

Predictive Modeling

Chapter 14: Classification Trees and Rule-Based Models

STA 6543

The University of Texas at San Antonio

Overview

- Part I: General Strategies
- Part II: Regression Models
 - Chapter 5: Measuring Performance in Regression Models
 - Chapter 6: Linear Regression and Its Cousins
 - Chapter 7: Nonlinear Regression Models
 - Chapter 8: Regression Trees and Rule-Based Models
- Part III: Classification Models
 - Chapter 11: Measuring Performance in Classification Models
 - Chapter 12: Discriminant Analysis and Other Linear Classification Models
 - Chapter 13: Nonlinear Classification Models
 - Chapter 14: Classification Trees and Rule-Based Models

Tree-based classification models

- Various types of tree-based classification models
 - Basic classification trees *✓ part / tree*
 - Bagged trees
 - Random forest
 - Boosted trees
- R demonstrations for the stock market data

Basic classification trees

Chapter 14: Classification Trees and Rule-Based Models

Overview

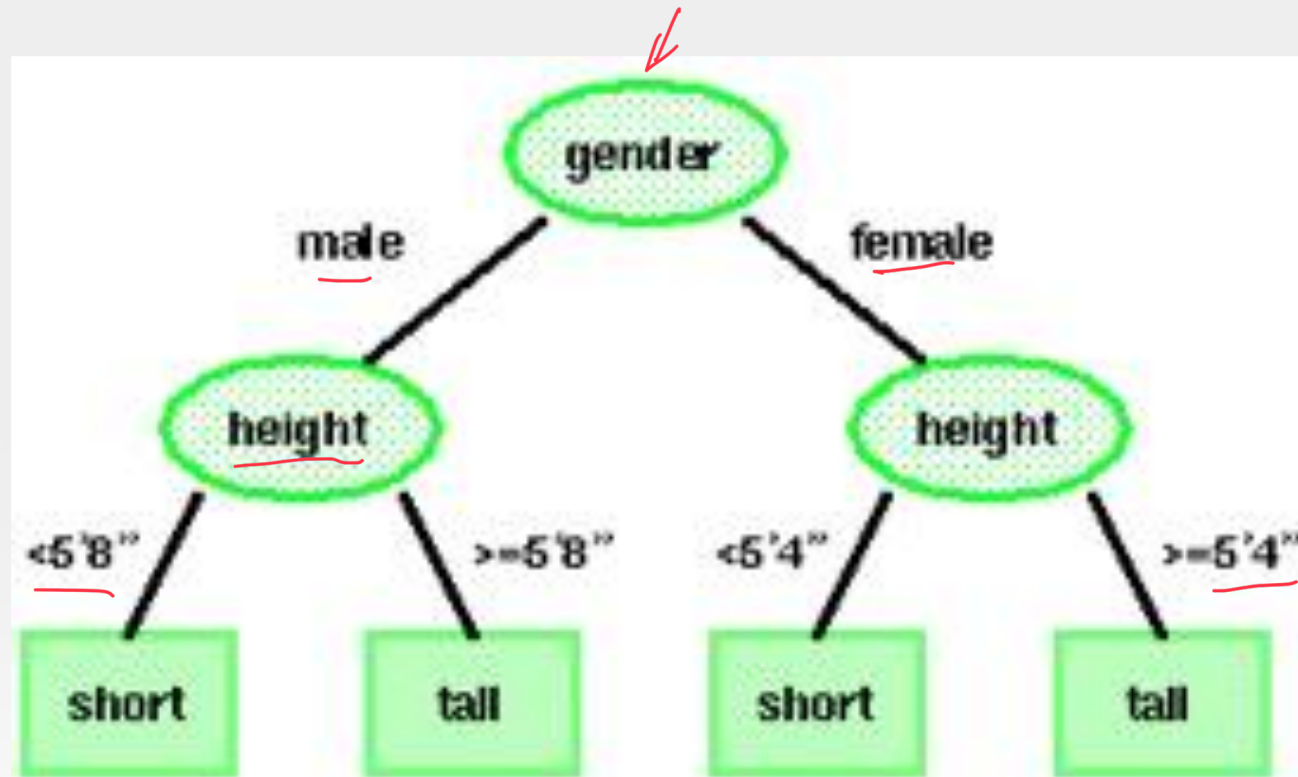
- In Chapter 8, we discussed tree-based models for regression.
- In this chapter, we study tree-based models for classification
 - Classification tree analysis: the predicted outcome is the class (discrete) to which the data belongs

