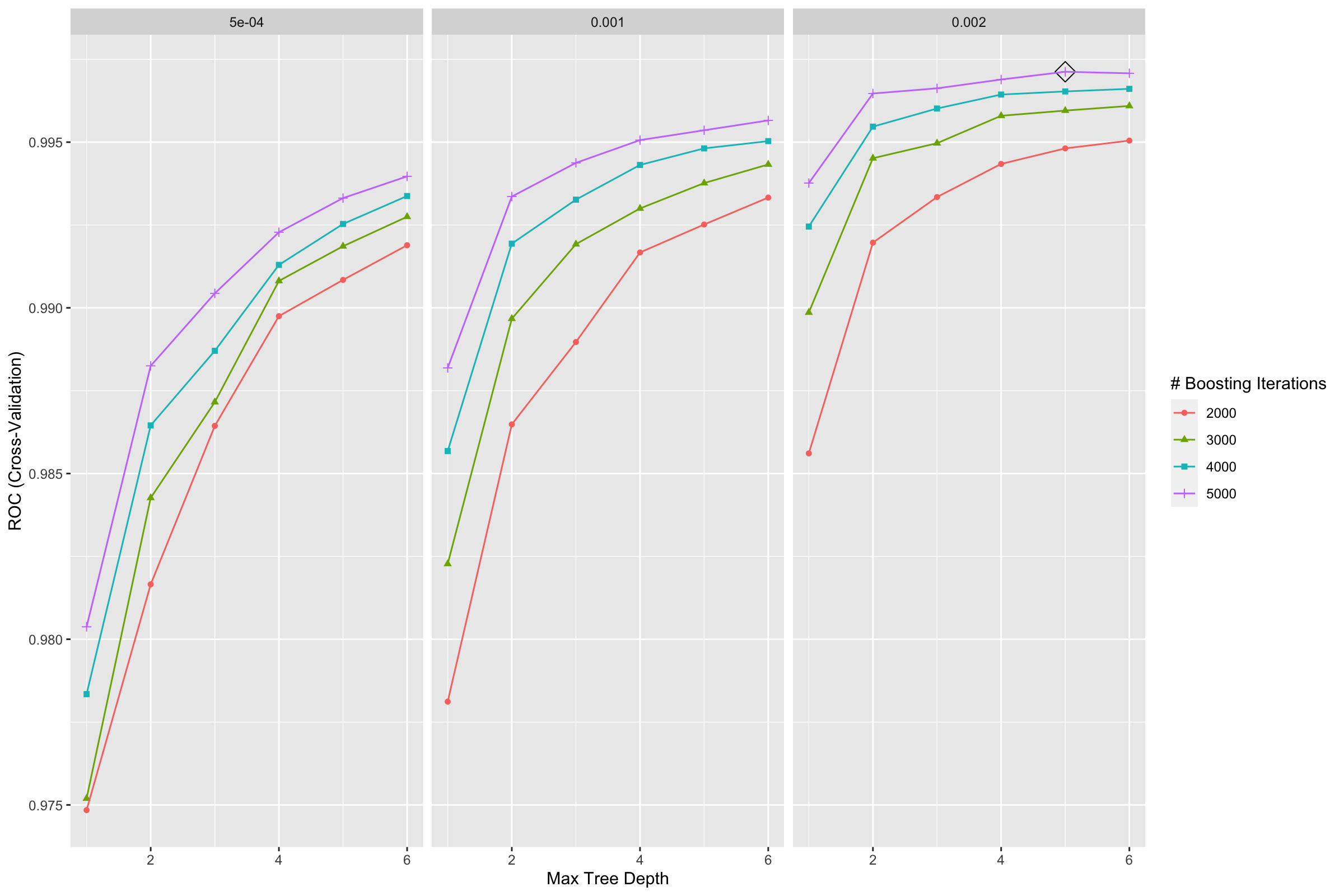
ggplot(rf.fit, highlight = TRUE)



## AdaBoost

gbm.grid <- expand.grid(n.trees = c(2000,3000,4000,5000),   
 interaction.depth = 1:6,   
 shrinkage = c(0.0005,0.001,0.002),   
 n.minobsinnode = 1)  
set.seed(1)   
gbm.fit <- train(price\_range ~ . ,   
 df,   
 subset = train\_index,   
 tuneGrid = gbm.grid,   
 trControl = ctrl,  
 method = "gbm",  
 distribution = "adaboost",  
 metric = "ROC",  
 verbose = FALSE)

