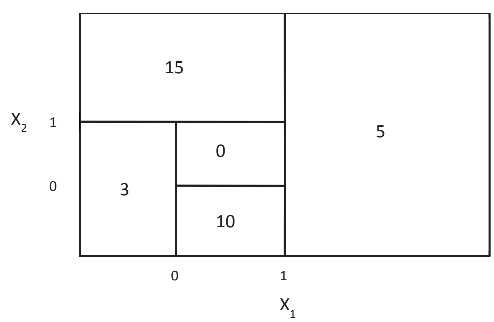
MATH 4322 Homework 5 Solutions

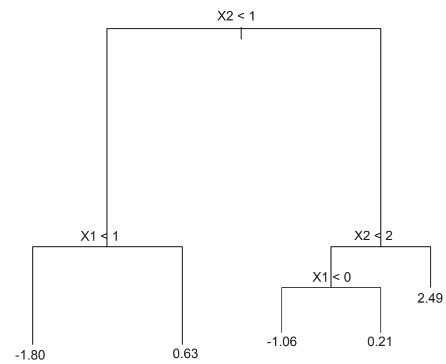
Cathy Poliak

Fall 2021

Problem 1

The questions relate to the following plots:

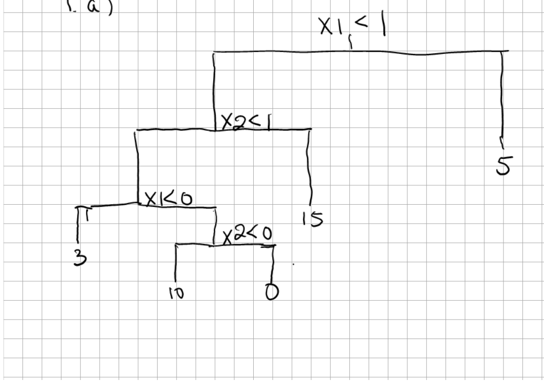




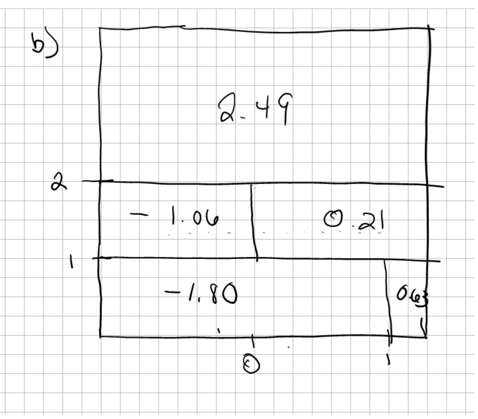
a) Sketch the tree corresponding to the partition of the predictor space illustrated on the left-hand plot. The numbers inside the boxes indicate the mean of Y within each region.

b) Create a diagram similar to the left-hand plot using the tree illustrated in the right-hand plot. You should divide up the predictor space inot the correct regions, and indicate the mean for each region.

1. a)



b)



Problem 2

Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of X, produce 10 estimates of $P(\text{Class is Red}|X)$:

$$0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, \text{ and } 0.75.$$

There are two common ways to combine these results together into a single class prediction. One is the majority vote approach discussed in this chapter. The second approach is to classify based on the average probability. In this example, what is the final classification under each of these two approaches?

```r
px = c(0.1,0.15,0.2,0.2,0.55,0.6,0.6,0.65,0.7,0.75)
#Majority vote
vote = ifelse(px>= 0.5,1,0)
sum(vote)
```

```
## [1] 6
```

```r
#Average
mean(px)
```

```
## [1] 0.45
```

With the majority vote we get 6 out of 10 to be Red thus this approach would say that we have Red.
The average approach is at 0.45 which is less than 0.5, thus we would say with this approach we have Green.

Problem 3

Provide a detailed explanation of the algorithm that is used to fit a regression tree.

Answer

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

(a) Repeat Steps 1 and 2 on all but the kth fold of the training data.
(b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Problem 4

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.

```
library(ISLR)
data(OJ)
set.seed(1000)
train = sample(nrow(OJ),800)
train.oj = OJ[train,]
test.oj = OJ[-train,]
```

b) Fit a tree to the training data, with `Purchase` as the response and the other variables as predictors. Use the `summary()` function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

```
library(tree)
tree.oj = tree(Purchase ~ ., OJ, subset = train)
summary(tree.oj)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ, subset = train)
## Variables actually used in tree construction:
## [1] "LoyalCH"     "PriceDiff"   "SalePriceMM"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7486 = 592.9 / 792
## Misclassification error rate: 0.16 = 128 / 800
```

Training error rate: 16% Number of 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.