Overview of a tree

- Consider a classification problem that involves nominal data.
 - Data described by a list of attributes (e.g., categorizing people as short or tall using gender, height, age, and ethnicity)
- How can we ^{x₁} use such ^{x₂} nominal ^{x₃} data ^{x₄} for classification?
- How can we learn the categories of such data?
- Nonparametric methods such as decision trees provide a way to deal with such data.

male/female: Gender
 True/false
 tail/short → ordinal
 $y = \begin{cases} 0 & \text{short} \\ 1 & \text{tall} \end{cases}$

Overview of a tree



Pros and cons

- Tree-based methods are simple and useful for interpretation.
- However, they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy.
- We could use bagging, random forests, and boosting to grow multiple trees which are then combined to yield a single consensus prediction.
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss interpretation.

rpart

- CART (Classification and Regression Trees) is developed by Breiman, Friedman, Olshen and Stone
 - CART is the trademarked name of a particular software implementation of these ideas
 - tree() has been used in R
- Hence, Recursive PARTitioning (rpart) was chosen
 - rpart has now become more common than the original and more descriptive “cart”
- An introduction of rpart() can be found [[here](#)]
- We look at the stock market data using rpart()

Classification trees

- A classification tree is very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one.
- For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.
- The changes needed in the tree algorithm pertain to the criteria for splitting nodes and pruning the tree.
- For regression we used the squared-error node impurity measure $Q_m = \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 / N_m$, but this is not suitable for classification.

RMSE

Node impurity measures for classification

- Let \hat{p}_{mk} represent the proportion of training observations in the m th region that are from the k th class

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

Indicator = $\begin{cases} 1 & y_i = k \\ 0 & y_i \neq k \end{cases}$

- Measures of node impurity include
 - Misclassification error: $Q_m = 1 - \max_k(\hat{p}_{mk})$
 - Gini index: $Q_m = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk})$
 - Cross-entropy: $Q_m = -\sum_{k=1}^K \hat{p}_{mk} \log(\hat{p}_{mk})$
- Misclassification error is not sufficiently sensitive for tree-growing, and in practice Gini index and cross-entropy are preferable.
- It turns out that the Gini index and the cross-entropy are very similar numerically.

Bernoulli distribution with \hat{p}_{mk}
 $\text{Var}(Y) = \hat{p}_{mk}(1 - \hat{p}_{mk})$

Summary of learning classification trees

- Advantages
 - Easily interpretable by human (as long as the tree is not too big)
 - Computationally efficient
 - Handles both numerical and categorical data
- Disadvantages
 - Finding partition of space that minimizes empirical error is NP-hard (nondeterministic polynomial hardness)
 - We resort to greedy approaches with limited theoretical underpinning
 - Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this course.

Summary of learning classification trees


- However, by aggregating many decision trees, using methods like **bagging**, **random forests**, and **boosting**, the predictive performance of trees can be substantially improved. We introduce these concepts in Chapter 8 for regression analysis.

The stock market data

```
# required packages
library(AppliedPredictiveModeling)
library(caret)
library(rpart)
library(ISLR) #the stock market data
library(ipred) #bagging
library(gbm) #boosting
library(randomForest) #random forest

#Access the data
attach(Smarket)

###Data splitting
train = which(Year<2005)
Smarket.train= Smarket[train,]; # observations before 2005 are served as training test data.
Smarket.test= Smarket[-train,]; # observations from 2005 are served as test data.
```