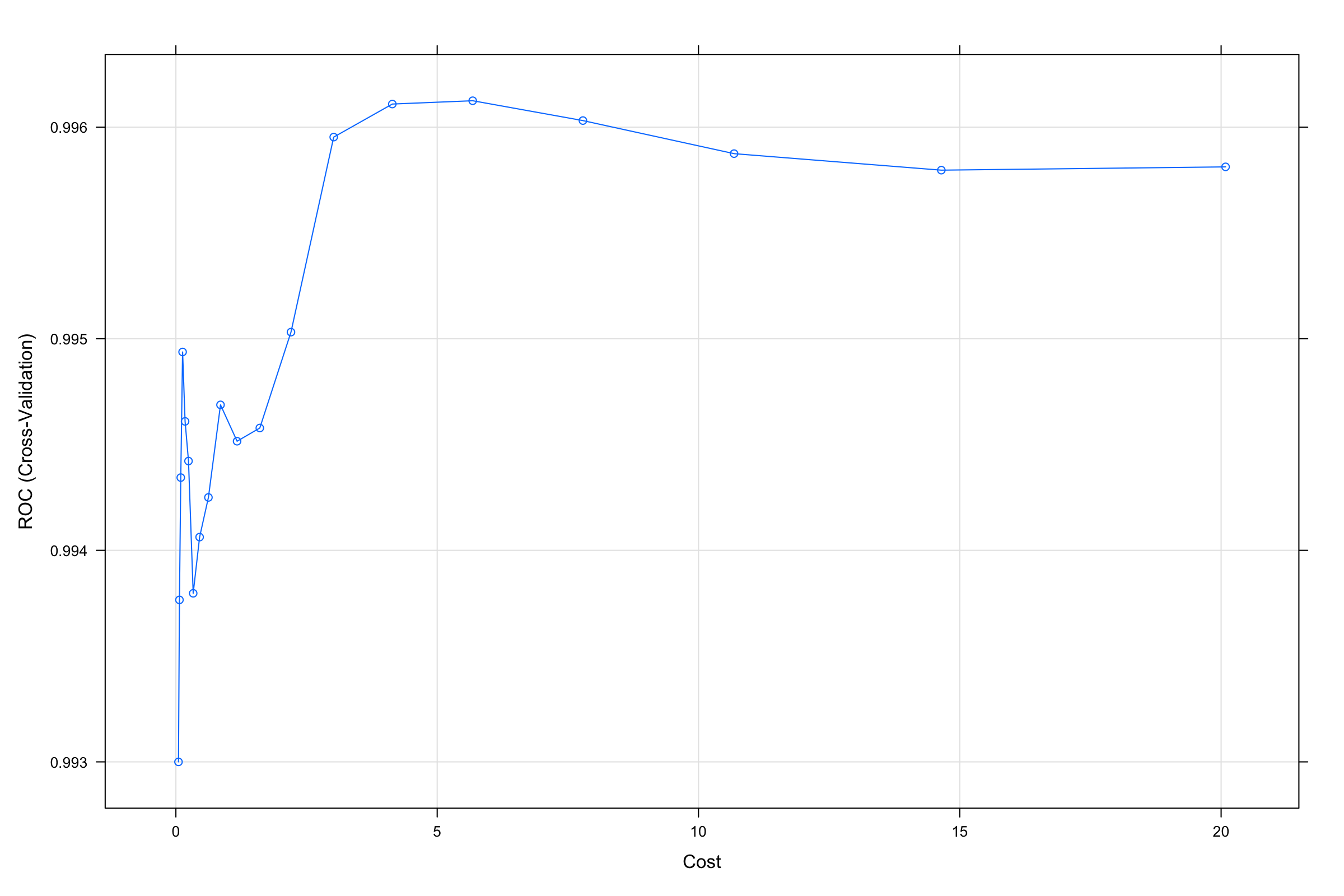
ggplot(gbm.fit, highlight = TRUE)