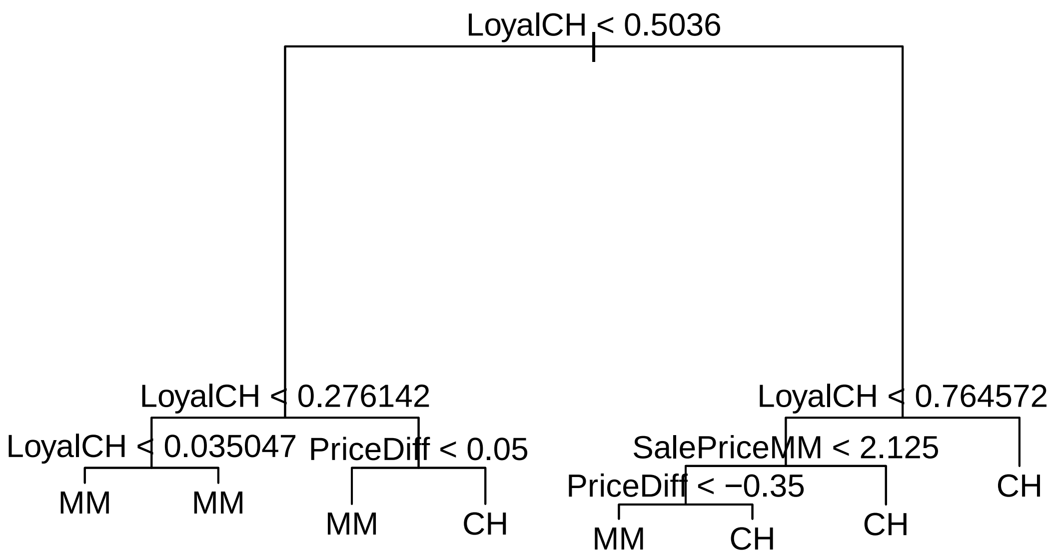
```
tree.oj
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1066.00 CH ( 0.61500 0.38500 )
##    2) LoyalCH < 0.5036 353  422.60 MM ( 0.28612 0.71388 )
##      4) LoyalCH < 0.276142 170  131.00 MM ( 0.12941 0.87059 )
##        8) LoyalCH < 0.035047 57   10.07 MM ( 0.01754 0.98246 ) *
##        9) LoyalCH > 0.035047 113  108.50 MM ( 0.18584 0.81416 ) *
##      5) LoyalCH > 0.276142 183  250.30 MM ( 0.43169 0.56831 )
##       10) PriceDiff < 0.05 78   79.16 MM ( 0.20513 0.79487 ) *
##       11) PriceDiff > 0.05 105  141.30 CH ( 0.60000 0.40000 ) *
##    3) LoyalCH > 0.5036 447  337.30 CH ( 0.87472 0.12528 )
##      6) LoyalCH < 0.764572 187  206.40 CH ( 0.75936 0.24064 )
##       12) SalePriceMM < 2.125 120  156.60 CH ( 0.64167 0.35833 )
##         24) PriceDiff < -0.35 16   17.99 MM ( 0.25000 0.75000 ) *
##         25) PriceDiff > -0.35 104  126.70 CH ( 0.70192 0.29808 ) *
##       13) SalePriceMM > 2.125 67   17.99 CH ( 0.97015 0.02985 ) *
##      7) LoyalCH > 0.764572 260   91.11 CH ( 0.95769 0.04231 ) *
```

From my node 2): If LoyalCH $<$ 0.5036 there are 353 customers with this criteria the deviance is 422.6, the chance that the customer will by Minute Made is 71.388%.

d) Create a plot of the tree, and interpret the results.