A train control function

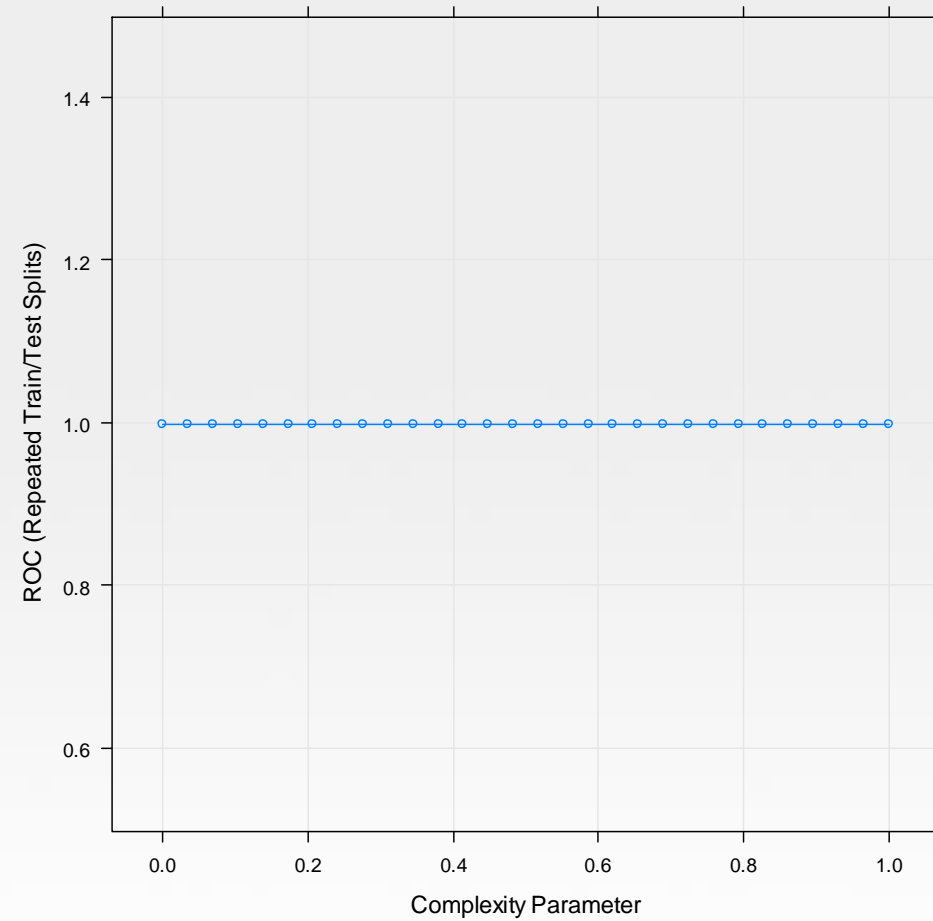
```
### Create a control function that will be used across models.  
set.seed(100)  
ctrl <- trainControl(method = "LGOVCV",  
                      summaryFunction = twoClassSummary,  
                      classProbs = TRUE,  
                      savePredictions = TRUE)  
  
# LGOVCV: Repeated Train/Test Splits Estimated (25 reps, 75%)
```

Classification trees

```
#####Classification Trees#####
set.seed(476)
rpartTune<- rtrain(x = as.matrix(Smarket.train[,1:8]),
  y = Smarket.train$Direction,
  method = "rpart",
  tuneLength = 30,
  metric = "ROC",
  trControl = ctrl)

rpartTune
plot(rpartTune)
```


The tuning parameter



Bagged trees

```
#####Bagged Trees#####  
set.seed(476)  
treebagTune <- train(x = as.matrix(Smarket.train[,1:8]),  
  y = Smarket.train$Direction,  
  method = "treebag",  
  nbagg = 50,  
  trControl = ctrl)  
  
treebagTune
```

Bagged tree output

```
> treebagTune
Bagged CART

998 samples
 8 predictor
 2 classes: 'Down', 'Up'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 750, 750, 750, 750, 750, 750, ...
Resampling results:
```