## SVM

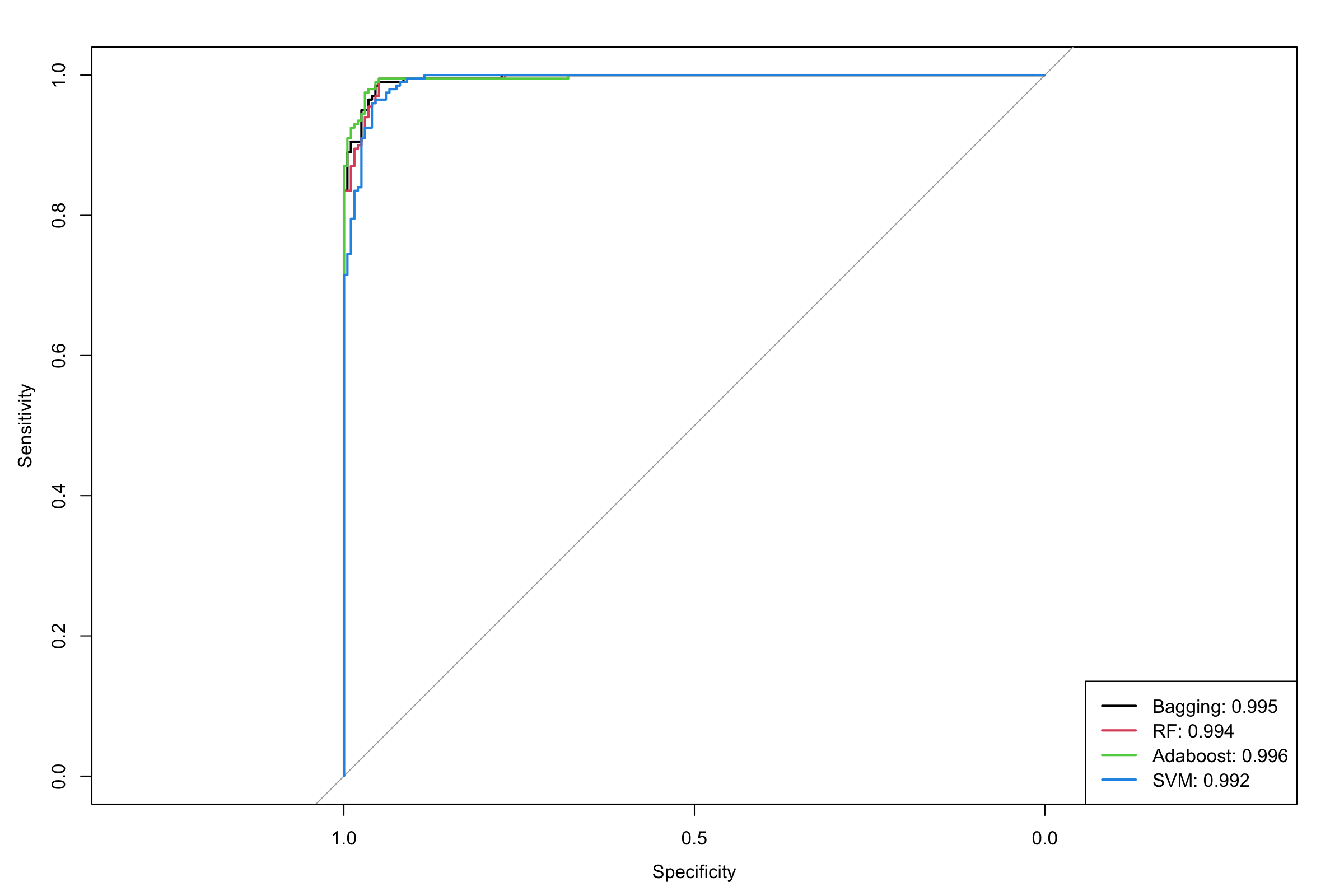
set.seed(1)  
svm.fit = train(price\_range ~ . ,   
 df,   
 subset = train\_index,   
 method = "svmRadialCost",  
 tuneGrid = data.frame(C = exp(seq(-3,3,len=20))),  
 trControl = ctrl,  
 metric = "ROC",  
 prob.model = TRUE,  
 verbose = FALSE)

plot(svm.fit)