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



The variables that appears to be used to predict if they will buy MM or CH is "LoyalCH", "SalePriceMM", and "PriceDiff".

e) Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

```
tree.pred = predict(tree.oj,test.oj,type = "class")
(con.matrix = table(tree.pred,test.oj$Purchase))
```

```
##
## tree.pred  CH  MM
##        CH 150  38
##        MM  11  71
```

```
#Test error rate
(con.matrix[1,2]+con.matrix[2,1])/sum(con.matrix)
```

```
## [1] 0.1814815
```

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

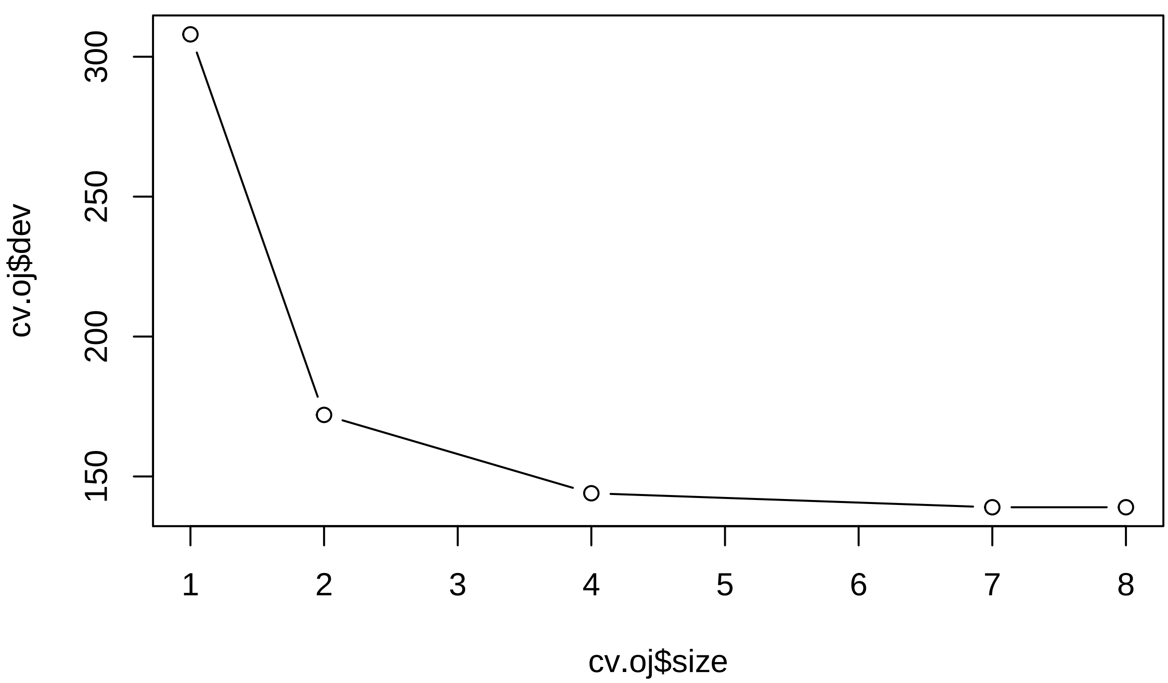
g) Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis.

h) Which tree size corresponds to the lowest cross-validated classification error rate?

```r
set.seed(2)
cv.oj = cv.tree(tree.oj,FUN = prune.misclass)
cv.oj
```

```
## $size
## [1] 8 7 4 2 1
##
## $dev
## [1] 139 139 144 172 308
##
## $k
## [1]       -Inf   0.000000   2.666667  10.500000 151.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

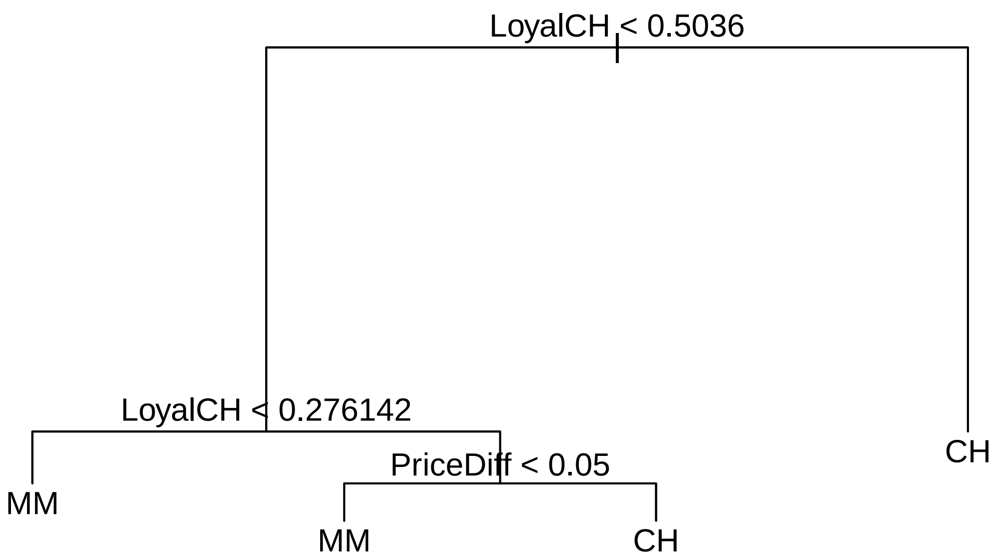
```r
plot(cv.oj$size,cv.oj$dev,type = "b")
```



It appears that the optimal tree size would be 4.

i) Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to selection of a pruned tree, then create a pruned tree with five terminal nodes.