ROC	Sens	Spec
<u>0.9998361</u>	0.9980328	0.9980952

Boosting

```
#####Boosting#####
gbmGrid = expand.grid( interaction.depth = seq( 1, 7, by=2 ),
  n.trees = seq( 100, 1000, by=100 ),
  shrinkage = c(0.01, 0.1),
  n.minobsinnode = 10 )

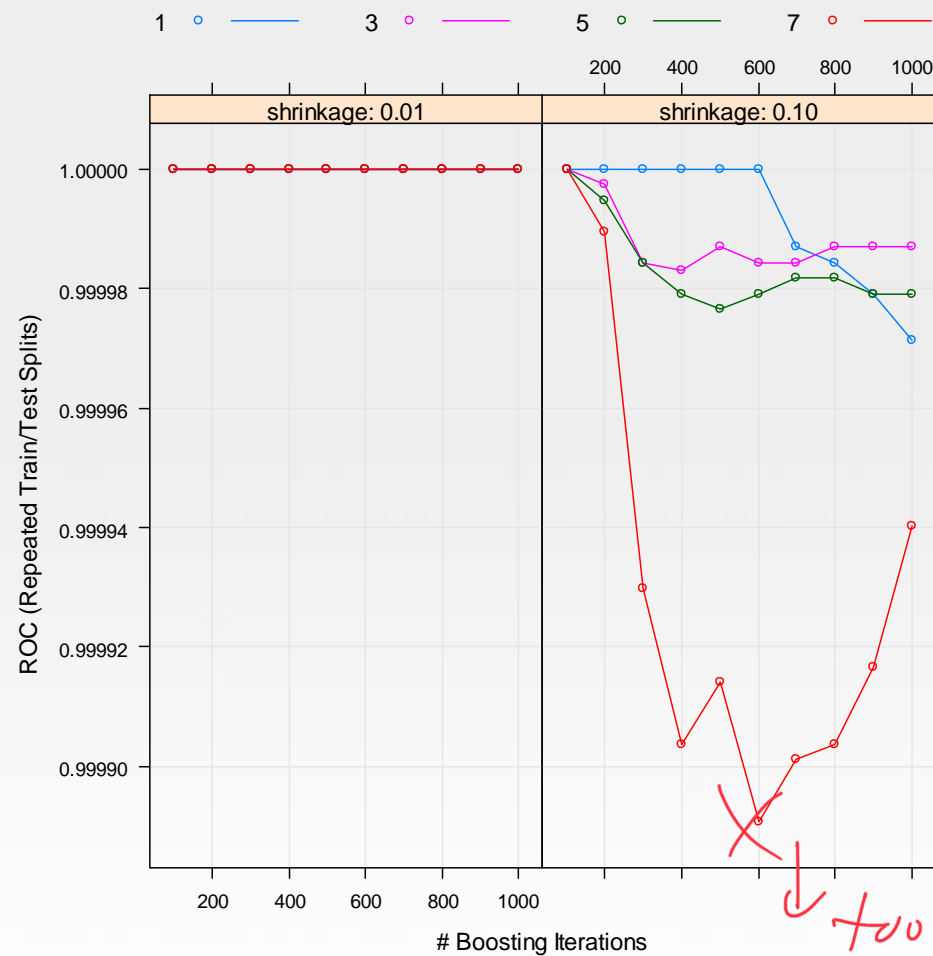
set.seed(476)
gbmTune <- train(x = as.matrix(Smarket.train[,1:8]),
  y = Smarket.train$Direction,
  method = "gbm",
  tuneGrid = gbmGrid,
  trControl = ctrl,
  verbose = FALSE)

gbmTune
plot(gbmTune, auto.key = list(columns = 4, lines = TRUE))
#variable importance
gbmImp <- varImp(gbmTune, scale = FALSE)
gbmImp
```

Handwritten notes:

