Chapter 8

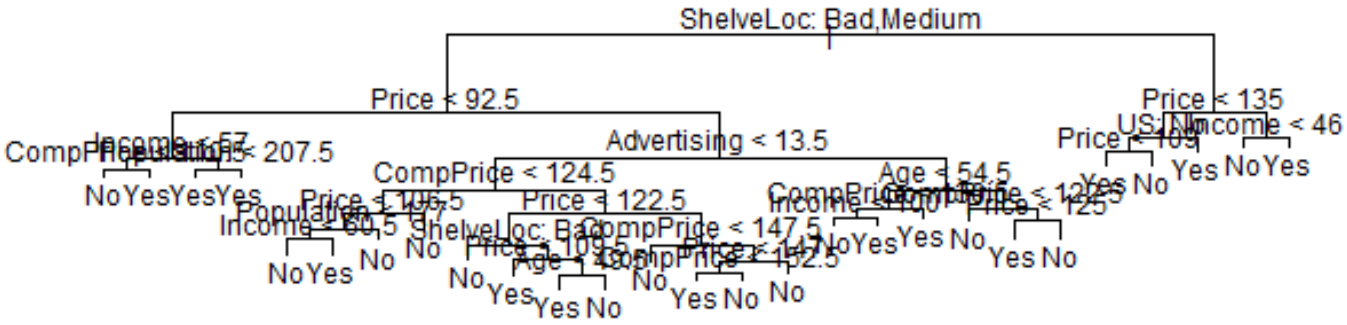
Lab

Fitting Classification Trees

```
carseats <- as.data.table(ISLR::Carseats)

carseats[, High := as.factor(ifelse(Sales <= 8, "No", "Yes"))]

tree.carseats <- tree(formula = High ~ .-Sales, data = carseats)

summary(tree.carseats)
```

```
Classification tree:
tree(formula = High ~ . - Sales, data = carseats)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
[6] "Advertising" "Age"         "US"
Number of terminal nodes:  27
Residual mean deviance:  0.4575 = 170.7 / 373
Misclassification error rate: 0.09 = 36 / 400
```

```
plot(tree.carseats)
text(tree.carseats, pretty = 0)
```




```
set.seed(2)
```

```
train <- sample(1:nrow(carseats), 200)

carseats.test <- carseats[-train]
high.test <- carseats[-train]$High

tree.carseats <- tree(High ~.-Sales, data = carseats, subset = train)
tree.pred <- predict(tree.carseats, carseats.test, type = "class")

table(tree.pred, high.test)
```

```
         high.test
tree.pred  No Yes
      No  104  33
      Yes  13  50
```

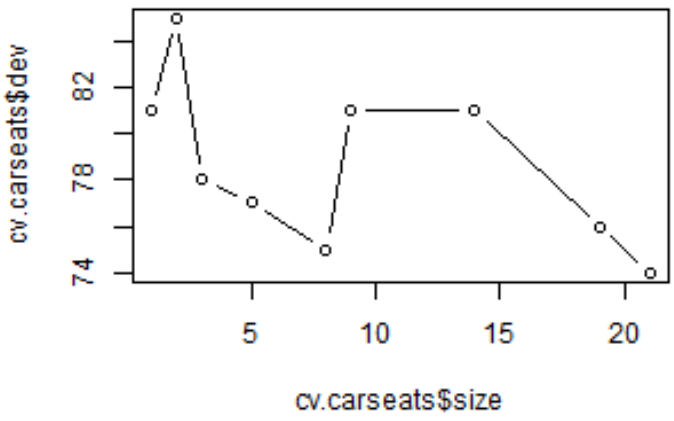
```
set.seed(3)

cv.carseats <- cv.tree(tree.carseats, FUN = prune.misclass)
names(cv.carseats)
```

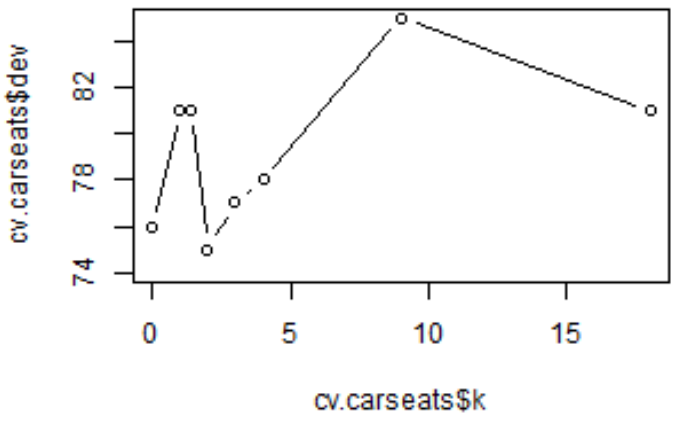
```
[1] "size"   "dev"    "k"      "method"
```

```
cv.carseats
```

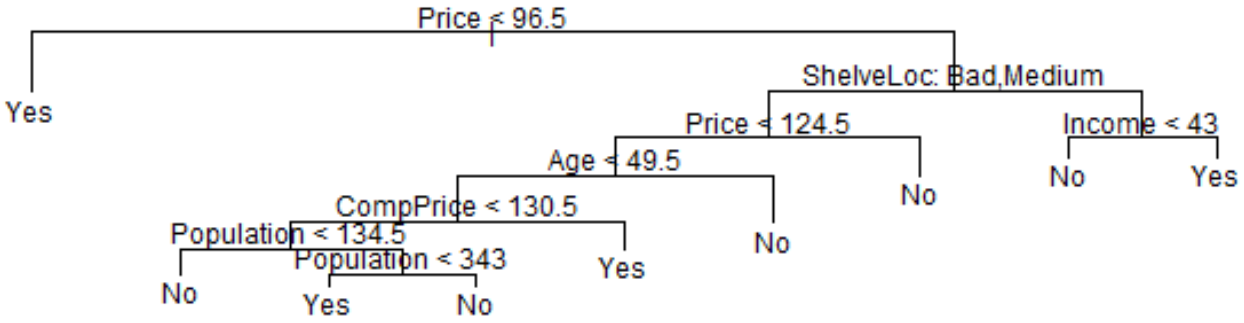
```
$size
[1] 21 19 14  9  8  5  3  2  1

$dev
[1] 74 76 81 81 75 77 78 85 81

$k
[1] -Inf  0.0  1.0  1.4  2.0  3.0  4.0  9.0 18.0

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```

```
par(mfrow = c(1, 2))
plot(cv.carseats$size, cv.carseats$dev, type = "b")
plot(cv.carseats$k, cv.carseats$dev, type="b")
```





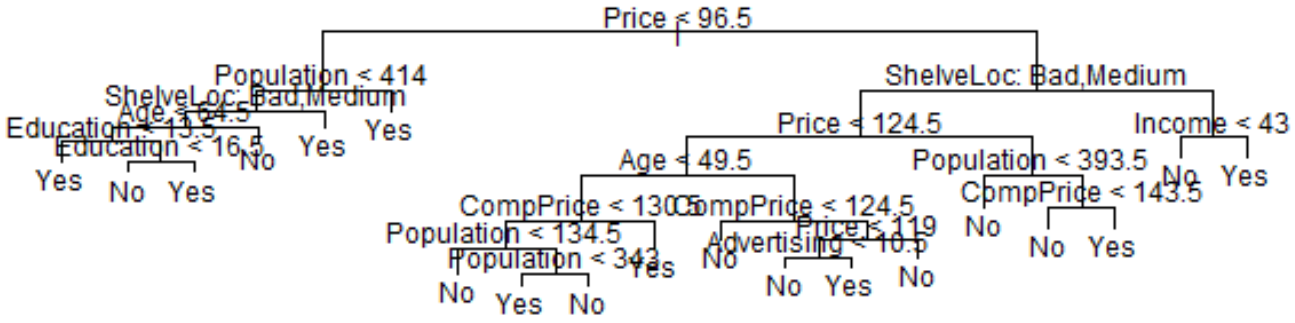

```
par(mfrow = c(1,1))
prune.carseats <- prune.misclass(tree.carseats, best = 9)
plot(prune.carseats)
text(prune.carseats, pretty = 0)
```




```
tree.pred <- predict(prune.carseats, carseats.test, type = "class")
table(tree.pred, high.test)
```

```
         high.test
tree.pred No Yes
      No  97  25
      Yes 20  58
```

```
prune.carseats <- prune.misclass(tree.carseats, best = 15)
plot(prune.carseats)
text(prune.carseats, pretty = 0)
```




```
tree.pred <- predict(prune.carseats, carseats.test, type = "class")
table(tree.pred, high.test)
```

```
         high.test
tree.pred  No Yes
      No  102  30
      Yes  15  53
```

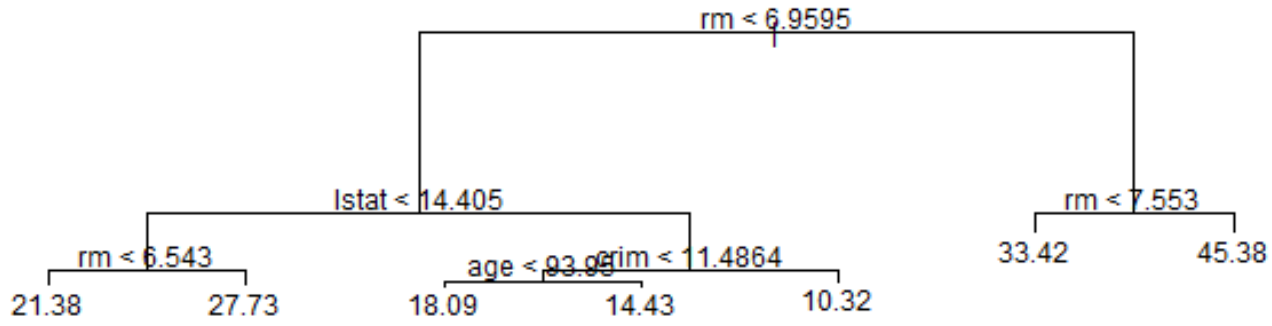
Regression Trees