```
prune.oj = prune.misclass(tree.oj,best = 4)
plot(prune.oj)
text(prune.oj,pretty = 0)
```



j) Compare the training error rates between the pruned and unpruned trees. Which is higher?

```
train.tree = predict(tree.oj,type = "class")
(train.matrix = table(train.tree,train.oj$Purchase))
```

```
##
## train.tree  CH  MM
##         CH 450  86
##         MM  42 222
```

```
#Train error rate Un-pruned
(train.matrix[1,2]+train.matrix[2,1])/sum(train.matrix)
```

```
## [1] 0.16
```

```
train.prune = predict(prune.oj,type = "class")
(prune.matrix = table(train.prune,train.oj$Purchase))
```

```
##
## train.prune  CH  MM
##          CH 454  98
##          MM  38 210
```

```
#Train error rate Pruned
(prune.matrix[1,2]+prune.matrix[2,1])/sum(prune.matrix)
```

```
## [1] 0.17
```

The test error rate is higher for the pruned trees.

k) Compare the test error rates between the pruned and unpruned trees. Which is higher?

```
test.tree = predict(tree.oj,test.oj,type = "class")
(test.matrix = table(test.tree,test.oj$Purchase))
```

```
##
## test.tree  CH  MM
##        CH 150  38
##        MM  11  71
```

```
#Train error rate Un-pruned
(test.matrix[1,2]+test.matrix[2,1])/sum(test.matrix)
```

```
## [1] 0.1814815
```

```
test.prune = predict(prune.oj,test.oj,type = "class")
(prune.test.matrix = table(test.prune,test.oj$Purchase))
```

```
##
## test.prune  CH  MM
##         CH 150  44
##         MM  11  65
```

```
#Train error rate Pruned
(prune.test.matrix[1,2]+prune.test.matrix[2,1])/sum(prune.test.matrix)
```

```
## [1] 0.2037037
```

Problem 5

We will use the `Carseats` data set that is in the `ISLR` package to see to predict `Sales` using regression trees and related approaches.

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

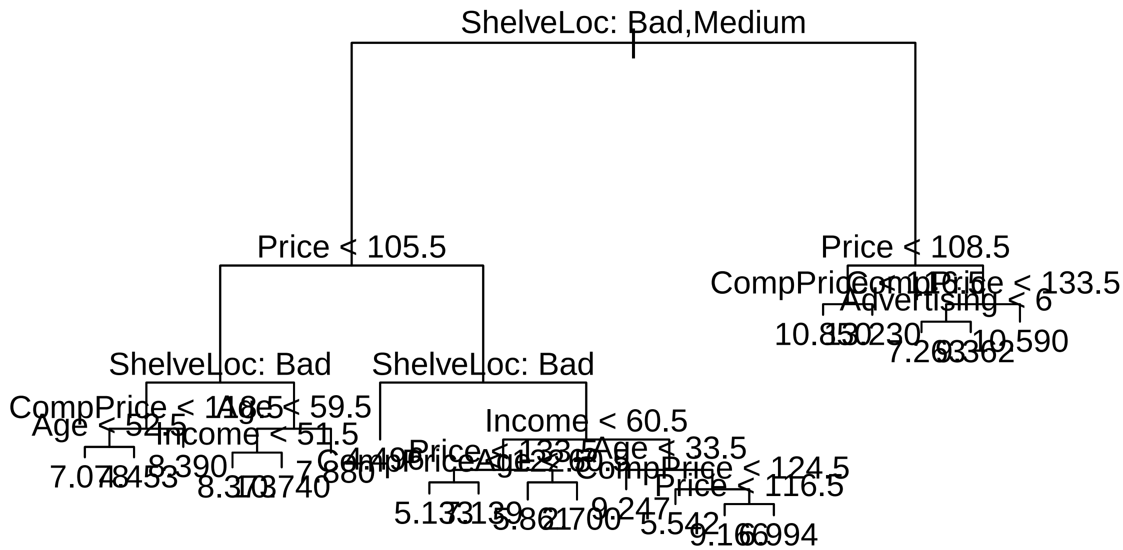
```
set.seed(20)
index = sample(nrow(Carseats),round(0.7*nrow(Carseats)))
train = Carseats[index,]
test = Carseats[-index,]
```

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