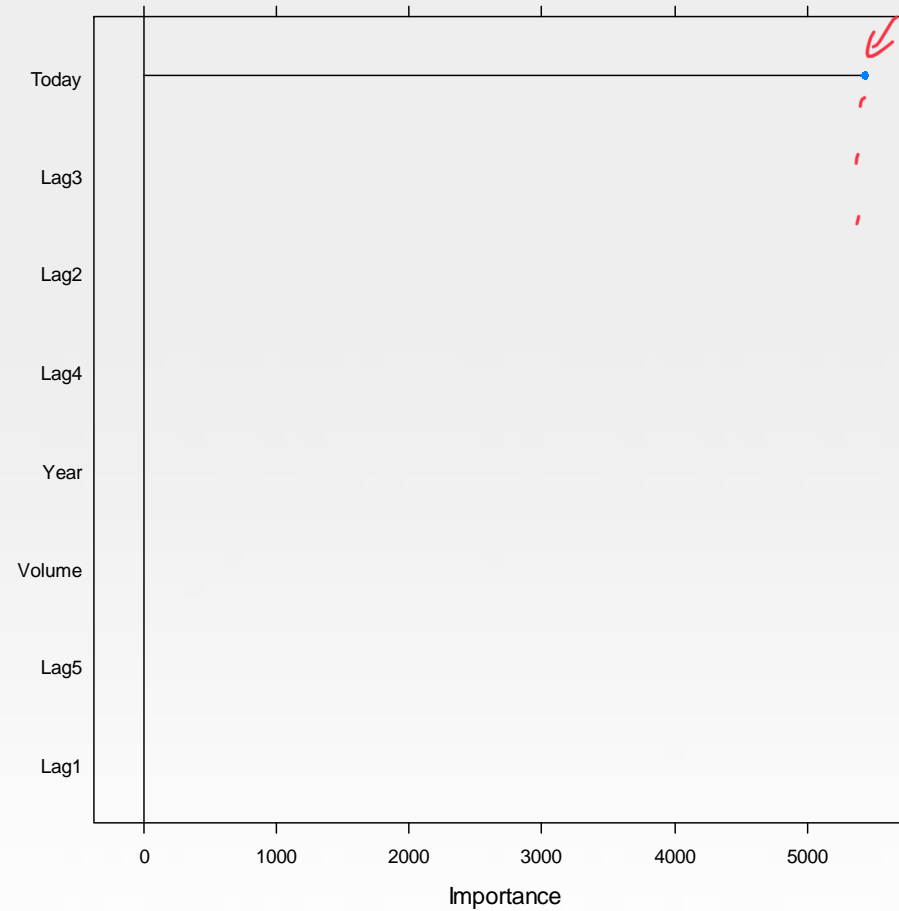
- $d=1$ (circled in red)
- small* (next to shrinkage)
- boosting* (next to method = "gbm")

The tuning parameter



too small for ROC

Variable importance



most important predictor

Radom forest

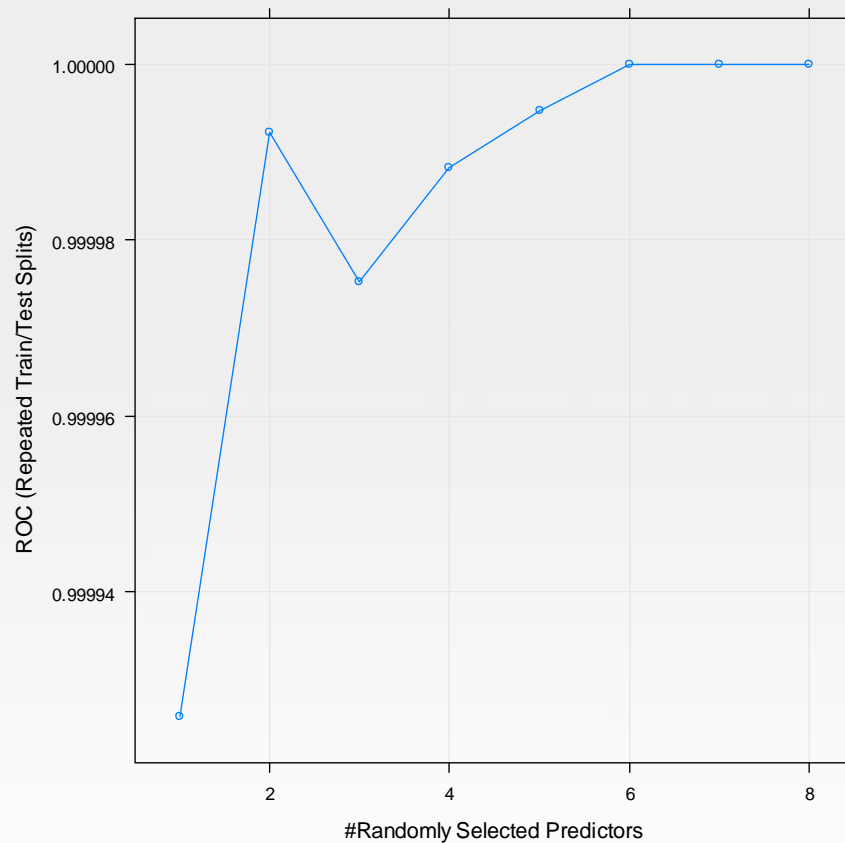
```
#####Random Forest#####
mtryGrid <- data.frame(mtry = 1:8) #since we only have 8 predictors

### Tune the model using cross-validation
set.seed(476)
rfTune <- train(x = as.matrix(Smarket.train[,1:8]),
               y = Smarket.train$Direction,
               method = "rf",
               tuneGrid = mtryGrid,
               ntree = 200,
               importance = TRUE,
               trControl = ctrl)

rfTune
plot(rfTune)
```