```
boston <- as.data.table(Boston)

N <- nrow(boston)

set.seed(1)

train <- sample(1:N, N /2)

tree.boston <- tree(medv ~ ., boston, subset = train)

summary(tree.boston)
```

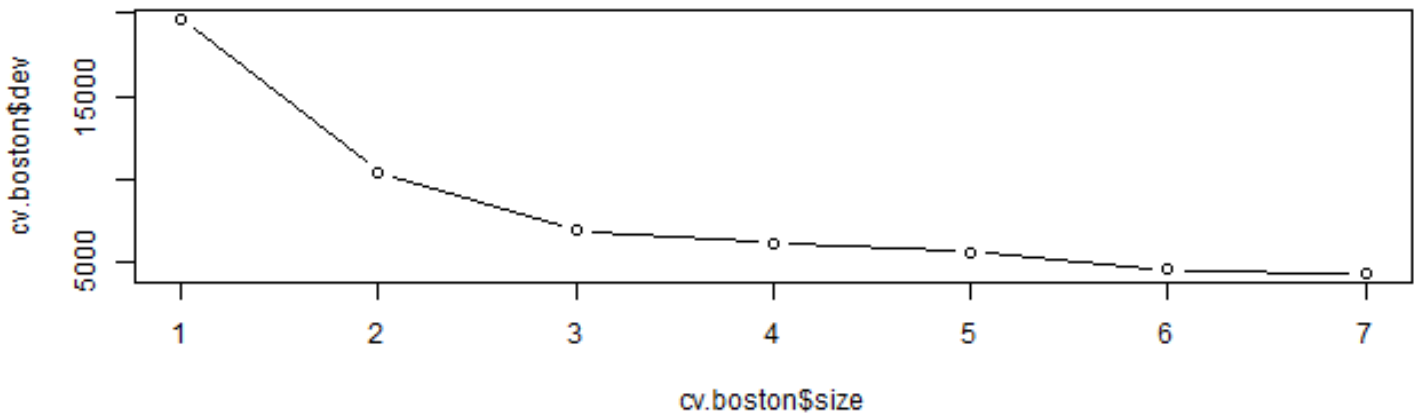
```
Regression tree:
tree(formula = medv ~ ., data = boston, subset = train)
```

```
Variables actually used in tree construction:
[1] "rm"    "lstat" "crim"  "age"
Number of terminal nodes:  7
Residual mean deviance:  10.38 = 2555 / 246
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-10.1800  -1.7770  -0.1775   0.0000   1.9230  16.5800
```

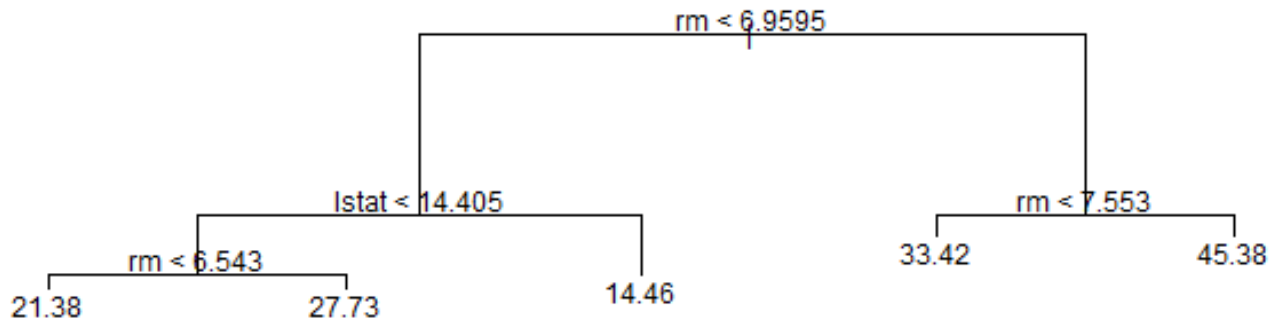
```
plot(tree.boston)
text(tree.boston, pretty = 0)
```



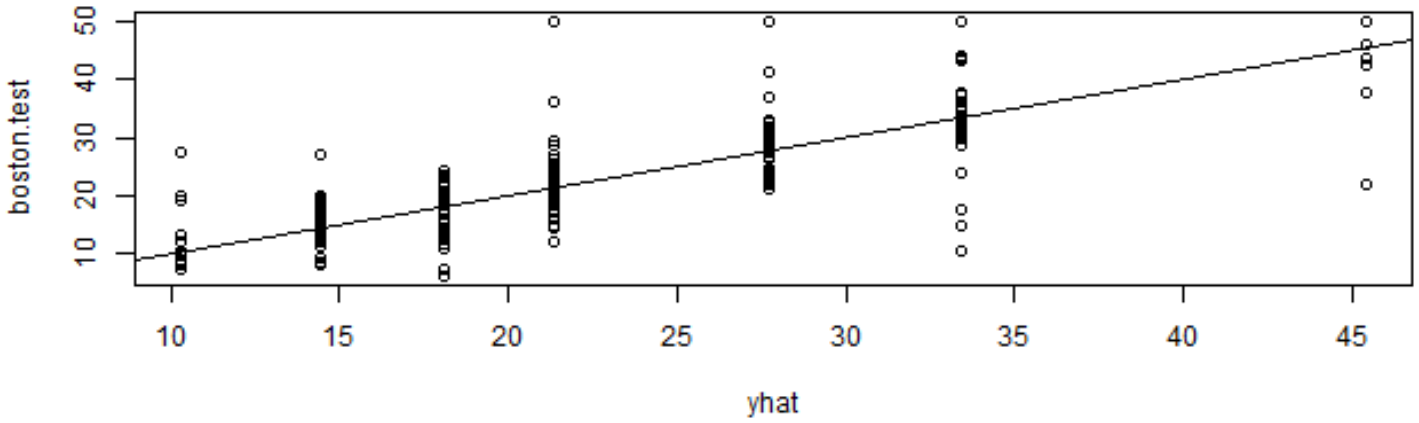

```
cv.boston <- cv.tree(tree.boston)
plot(cv.boston$size, cv.boston$dev, type = 'b')
```



```
prune.boston <- prune.tree(tree.boston, best = 5)
plot(prune.boston)
text(prune.boston, pretty = 0)
```




```
yhat <- predict(tree.boston, newdata = boston[-train,])
boston.test <- boston[-train]$medv

plot(yhat, boston.test)
abline(0, 1)
```




```
mean((yhat - boston.test)^2)
```

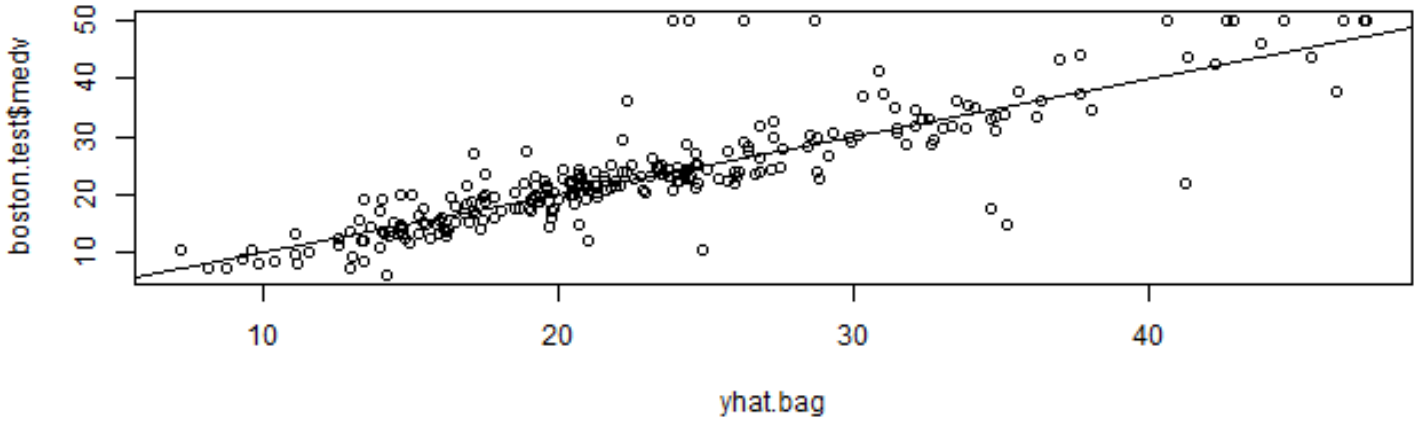
```
[1] 35.28688
```

Bagging and Boosting

```
set.seed(1)

bag.boston <- randomForest(medv ~ ., data = boston, subset = train, mtry = 13, importance = T)
bag.boston
```

```
Call:
 randomForest(formula = medv ~ ., data = boston, mtry = 13, importance = T,      subset = train
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 13

          Mean of squared residuals: 11.39601
                    % Var explained: 85.17
```

```
boston.test <- boston[!train]
yhat.bag <- predict(bag.boston, newdata = boston.test)
plot(yhat.bag, boston.test$medv)
abline(0, 1)
```




```
mean((yhat.bag-boston.test$medv)^2)
```

```
[1] 23.59273
```

```
bag.boston <- randomForest(medv ~ ., data = boston, subset = train, mtry = 13, ntree = 25)
yhat.bag <- predict(bag.boston, newdata = boston[-train,])
mean((yhat.bag - boston.test$medv)^2)
```