```
tree.carseats = tree(Sales ~ ., train)
summary(tree.carseats)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "CompPrice"   "Age"         "Income"
## [6] "Advertising"
## Number of terminal nodes:  20
## Residual mean deviance:  2.363 = 614.4 / 260
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.18400 -0.88550 -0.08422  0.00000  0.95770  4.61000
```

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



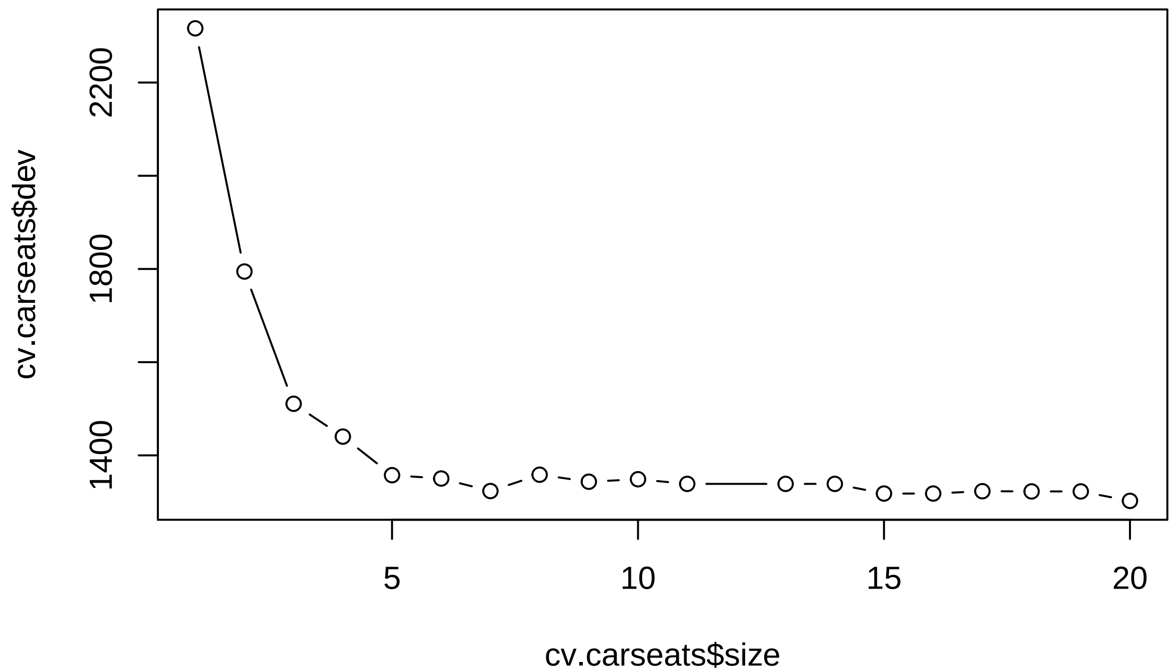

```
yhat = predict(tree.carseats,newdata = test)
#Test MSE
(test.mse = mean((yhat - test$Sales)^2))
```

```
## [1] 5.157823
```

There are 20 nodes to this tree. The variables that are used is ShelveLoc, Price, CompPrice, Age, Income and Advertising. With 20 nodes this is very hard to interpret.

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