The tuning parameter (mtry = 6)

Optimal value



Prediction based on different models *for test data*

```
### Predict the test set based on four models
```

```
#Classification Trees
```

```
Smarket.test$rpart<- predict(rpartTune,Smarket.test, type = "prob"),1]
```

```
#Bagged Trees
```

```
Smarket.test$treebag <- predict(treebagTune,Smarket.test, type = "prob"),1]
```

```
#Boosting
```

```
Smarket.test$Boosting <- predict(gbmTune,Smarket.test, type = "prob"),1]
```

```
#Random Forest
```

```
Smarket.test$RF <- predict(rfTune,Smarket.test, type = "prob"),1]
```

predictors for the test data

#ROC for Classification Trees

RPARTROC <- roc(Smarket.test\$Direction, Smarket.test\$rpart)

plot(RPARTROC, col=1, lty=1, lwd=2)

response from the test data

↓ response from the rpart.

#ROC for Bagged Trees

TREEBAGROC <- roc(Smarket.test\$Direction, Smarket.test\$treebag)

lines(TREEBAGROC, col=2, lty=2, lwd=2)

#ROC for Boosting

BoostingROC <- roc(Smarket.test\$Direction, Smarket.test\$Boosting)

lines(BoostingROC, col=3, lty=3, lwd=2)