```
[1] 23.66716
```

```
set.seed(1)

rf.boston <- randomForest(medv ~  ., data = boston, subset = train, mtry = 6, importance = T)
yhat.rf <- predict(rf.boston, newdata = boston[-train,])
mean((yhat.rf-boston.test)^2)
```

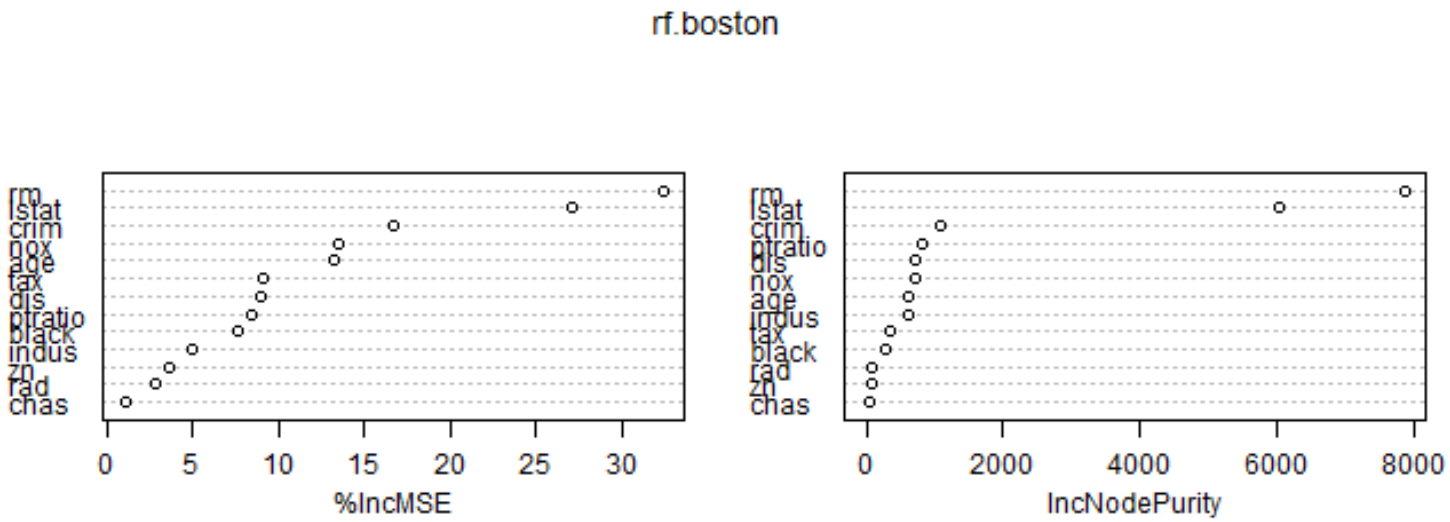
```
Warning in mean.default((yhat.rf - boston.test)^2): argument is not numeric or
logical: returning NA
```

```
[1] NA
```

```
importance(rf.boston)
```

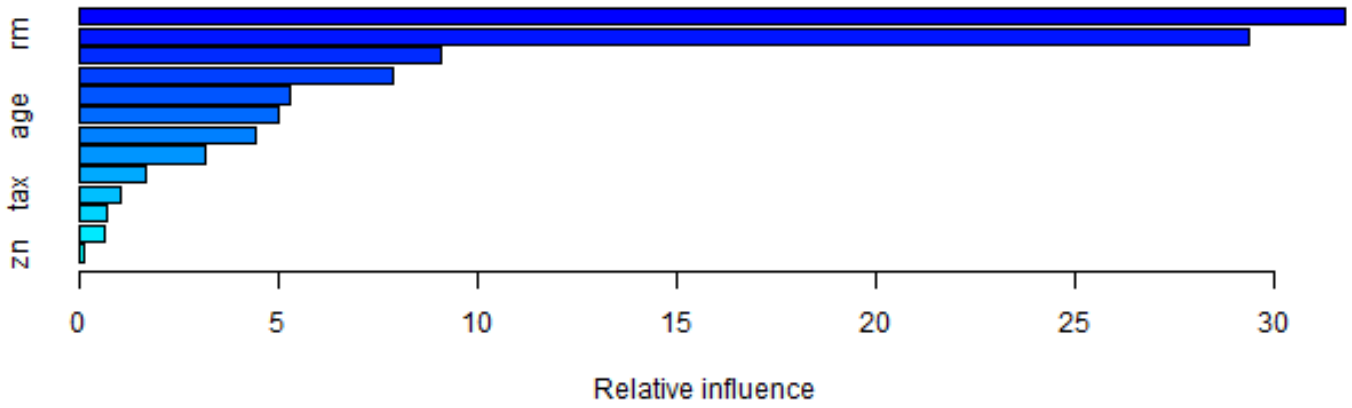
```
          %IncMSE IncNodePurity
crim    16.697017    1076.08786
zn       3.625784      88.35342
indus    4.968621     609.53356
chas     1.061432      52.21793
nox     13.518179     709.87339
rm      32.343305    7857.65451
age     13.272498     612.21424
dis      9.032477     714.94674
rad      2.878434      95.80598
tax      9.118801     364.92479
ptratio  8.467062     823.93341
black    7.579482     275.62272
lstat   27.129817    6027.63740
```

```
varImpPlot(rf.boston)
```



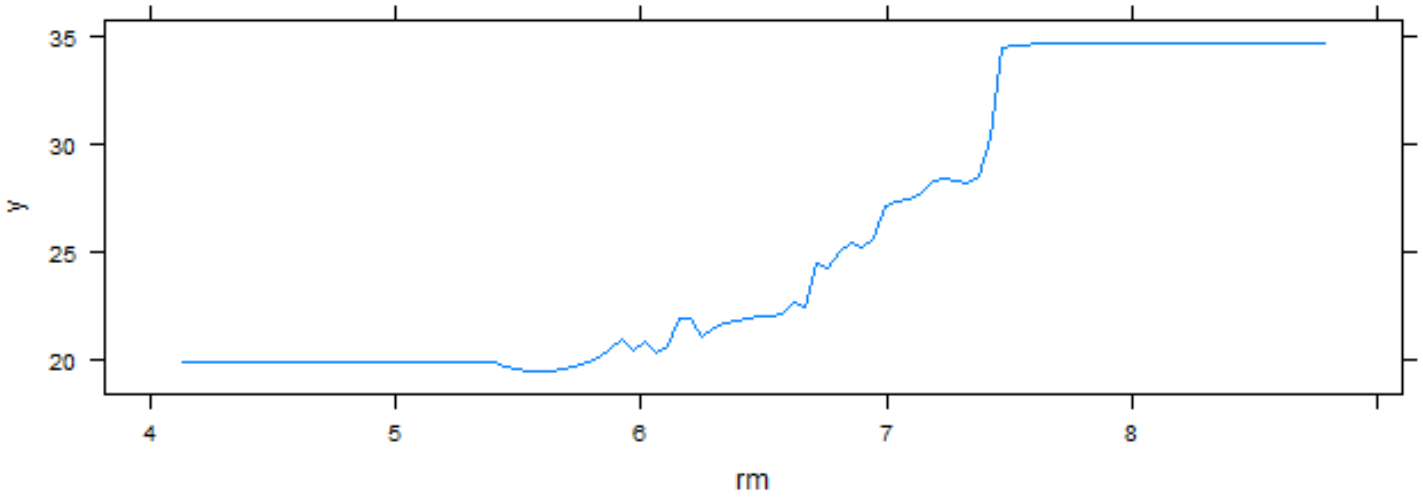
Boosting

```
boost.boston <- gbm(medv ~ ., data = boston[-train,], distribution = "gaussian", n.trees = 5000

summary(boost.boston)
```

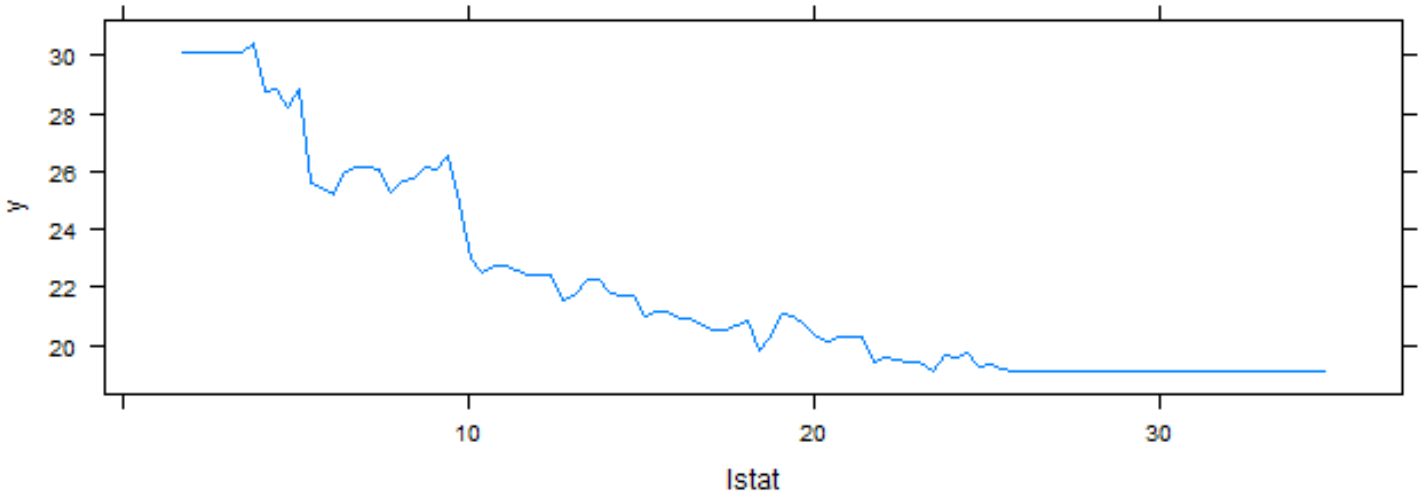



```
            var    rel.inf
lstat     lstat 31.7768815
rm           rm 29.3618406
dis         dis  9.0801247
crim       crim  7.8431133
nox         nox  5.3077691
age         age  4.9799119
black     black  4.4118851
ptratio ptratio  3.1614540
indus     indus  1.6642386
tax         tax  1.0107120
chas       chas  0.6850917
rad         rad  0.6031076
zn           zn  0.1138700
```

```
par(mfrow = c(1, 2))
plot(boost.boston, i = "rm")
```




```
plot(boost.boston, i = "lstat")
```




```
yhat.boost <- predict(boost.boston, newdata = boston[-train,], n.trees = 5000)
mean((yhat.boost - boston.test)^2)
```

```
Warning in mean.default((yhat.boost - boston.test)^2): argument is not numeric
or logical: returning NA
```

```
[1] NA
```

Applied

In the lab, we applied random forests to the *Boston* data using mtry = 6 and using ntree = 25 and ntree = 500. Create a plot displaying the test error resulting from random forests on this data set for a more

comprehensive range of values for mtry ntree.