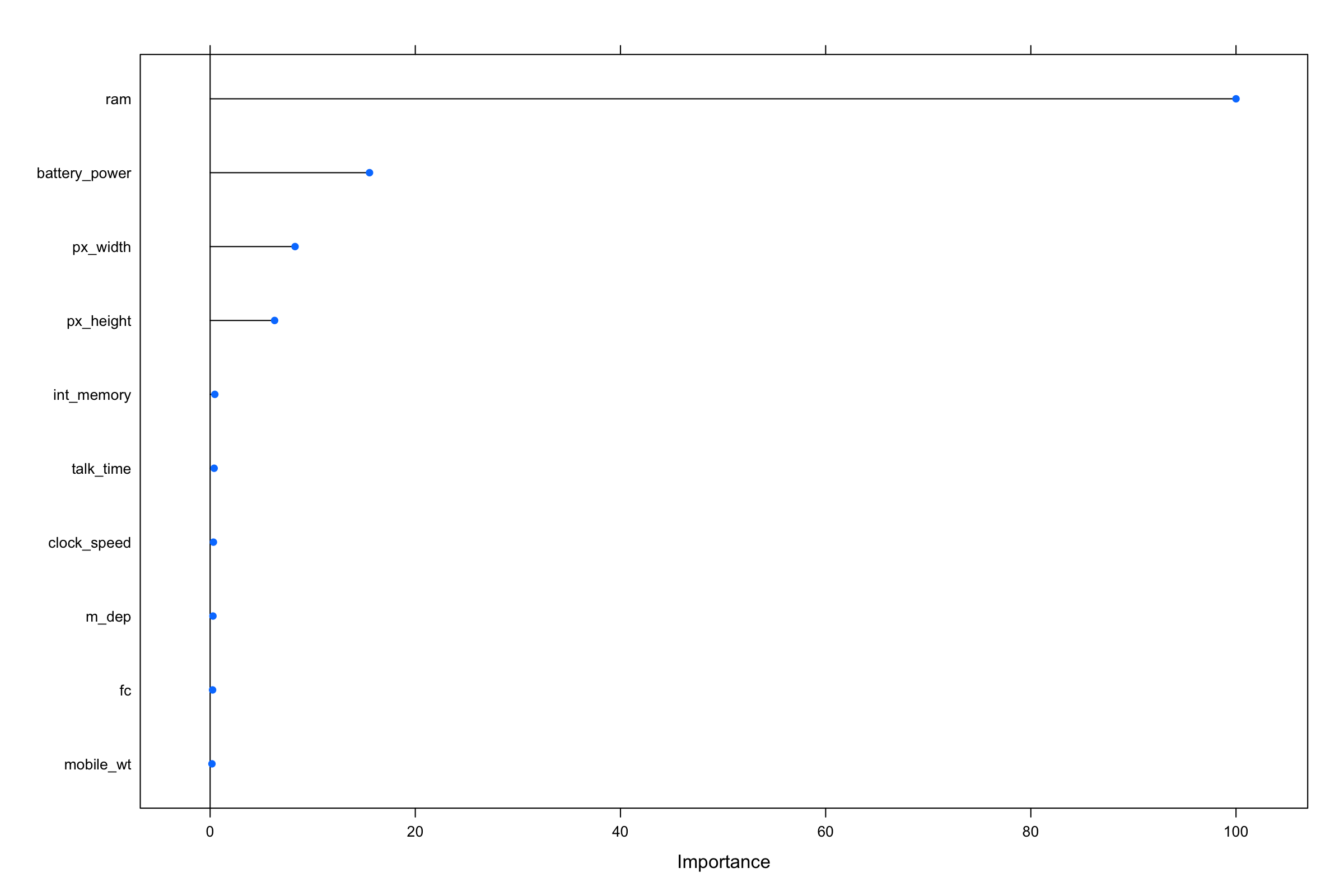
## ROC camparison

pred.bagging = predict(bagging.fit, newdata = df[-train\_index, ], type = "prob")[,1]  
roc.bagging = pROC::roc(df$price\_range[-train\_index], pred.bagging)  
pred.rf = predict(rf.fit, newdata = df[-train\_index, ], type = "prob")[,1]  
roc.rf = pROC::roc(df$price\_range[-train\_index], pred.rf)  
pred.gbm = predict(gbm.fit, newdata = df[-train\_index, ], type = "prob")[,1]  
roc.gbm = pROC::roc(df$price\_range[-train\_index], pred.gbm)  
pred.svm = predict(svm.fit, newdata = df[-train\_index, ], type = "prob")[,1]  
roc.svm = pROC::roc(df$price\_range[-train\_index], pred.svm)  
  
plot(roc.bagging, col = 1)   
plot(roc.rf, add = TRUE, col = 2)  
plot(roc.gbm, add = TRUE, col = 3)  
plot(roc.svm, add = TRUE, col = 4)  
auc <- c(roc.bagging$auc[1], roc.rf$auc[1], roc.gbm$auc[1], roc.svm$auc[1])  
modelNames <- c("Bagging", "RF","Adaboost", "SVM")   
legend("bottomright",   
 legend = paste0(modelNames, ": ",   
 round(auc,3)),   
 col = 1:4, lwd = 2)