#ROC for Random Forest

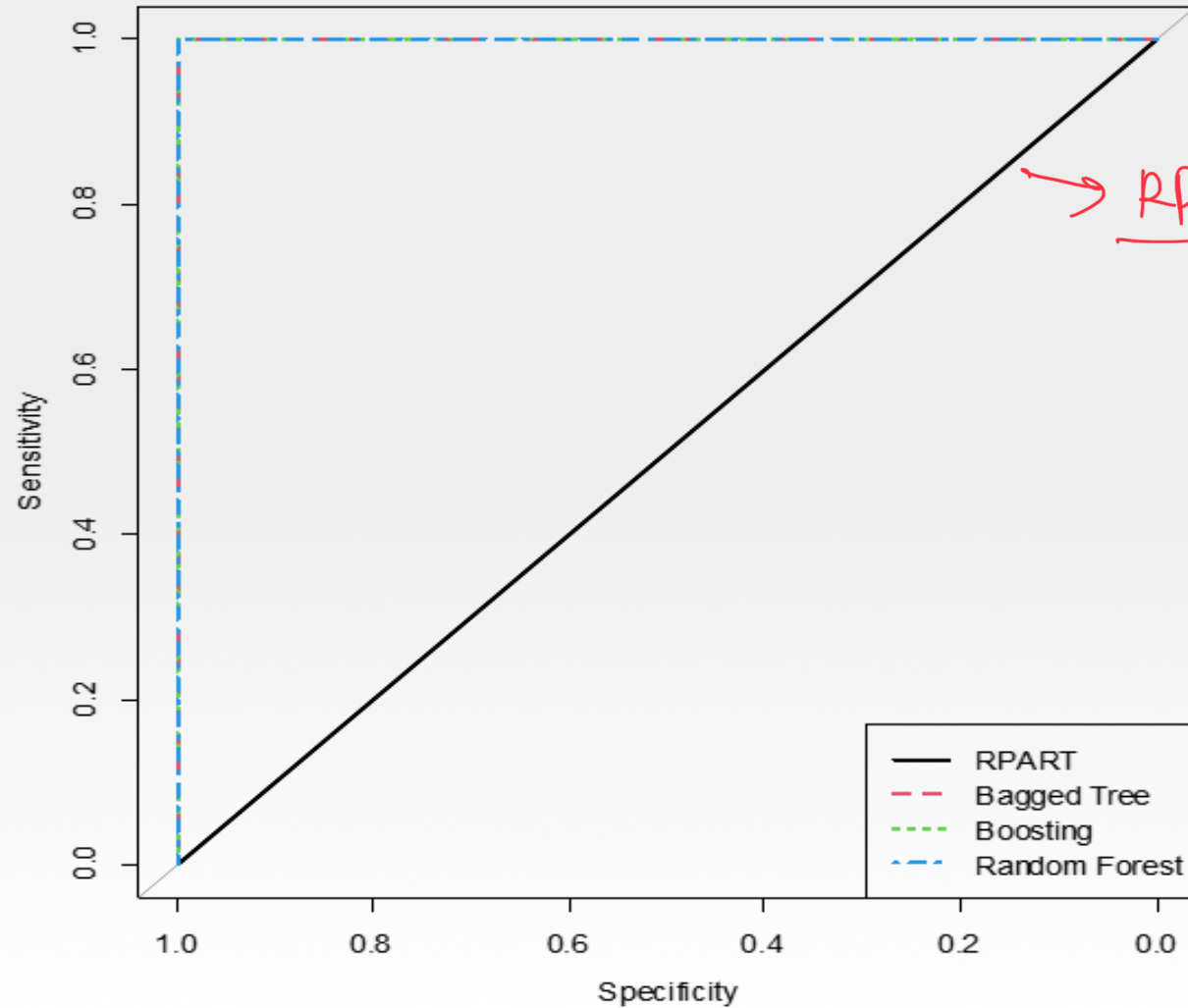
RFROC <- roc(Smarket.test\$Direction, Smarket.test\$RF)

lines(RFROC, col=4, lty=4, lwd=2)

legend('bottomright', c('RPART', 'Bagged Tree', 'Boosting', 'Random Forest'), col=1:4,

lty=1:4, lwd=2)

Roc curves of different models



→ RPART (does not work)

Confusion matrices of different models

#Confusion matrix of Classification Trees

confusionMatrix(data = predict(rpartTune, Smarket.test), reference = Smarket.test\$Direction)

response from the test data

#Confusion Matrix of Bagged Trees

confusionMatrix(data = predict(treebagTune, Smarket.test), reference = Smarket.test\$Direction)

#Confusion matrix of Boosting

confusionMatrix(data = predict(gbmTune, Smarket.test), reference = Smarket.test\$Direction)

#Confusion matrix of Random Forest

confusionMatrix(data = predict(rfTune, Smarket.test), reference = Smarket.test\$Direction)

Summary of output

this classification tree model does not work!