```
N <- nrow(boston)

train <- sample(1:N, N * .7)

boston.train <- boston[train]
boston.test <- boston[!train]

ntrees <- seq(0, 500, 10)
n.features <- ncol(boston[, !"medv"])

mtry <- round(c(n.features, n.features/2, sqrt(n.features)))

results <- matrix(vector(mode = "numeric", length(ntrees) * length(mtry)), nrow = 3)

for(i in 1:length(ntrees))
{
   for(j in 1:length(mtry))
   {
      tree.count <- ntrees[i]

      rf <- randomForest(medv ~ ., data = boston.train, mtry = mtry[j], n.trees = tree.count)

      pred <- predict(rf, newdata = boston.test)

      results[j, i] <- sqrt(mean((pred - boston.test$medv)^2)) # store the rmse
   }
}

df_results <- as.data.table(results)
colnames(df_results) <- paste0(ntrees)

df_results <- melt(cbind(mtry, df_results), id.vars = "mtry")
df_results$mtry <- as.factor(df_results$mtry)

ggplot(df_results, aes(variable, value, group = mtry)) +
   geom_line(aes(col = mtry))
```

Carseats

In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitaive variable.

a.) Split the data into a t training set and a test set.

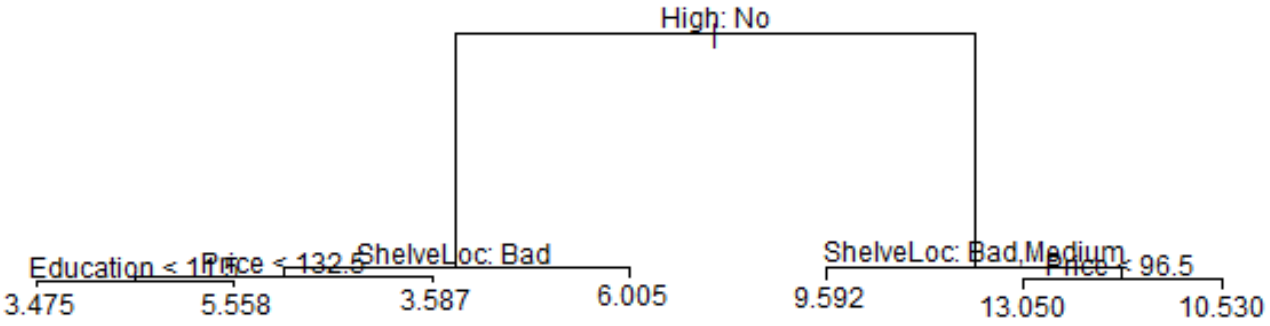
```
N <- nrow(carseats)

index <- sample(1:N, N * .7)

carseats.train <- carseats[index]
carseats.test <- carseats[!index]
```

b.) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

```
carseat.tree <- tree(Sales ~., data = carseats.train)
pred <- predict(carseat.tree, newdata = carseats.test)
mse <- mean((pred - carseats.test$Sales)^2)

mse
```

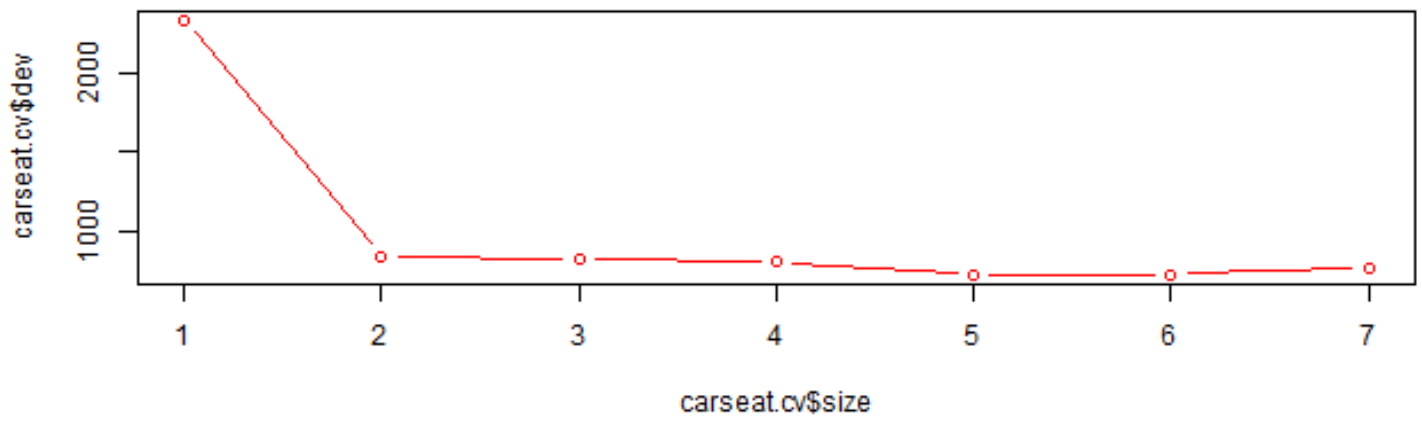
```
[1] 2.399014
```

```
par(mfrow = c(1,1))
plot(carseat.tree)
text(carseat.tree, pretty = 0)
```

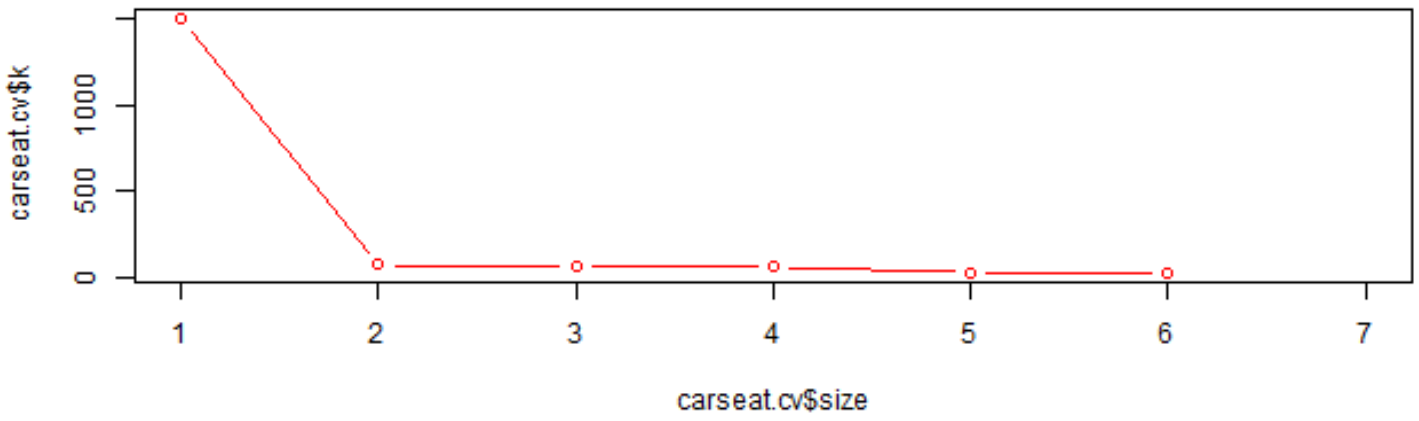


c.) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

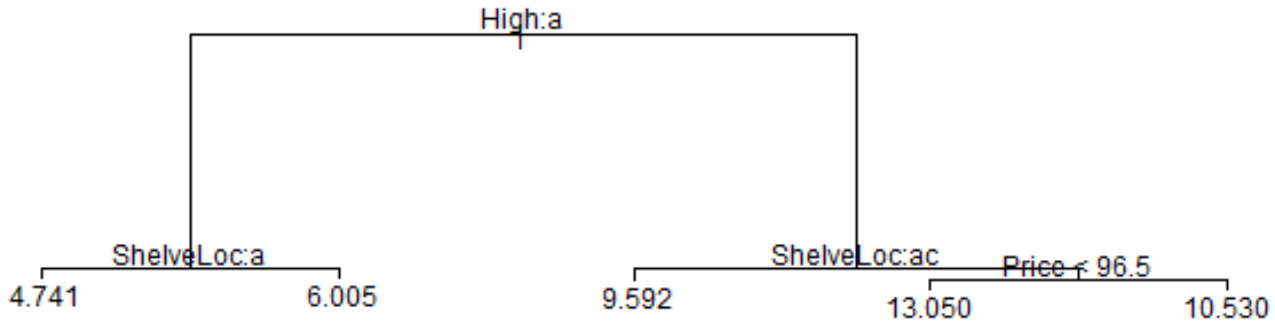
```
carseat.cv <- cv.tree(carseat.tree)

plot(carseat.cv$size, carseat.cv$dev, type = "b", col = "red")
```




```
plot(carseat.cv$size, carseat.cv$k, type = "b", col = "red")
```




```
prune.carseats <- prune.tree(carseat.tree, best = 5)