```
cv.carseats = cv.tree(tree.carseats)
plot(cv.carseats$size,cv.carseats$dev,type = "b")
```




```
cv.carseats
```

```
## $size
##  [1] 20 19 18 17 16 15 14 13 11 10  9  8  7  6  5  4  3  2  1
##
## $dev
##  [1] 1302.356 1322.618 1322.618 1323.071 1318.090 1318.090 1338.809 1338.809
##  [9] 1338.809 1348.762 1343.454 1358.499 1323.423 1350.313 1357.446 1440.282
## [17] 1510.697 1794.510 2316.361
##
## $k
##  [1]      -Inf  24.80625  24.89140  25.46601  26.76843  26.80908  35.20185
##  [8]  35.24160  35.43344  41.76738  43.47571  46.30820  56.97002  72.49995
## [15]  92.24290 110.17098 135.94069 279.25998 531.52217
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

It appears that pruning to 7 would be best.

```
prune.carseats = prune.tree(tree.carseats,best = 7)
prune.yhat = predict(prune.carseats,newdata = test)
```

```
#Test MSE
(mse.prune = mean((prune.yhat - test$Sales)^2))
```

```
## [1] 4.679617
```

This does improve the test MSE.

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.

```
library(randomForest)
```

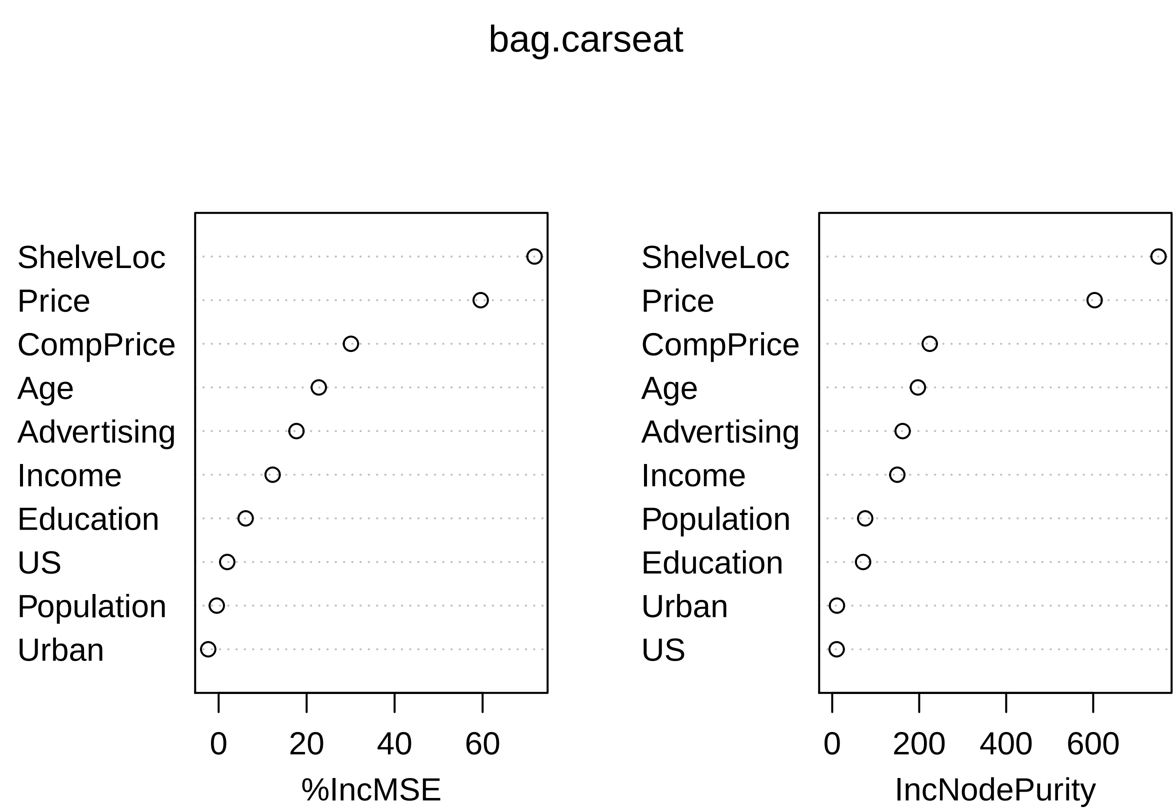
```
## randomForest 4.6-14
```

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

```
bag.carseat = randomForest(Sales ~., train, mtry = 10, importance = TRUE)
bag.yhat = predict(bag.carseat, newdata = test)
#Test MSE
(bag.mse = mean((bag.yhat - test$Sales)^2))
```

```
## [1] 2.587705
```

```
#Imporant Variables
varImpPlot(bag.carseat)
```