Model	Accuracy	Kappa	Sensitivity	Specificity
Classification Tree	0.5595	0.0000	0.0000	1.0000
Bagged Tree	1.0000	1.0000	1.0000	1.0000
Boosting	1.0000	1.0000	1.0000	1.0000
Random forest	1.0000	1.0000	1.0000	1.0000

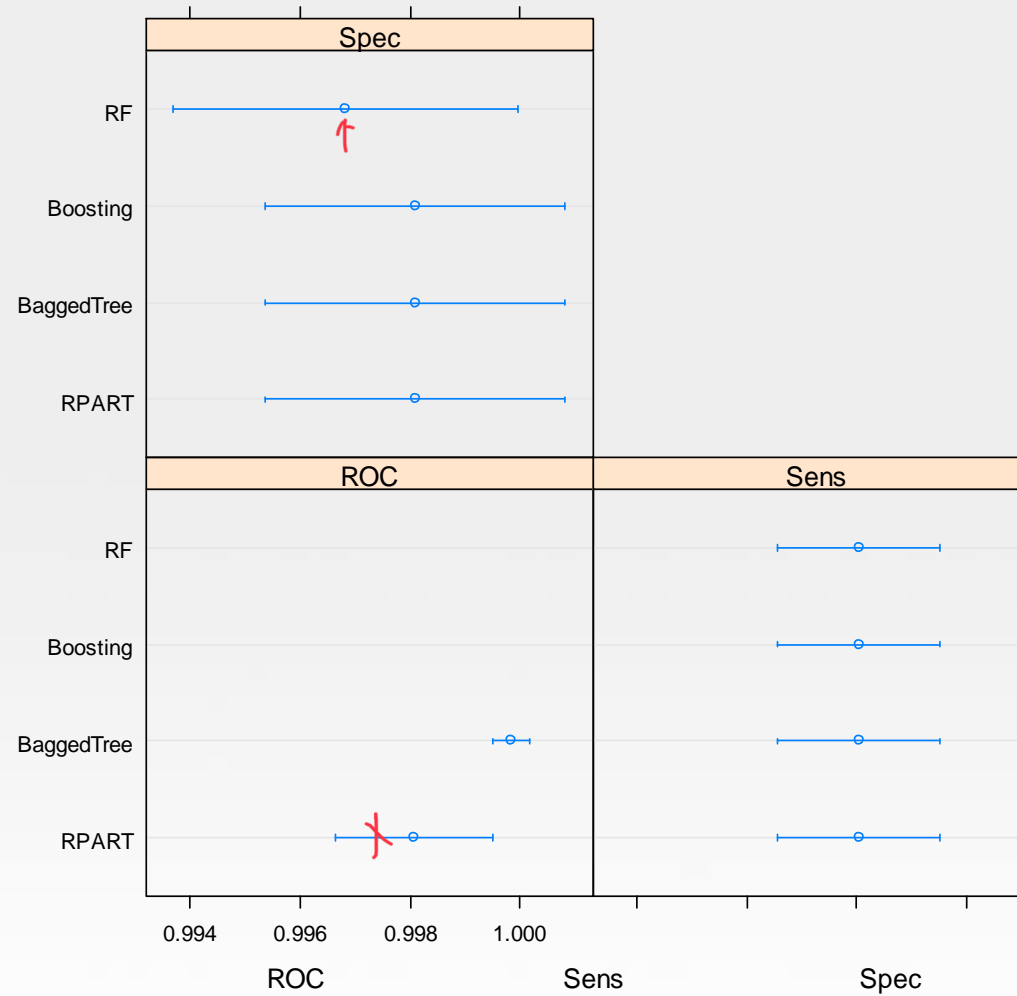
Model comparisons based on **resamples** of Training data

#Resamples of Training data

names that you defined.
res = resamples(list(RPART = rpartTune, BaggedTree = treebagTune,
Boosting = gbmTune, RF = rfTune))
dotplot(res)

#We can add the models from Chapters 12 and 13

res2 = resamples(list(Logistic = logisticTune, LDA = ldaTune, PLSDA = plsdaTune,
Penalized = glmnTune, NSC = nscTune, QDA = QDATune, RDA = RDATune, MDA =
MDATune, NB = NBTune, KNN = KNNTune,
NN = NNTune, FDA = FDATune, SVM = SVMTune, RPART = rpartTune, BaggedTree =
treebagTune,
Boosting = gbmTune, RF = rfTune))
dotplot(res2)



Confidence Level: 0.95

Model comparisons from Chapters 12, 13, and 14