plot(prune.carseats)
text(prune.carseats)
```



d.) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

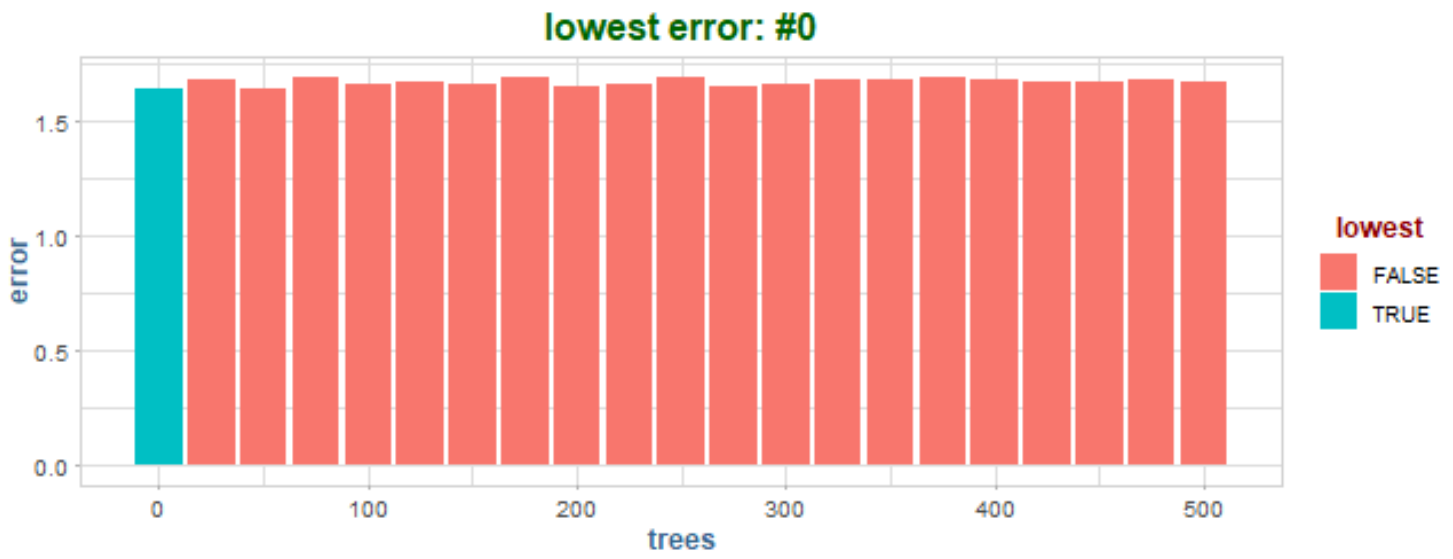
```
p <- ncol(carseats)

ntrees <- seq(0, 500, 25)
rf.error <- numeric(length(ntrees))

for(i in 1:length(ntrees))
```

```
{
   carseats.rf <- randomForest(Sales ~ ., data = carseats.train, mtry = p, n.trees = ntrees[i])

   pred <- predict(carseats.rf, newdata = carseats.test)

   rf.error[i] <- mean((pred - carseats.test$Sales)^2)
}
```

```
Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
```

```
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range

Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range
```

```r
lowest.error <- which.min(rf.error)

results <- data.table(trees = ntrees, error = rf.error)[, lowest := .I == lowest.error]

ggplot(results, aes(trees, error, fill = lowest)) +
   geom_bar(stat = "identity") +
   labs(title = paste0("lowest error: #", ntrees[lowest.error]))
```

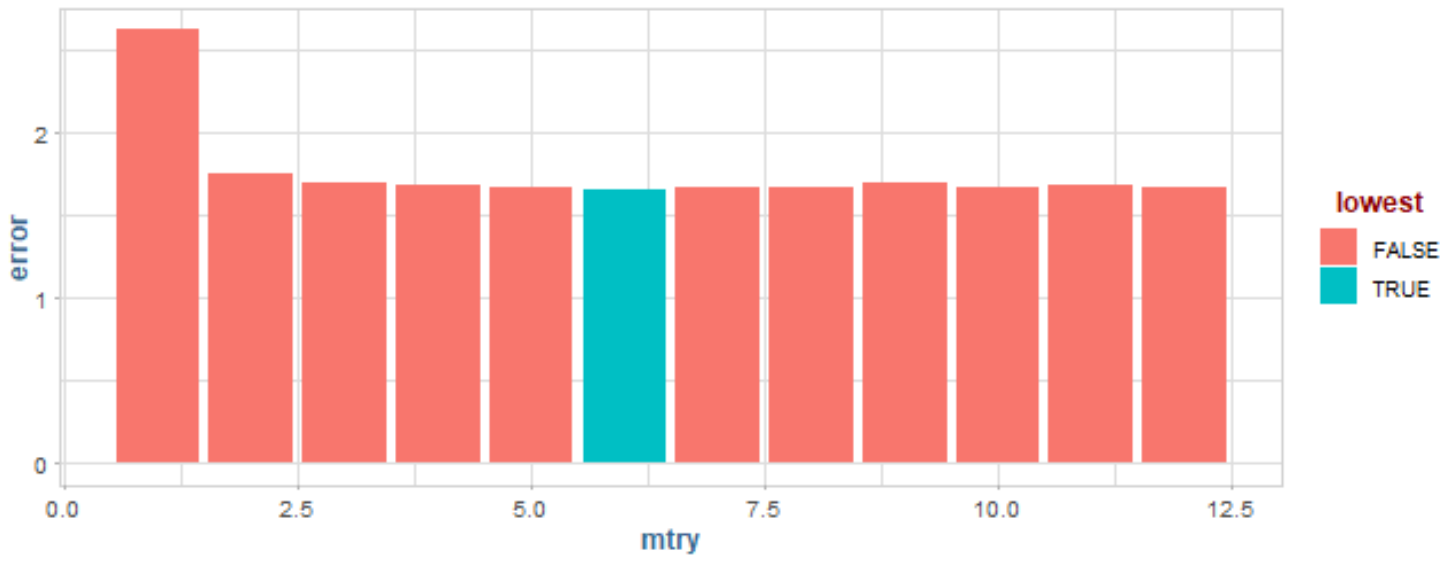


e.) Use random forest to analyze this data. What test MSE do you obtain? Use the importance() fucntion to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error obtained.

```
p <- ncol(carseats)

mtry <- seq(1, p)

rf.error <- numeric(p)

for(i in 1:p)
{
   carseats.rf <- randomForest(Sales ~ ., data = carseats.train, mtry = mtry[i], n.trees = 250)

   pred <- predict(carseats.rf, newdata = carseats.test)

   rf.error[i] <- mean((pred - carseats.test$Sales)^2)
}
```

```
Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
range
```

```
lowest.error <- which.min(rf.error)

results <- data.table(mtry = mtry, error = rf.error, lowest = mtry == lowest.error)

ggplot(results, aes(mtry, error, fill = lowest)) +
   geom_bar(stat = "identity")
```



OJ

This problem involves the **OJ** data set which is part of the ISLR package.

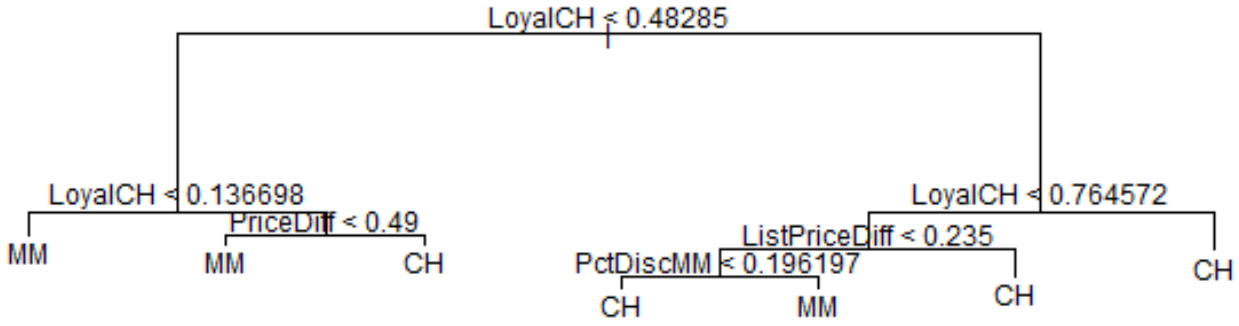
a.) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

```
oj <- as.data.table(ISLR::OJ)
n <- nrow(oj)

index <- sample(1:n, n *.7, replace = F)

oj.train <- oj[index]
oj.test <- oj[!index]

head(train)
```

```
[1]  45 110 113  84 104 485
```

b.) Fit a tree to the training data, which **Purchases** as the first response and the other variables as predictors. Use the **summary** function to produce summary statistics about the tree, and describe the results obtained.