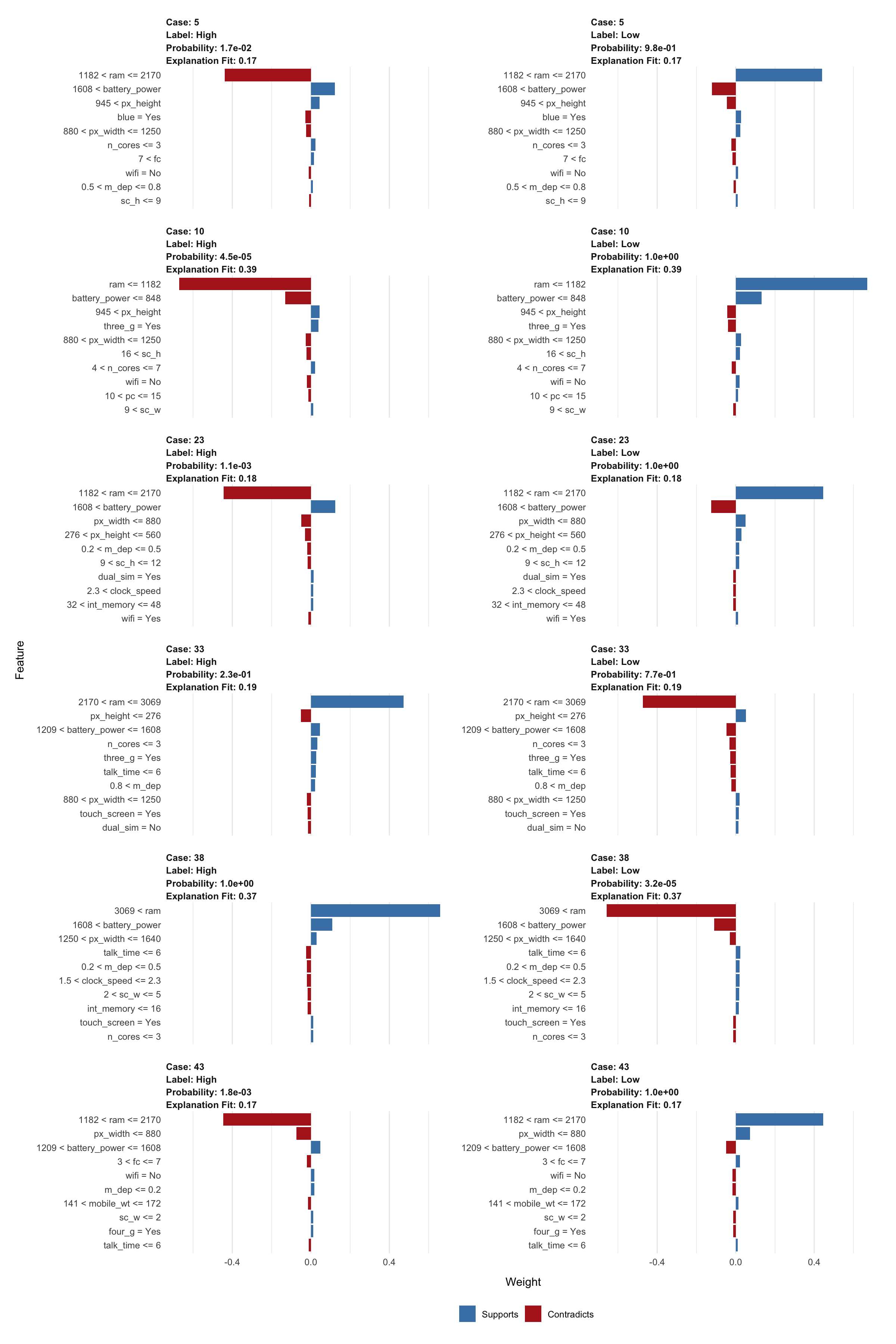
## Global Importance

gbmImp <- varImp(gbm.fit, scale = TRUE)  
plot(gbmImp, top = 10)



## LIME

explainer.rf <- lime(df[train\_index, -21], gbm.fit)  
new\_obs = df[-train\_index, -21][1:6, ]  
explaination.obs = explain(new\_obs,   
 explainer = explainer.rf,  
 n\_features = 10,  
 n\_labels = 2)  
plot\_features(explaination.obs)