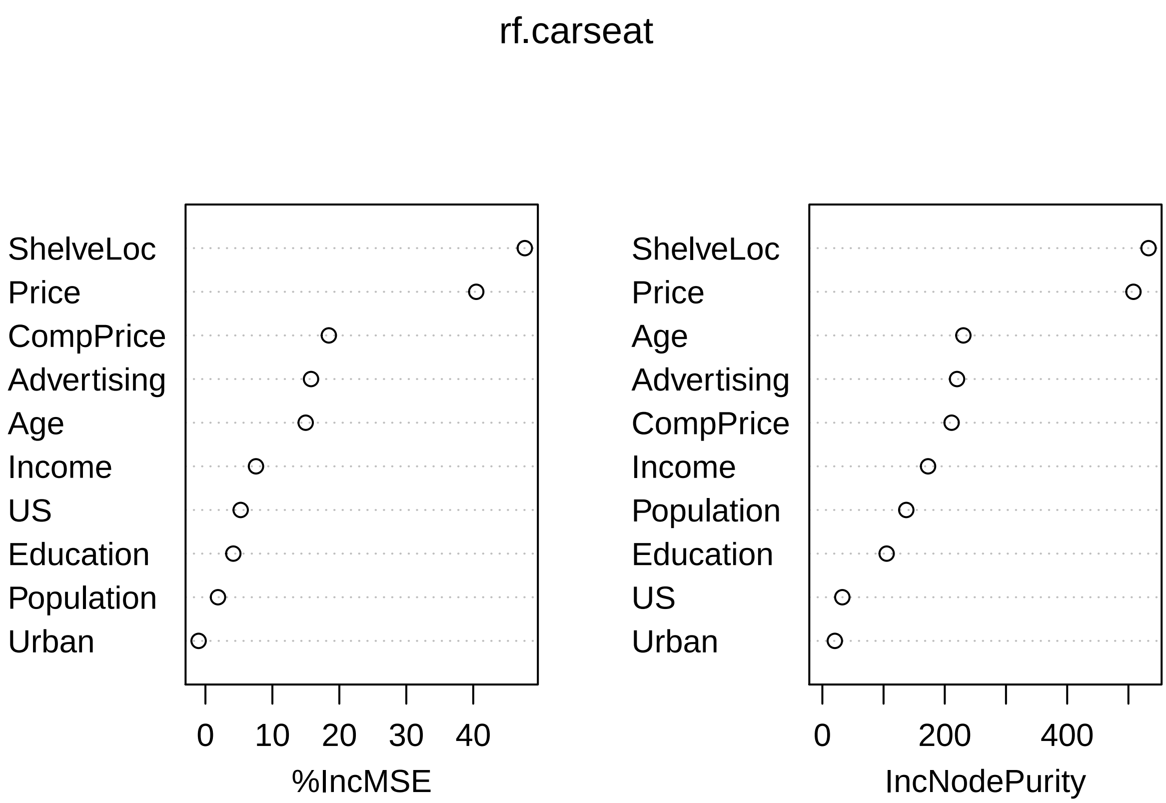
The two most important variables are `ShelveLoc` and `Price`.

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

```
rf.carseat = randomForest(Sales ~., train, mtry = sqrt(10), importance = TRUE)
rf.yhat = predict(rf.carseat, newdata = test)
#Test MSE
(rf.mse = mean((rf.yhat - test$Sales)^2))
```

```
## [1] 2.89114
```

```
#Imporant Variables
varImpPlot(rf.carseat)
```



For my random samples, the random forests did not yeild much of an imporvement over the bagging.

Problem 6

We will 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.

```
Hitters2 = na.omit(Hitters)
Hitters2$Salary = log(Hitters2$Salary)
dim(Hitters2)
```

```
## [1] 263  20
```

b) Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

```
train.hitters = Hitters2[1:200,]
test.hitters = Hitters2[201:263,]
```

c) Perform boosting on the training set with 1,000 trees. What is the the test set MSE? Compare this to the MSE for the regression tree we did in class.