```
oj.tree <- tree(Purchase ~., data = oj.train)

plot(oj.tree)
text(oj.tree)
```




```
summary(oj.tree)
```

```
Classification tree:
tree(formula = Purchase ~ ., data = oj.train)
Variables actually used in tree construction:
[1] "LoyalCH"       "PriceDiff"     "ListPriceDiff" "PctDiscMM"
Number of terminal nodes:  7
Residual mean deviance:  0.7672 = 569.3 / 742
Misclassification error rate: 0.1656 = 124 / 749
```

What is the training error rate? *0.7469*

How many terminal nodes? *8*

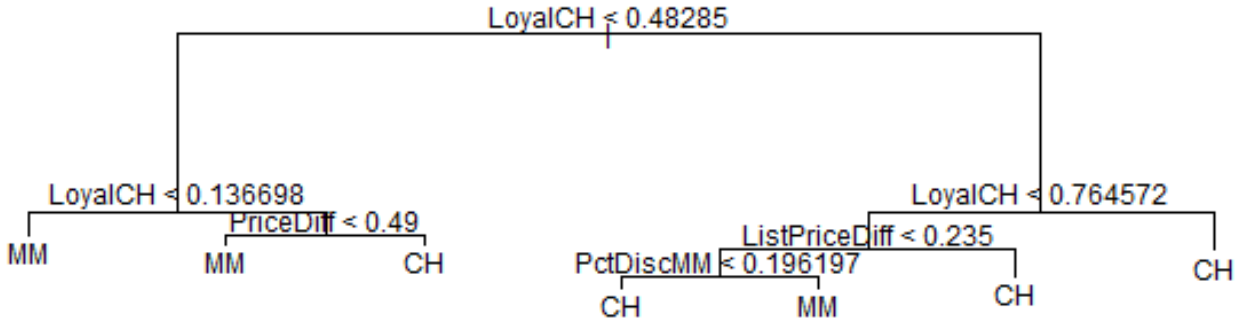
c.) Type in the name of the tree object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

```
summary(oj.tree)
```

```
Classification tree:
tree(formula = Purchase ~ ., data = oj.train)
Variables actually used in tree construction:
[1] "LoyalCH"       "PriceDiff"     "ListPriceDiff" "PctDiscMM"
Number of terminal nodes:  7
Residual mean deviance:  0.7672 = 569.3 / 742
Misclassification error rate: 0.1656 = 124 / 749
```

d.) Create a plot of the tree.

```
plot(oj.tree)
text(oj.tree)
```



e.) Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels.

```
pred <- predict(oj.tree, newdata = oj.test, type = "class")

confusionMatrix(oj.test$Purchase, pred)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  CH  MM
        CH 174  26
        MM  29  92

               Accuracy : 0.8287
                 95% CI : (0.7829, 0.8682)
    No Information Rate : 0.6324
    P-Value [Acc > NIR] : 9.271e-15

                  Kappa : 0.6334

 Mcnemar's Test P-Value : 0.7874

            Sensitivity : 0.8571
            Specificity : 0.7797
         Pos Pred Value : 0.8700
         Neg Pred Value : 0.7603
```

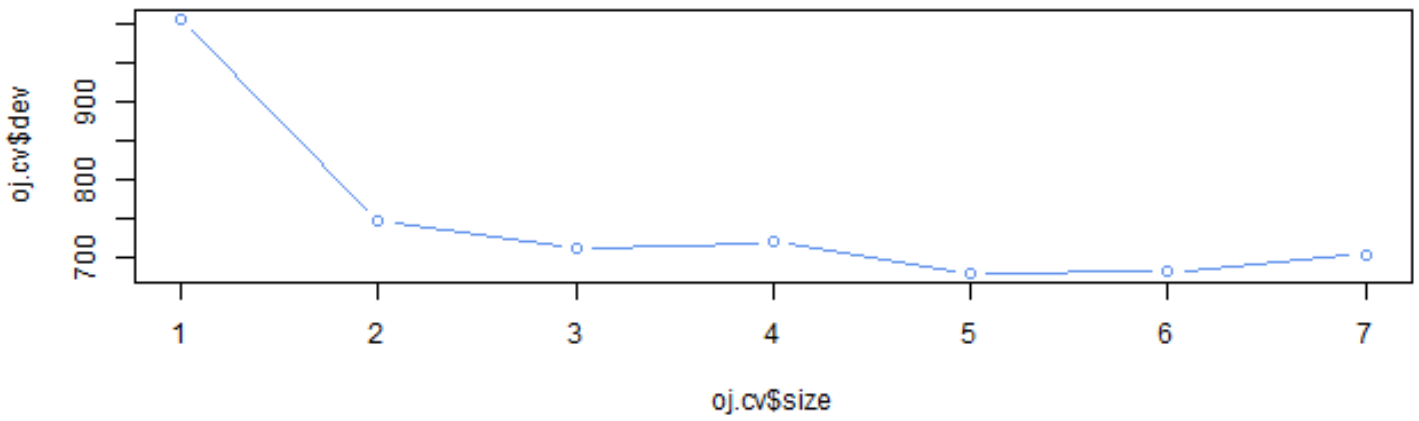
```
            Prevalence : 0.6324
        Detection Rate : 0.5421
  Detection Prevalence : 0.6231
     Balanced Accuracy : 0.8184

      'Positive' Class : CH
```

f.) Apply the cv.tree function to the training data set in order to determine the optimal tree size.

```
oj.cv <- cv.tree(oj.tree)
```

g.) Produce a plot with the tree size and cross-validated error rate.

```
plot(oj.cv$size, oj.cv$dev, type ='b', col =  "cornflowerblue") # 5
```



h.) What tree size corresponds to the lowest cv error rate?

5

i.) Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation.

```
oj.pruned <- prune.tree(oj.tree, k = 5)
```

j.) Compare the training error rates between the pruned and unpruned trees.

```
summary(oj.tree)
```

```
Classification tree:
tree(formula = Purchase ~ ., data = oj.train)
Variables actually used in tree construction:
[1] "LoyalCH"       "PriceDiff"     "ListPriceDiff" "PctDiscMM"
Number of terminal nodes:  7
```

```
Residual mean deviance:  0.7672 = 569.3 / 742
Misclassification error rate: 0.1656 = 124 / 749
```

```
summary(oj.pruned)
```

```
Classification tree:
tree(formula = Purchase ~ ., data = oj.train)
Variables actually used in tree construction:
[1] "LoyalCH"       "PriceDiff"     "ListPriceDiff" "PctDiscMM"
Number of terminal nodes:  7
Residual mean deviance:  0.7672 = 569.3 / 742
Misclassification error rate: 0.1656 = 124 / 749
```

Same.

k.) Compare the test error rates.

```
pruned.pred <- predict(oj.pruned, newdata = oj.test, type = "class")

confusionMatrix(oj.test$Purchase, pred)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  CH  MM
        CH 174  26
        MM  29  92

               Accuracy : 0.8287
                 95% CI : (0.7829, 0.8682)
    No Information Rate : 0.6324
    P-Value [Acc > NIR] : 9.271e-15

                  Kappa : 0.6334

 Mcnemar's Test P-Value : 0.7874

            Sensitivity : 0.8571
            Specificity : 0.7797
         Pos Pred Value : 0.8700
         Neg Pred Value : 0.7603
             Prevalence : 0.6324
         Detection Rate : 0.5421
   Detection Prevalence : 0.6231
      Balanced Accuracy : 0.8184
```

```
       'Positive' Class : CH
```

```
confusionMatrix(oj.test$Purchase, pruned.pred)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  CH  MM
        CH 174  26
        MM  29  92

               Accuracy : 0.8287
                 95% CI : (0.7829, 0.8682)
    No Information Rate : 0.6324
    P-Value [Acc > NIR] : 9.271e-15

                  Kappa : 0.6334

 Mcnemar's Test P-Value : 0.7874

            Sensitivity : 0.8571
            Specificity : 0.7797
         Pos Pred Value : 0.8700
         Neg Pred Value : 0.7603
             Prevalence : 0.6324
         Detection Rate : 0.5421
   Detection Prevalence : 0.6231
      Balanced Accuracy : 0.8184

       'Positive' Class : CH
```

Hitters

We now use boosting to predict **Salary** in the hitters data set.

a.) Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

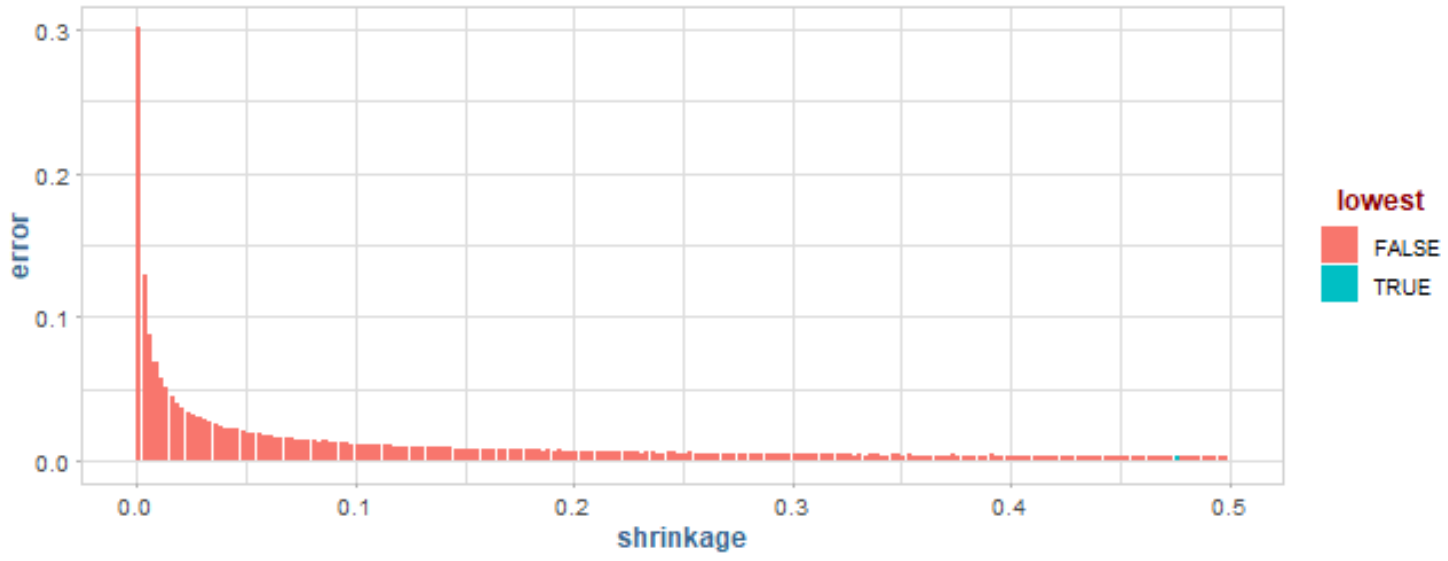
```
hitters <- as.data.table(ISLR::Hitters)
hitters <- hitters[complete.cases(hitters)]
hitters$Salary <- log(hitters$Salary)
```

b.) Create a training set consisting of the first 200 observations and a test set containing the rest.