## Prediction error

pred.gbm.train = predict(gbm.fit, newdata = df[train\_index, ], type = "prob")[, 1]  
train\_df$pred.gbm = pred.gbm.train  
cp <- cutpointr(train\_df, pred.gbm, price\_range,   
 method = maximize\_metric, metric = sum\_sens\_spec)  
summary(cp)

## Method: maximize\_metric   
## Predictor: pred.gbm   
## Outcome: price\_range   
## Direction: >=   
##   
## AUC n n\_pos n\_neg  
## 1 1600 800 800  
##   
## optimal\_cutpoint sum\_sens\_spec acc sensitivity specificity tp fn fp tn  
## 0.458 2 1 1 1 800 0 0 800  
##   
## Predictor summary:   
## Data Min. 5% 1st Qu. Median Mean 3rd Qu. 95% Max. SD  
## Overall 1.62e-05 3.85e-05 1.51e-04 0.448593 0.4989 0.99988 1.000 1.000 0.4867  
## High 4.58e-01 8.93e-01 9.99e-01 0.999885 0.9819 0.99995 1.000 1.000 0.0629  
## Low 1.62e-05 3.14e-05 5.81e-05 0.000151 0.0159 0.00198 0.108 0.439 0.0535  
## NAs  
## 0  
## 0  
## 0

test\_df$pred.gbm = as.factor(ifelse(pred.gbm > cp$optimal\_cutpoint, "High", "Low"))  
cft = confusionMatrix(test\_df$pred.gbm, test\_df$price\_range)  
print(cft)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction High Low  
## High 192 4  
## Low 8 196  
##   
## Accuracy : 0.97   
## 95% CI : (0.948, 0.984)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.94   
##   
## Mcnemar's Test P-Value : 0.386   
##   
## Sensitivity : 0.960   
## Specificity : 0.980   
## Pos Pred Value : 0.980   
## Neg Pred Value : 0.961   
## Prevalence : 0.500   
## Detection Rate : 0.480   
## Detection Prevalence : 0.490   
## Balanced Accuracy : 0.970   
##   
## 'Positive' Class : High   
##

* In order to predict the high cost phone, we decided to build a binary classification model. We randomly divided our dataset into two data sets before training the classification algorithms: the training and the test sets. The training and test sets each included 80% and 20% of the total data, respectively.
* The parameters of each algorithm were determined based on the classification performance of the training set as measured by five-fold cross-validation. On the test set, the performance of all algorithms was tested and compared. We evaluated and compared the results of five different algorithms since different classification methods are better suited to different types of data. Bagging, random forest, ada boosting, and radical kernel SVM are among the models under consideration.
* We plotted the ROC curves of all the different algorithms on the test dataset. Over all reasonable sensitivity thresholds and recall thresholds, the ada boosting model is consistently better than all the other models. The feature importance of the ADA boosting model is scaled between 0 and 100. Random access memory (RAM) is the most important predictor since the importance value goes to 100. Battery power is 20% as important as RAM. Pixel height and pixel width are each around 10% as important as RAM. All the other predictors are less than 5% as important as RAM.
* For the first six test cases and label combinations, we utilized LIME to visually represent the explanations for the relationship between mobile phone price level and features. Positively associated features are displayed in blue, while negatively correlated features are displayed in red. For example, case 30, which refers to the row 30 of the test data, has the highest explanation fit 0.40. Label which is high means this case is for predicting the high price mobile phone. ‘RAM smaller than 1209’ feature which is red color implies the phone with this ‘RAM smaller than 1209’ feature has large possibility that it does not belong to the high price phone. For case 18, label which is high is for predicting the high price mobile phone. ‘RAM > 3033’ feature which is blue color implies the phone with this ‘RAM > 3033’ feature has large possibility that it belongs to high price phone. We can also observe that all of the predictors for the phone pricing outcome selected the same features, showing that these are important features both locally and globally. For example, these features include ram, battery power, talking time, pixel height, pixel width, 3g internet, WiFi, Bluetooth.
* We selected to maximize the sum of sensitivity and specificity in order to determine the best cut point for the prediction probability. On the training dataset, the best cut point is 0.458, which yields an accuracy of 1, a sensitivity of 1, and a specificity of 1. On the test dataset, we have an accuracy of 0.97, sensitivity of 0.96, and specificity of 0.98 using the optimal cut point. As a consequence, our model appears to be extremely effective in predicting high prices for phones.