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

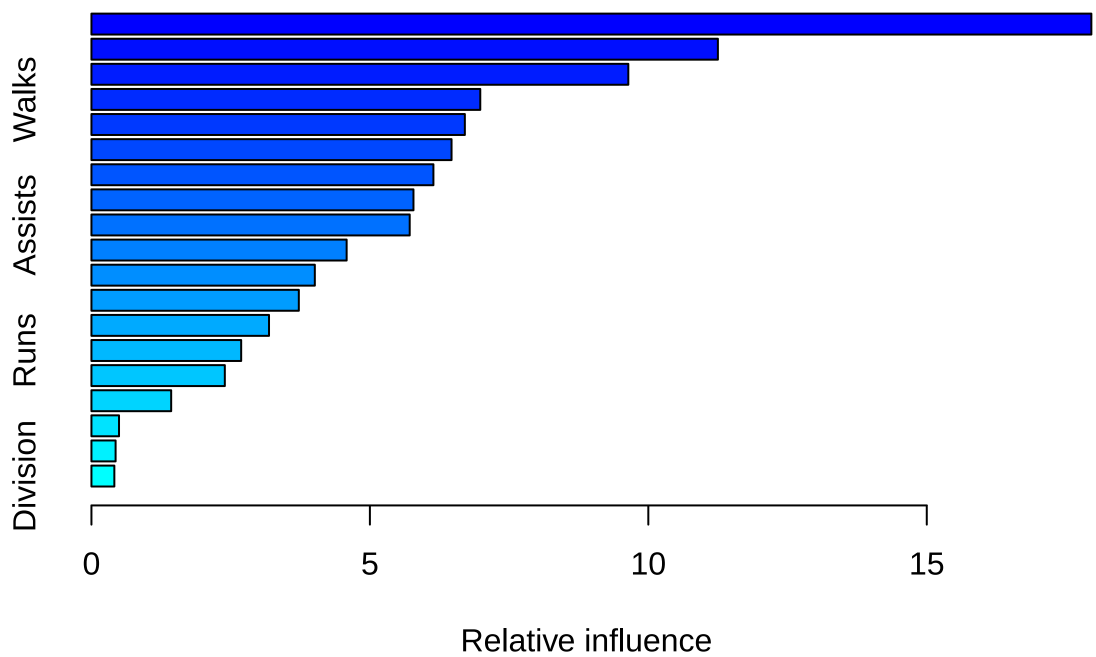
```
set.seed(2)
boost.hit = gbm(Salary ~., train.hitters, distribution = "gaussian",
                n.trees = 1000)
yhat.boost = predict(boost.hit, newdata = test.hitters,n.trees = 1000)
mean((yhat.boost - test.hitters$Salary)^2)
```

```
## [1] 0.2678773
```

This is smaller than the test MSE we received in class.

d) Which variables appear to be the most imporatnt predictors in the boosted model?

```
summary(boost.hit)
```




```
##                 var    rel.inf
## CAtBat       CAtBat 17.9553887
## CRuns         CRuns 11.2492102
## CRBI           CRBI  9.6408476
## Walks         Walks  6.9820679
## PutOuts     PutOuts  6.7054644
## CWalks       CWalks  6.4662858
## CHmRun       CHmRun  6.1405886
## Years         Years  5.7824775
## Assists     Assists  5.7148646
## AtBat         AtBat  4.5825777
## RBI             RBI  4.0114290
## Hits           Hits  3.7240791
## HmRun         HmRun  3.1885899
## Runs           Runs  2.6886674
## Errors       Errors  2.3953373
## CHits         CHits  1.4318054
## NewLeague NewLeague  0.4953537
## League       League  0.4337771
## Division   Division  0.4111881
```

Most important 3 variables: CAtBat, CRuns, and CRBI

e) Now apply bagging to the training set. What is the test set MSE for this approach?

```
bag.hit = randomForest(Salary ~ ., train.hitters, mtry = 19, importance = TRUE)
yhat.bag.hit = predict(bag.hit,test.hitters)
mean((yhat.bag.hit - test.hitters$Salary)^2 )
```

```
## [1] 0.2307637
```

This is a little bit smaller but not much of an improvement over the boosting.