```
n <- nrow(hitters)

index <- 1:200

hitters.train <- hitters[index]
hitters.test <- hitters[!index]
```

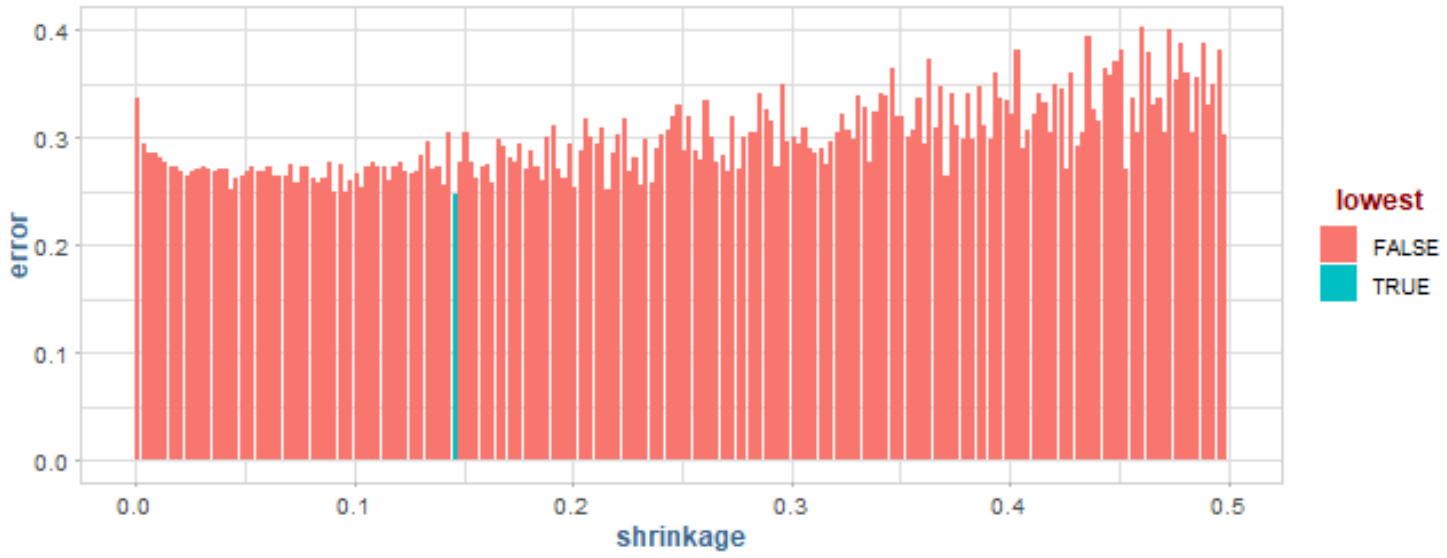
c.) Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with the different shrinkage values and the training MSE.

```
shrinkage <- seq(0.001, 0.5, by = 0.0025)
iters <- length(shrinkage)

results <- vector(mode = "numeric", length = iters)

for(i in 1:iters)
{
   gbm <- gbm(Salary ~., data = hitters.train, n.trees = 1000, shrinkage = shrinkage[i], distri
   results[i] <- mean(gbm$train.error^2)
}

lowest.error <- which.min(results)

df_results <- data.table(shrinkage, error = results)[, lowest := .I == lowest.error]

ggplot(df_results, aes(shrinkage, error, fill = lowest)) +
   geom_bar(stat = "identity")
```



d.) Produce a plot with different shrinkage methods and the corresponding test mse.

```
shrinkage <- seq(0.001, 0.5, by = 0.0025)
iters <- length(shrinkage)

results <- vector(mode = "numeric", length = iters)

for(i in 1:iters)
{
   gbm <- gbm(Salary ~., data = hitters.train, n.trees = 1000, shrinkage = shrinkage[i], distri

   pred <- predict(gbm, n.trees = 1000, newdata = hitters.test, type = "response")

   results[i] <- mean((hitters.test$Salary - pred)^2)
}

lowest.error <- which.min(results)

df_results <- data.table(shrinkage, error = results)[, lowest := .I == lowest.error]

ggplot(df_results, aes(shrinkage, error, fill = lowest)) +
   geom_bar(stat = "identity")
```

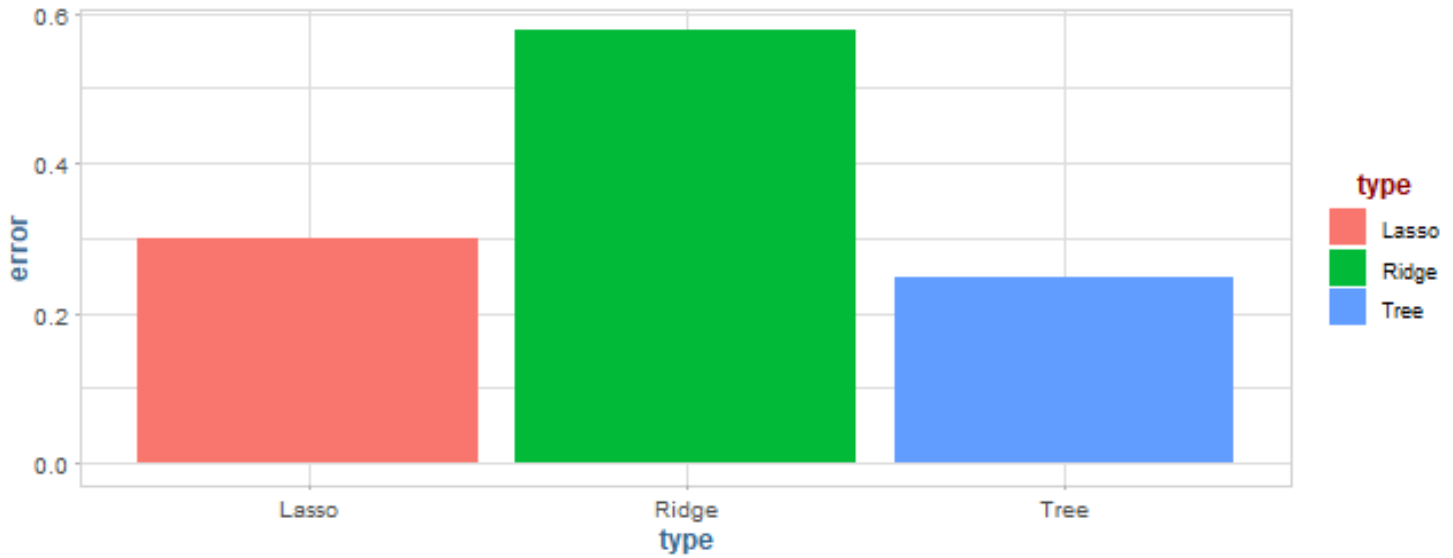



```
tree.mse <- results[lowest.error]
```

e.) Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in chapters 3 & 6.

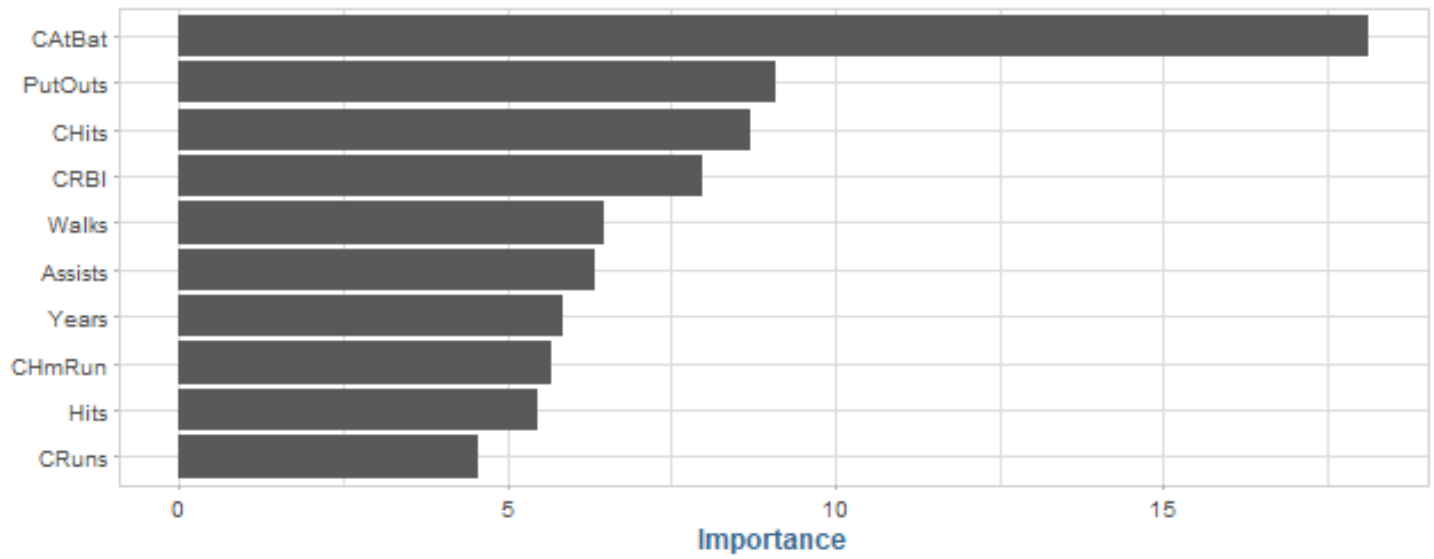
```
train.mat <- model.matrix(Salary ~., data = hitters.train)
test.mat <- model.matrix(Salary ~., data = hitters.test)

cv.ridge <- cv.glmnet(train.mat, hitters.train$Salary, data = hitters.train, alpha = 0)
```

```
ridge.fit <- glmnet(test.mat, hitters.test$Salary, alpha = 1, lambda = cv.ridge$lambda.min)
ridge.pred <- predict(ridge.fit, newx = test.mat, type = "response")
ridge.error <- mean((hitters.test$Salary - ridge.pred)^2)

cv.lasso <- cv.glmnet(train.mat, hitters.train$Salary, data = hitters.train, alpha = 1)

lasso.fit <- glmnet(test.mat, hitters.test$Salary, alpha = 0, lambda = cv.lasso$lambda.min)
lasso.pred <- predict(lasso.fit, newx = test.mat, type = "response")
lasso.error <- mean((hitters.test$Salary - lasso.pred)^2)

results <- data.table(type = c("Tree", "Ridge", "Lasso"), error = c(tree.mse, ridge.error, lass

ggplot(results, aes(type, error, fill = type)) +
   geom_bar(stat = "identity")
```



f.) Which variables appear to be the most important predictors in the boosted model?

```
best.tree <- gbm(Salary ~., data = hitters.train, n.trees = 1000, shrinkage = shrinkage[lowest.

vip::vip(best.tree)
```



Caravan

This question uses the caravan data set.

a.) Create a training set consisting of the first 1,000 observations, and a test set consisting of the remaining observations.

```
caravan <- as.data.table(ISLR::Caravan)
caravan$Purchase <- ifelse(caravan$Purchase == "Yes", 1, 0)
index <- 1:1000

caravan.train <- caravan[index]
caravan.test <- caravan[-index]

head(caravan.train)
```

```
   MOSTYPE MAANTHUI MGEMOMV MGEMLEEF MOSHOOFD MGODRK MGODPR MGODOV MGODGE
1:      33        1       3        2        8      0      5      1      3
2:      37        1       2        2        8      1      4      1      4
3:      37        1       2        2        8      0      4      2      4
4:       9        1       3        3        3      2      3      2      4
5:      40        1       4        2       10      1      4      1      4
6:      23        1       2        1        5      0      5      0      5
   MRELGE MRELSA MRELOV MFALLEEN MFGEKIND MFWEKIND MOPLHOOG MOPLMIDD MOPLLAAG
1:      7      0      2        1        2        6        1        2        7
2:      6      2      2        0        4        5        0        5        4
3:      3      2      4        4        4        2        0        5        4
4:      5      2      2        2        3        4        3        4        2
5:      7      1      2        2        4        4        5        4        0
6:      0      6      3        3        5        2        0        5        4
```

```
   MBERHOOG MBERZELF MBERBOER MBERMIDD MBERARBG MBERARBO MSKA MSKB1 MSKB2 MSKC
1:        1        0        1        2        5        2    1     1     2    6
2:        0        0        0        5        0        4    0     2     3    5
3:        0        0        0        7        0        2    0     5     0    4
4:        4        0        0        3        1        2    3     2     1    4
5:        0        5        4        0        0        0    9     0     0    0
6:        2        0        0        4        2        2    2     2     2    4
   MSKD MHHUUR MHKOOP MAUT1 MAUT2 MAUT0 MZFONDS MZPART MINKM30 MINK3045
1:    1      1      8     8     0     1        8      1        0        4
2:    0      2      7     7     1     2        6      3        2        0
3:    0      7      2     7     0     2        9      0        4        5
4:    0      5      4     9     0     0        7      2        1        5
5:    0      4      5     6     2     1        5      4        0        0
6:    2      9      0     5     3     3        9      0        5        2
   MINK4575 MINK7512 MINK123M MINKGEM MKOOPKLA PWAPART PWABEDR PWALAND PPERSAUT
1:        5        0        0       4        3       0       0       0        6
2:        5        2        0       5        4       2       0       0        0
3:        0        0        0       3        4       2       0       0        6
4:        3        0        0       4        4       0       0       0        6
5:        9        0        0       6        3       0       0       0        0
6:        3        0        0       3        3       0       0       0        6
   PBESAUT PMOTSCO PVRAAUT PAANHANG PTRACTOR PWERKT PBROM PLEVEN PPERSONG
1:       0       0       0        0        0      0     0      0        0
2:       0       0       0        0        0      0     0      0        0
3:       0       0       0        0        0      0     0      0        0
4:       0       0       0        0        0      0     0      0        0
5:       0       0       0        0        0      0     0      0        0
6:       0       0       0        0        0      0     0      0        0
   PGEZONG PWAOREG PBRAND PZEILPL PPLEZIER PFIETS PINBOED PBYSTAND AWAPART
1:       0       0      5       0        0      0       0        0       0
2:       0       0      2       0        0      0       0        0       2
3:       0       0      2       0        0      0       0        0       1
4:       0       0      2       0        0      0       0        0       0
5:       0       0      6       0        0      0       0        0       0
6:       0       0      0       0        0      0       0        0       0
   AWABEDR AWALAND APERSAUT ABESAUT AMOTSCO AVRAAUT AAANHANG ATRACTOR AWERKT
1:       0       0        1       0       0       0        0        0      0
2:       0       0        0       0       0       0        0        0      0
3:       0       0        1       0       0       0        0        0      0
4:       0       0        1       0       0       0        0        0      0
5:       0       0        0       0       0       0        0        0      0
6:       0       0        1       0       0       0        0        0      0
   ABROM ALEVEN APERSONG AGEZONG AWAOREG ABRAND AZEILPL APLEZIER AFIETS AINBOED
1:     0      0        0       0       0      1       0        0      0       0
2:     0      0        0       0       0      1       0        0      0       0
```

```
3:     0      0        0       0       0       1       0       0       0       0
4:     0      0        0       0       0       1       0       0       0       0
5:     0      0        0       0       0       1       0       0       0       0
6:     0      0        0       0       0       0       0       0       0       0
   ABYSTAND Purchase
1:        0        0
2:        0        0
3:        0        0
4:        0        0
5:        0        0
6:        0        0
```

b.) Fit a boosting model to the training set with **Purchases** as the response and the other variables as predictors. Use 1,000 trees and a shrinkage value of .01.

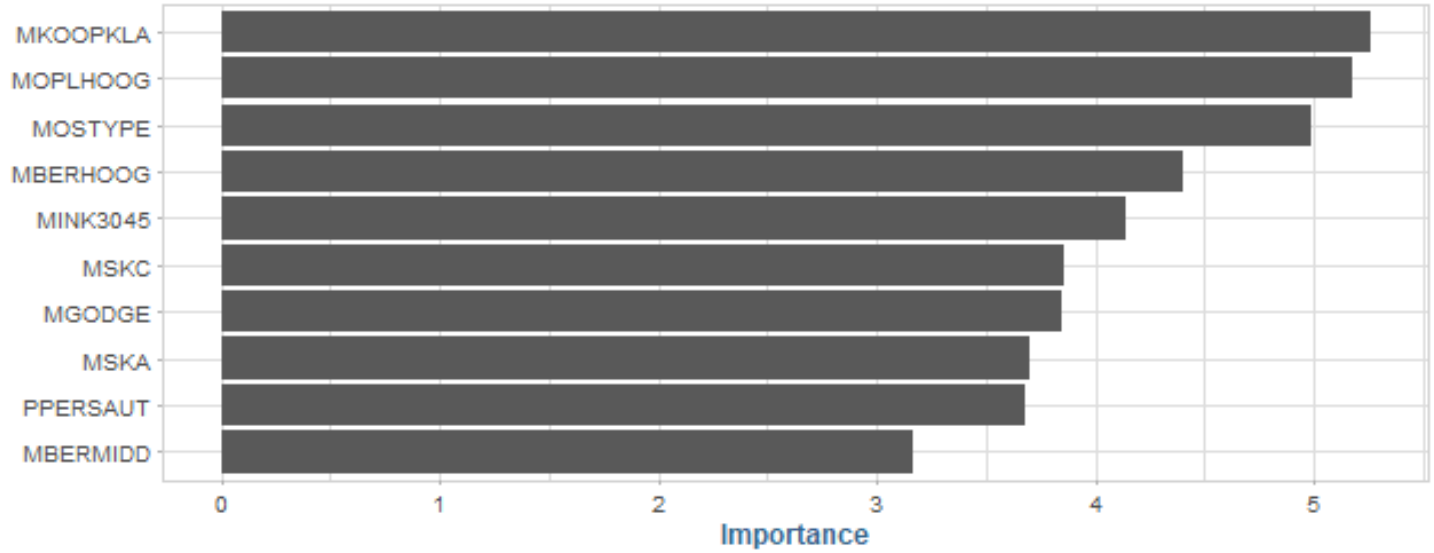
```
boost.fit <- gbm(Purchase ~ ., data = caravan.train, n.trees = 1000, shrinkage = 0.1, distribut
```

```
Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
variable 50: PVRAAUT has no variation.
```

```
Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
variable 71: AVRAAUT has no variation.
```

What predictors appear to be most important?

```
vip::vip(boost.fit)
```



c.) Use the boosting model to predict the response on the test data. Predict that a person will make a purchase if the estimated probablity of purchase is greater than 20%.

```
probs <- exp(predict.gbm(boost.fit, newdata = caravan.test, type = "response", n.trees = 1000))

pred <- ifelse(probs > .2, 1, 0)
```

```
confusionMatrix(as.factor(caravan.test$Purchase), as.factor(pred))
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4158  375
         1  233   56

               Accuracy : 0.8739
                 95% CI : (0.8642, 0.8832)
    No Information Rate : 0.9106
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0903

 Mcnemar's Test P-Value : 1.076e-08

            Sensitivity : 0.9469
            Specificity : 0.1299
         Pos Pred Value : 0.9173
         Neg Pred Value : 0.1938
             Prevalence : 0.9106
         Detection Rate : 0.8623
   Detection Prevalence : 0.9401
      Balanced Accuracy : 0.5384

       'Positive' Class : 0
```