Bruce Campbell ST-617 Homework 4

Wed Jul 20 15:04:33 2016

```
rm(list = ls())
set.seed(7)
```

Chapter 8

Problem 8

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 quantitative variable.

a)

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

```
library(ISLR)
attach(Carseats)
train = sample(nrow(Carseats), floor(nrow(Carseats) * 2/3))
DF <- Carseats
DFTrain <- DF[train, ]
DFTest <- DF[-train, ]
```

b)

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

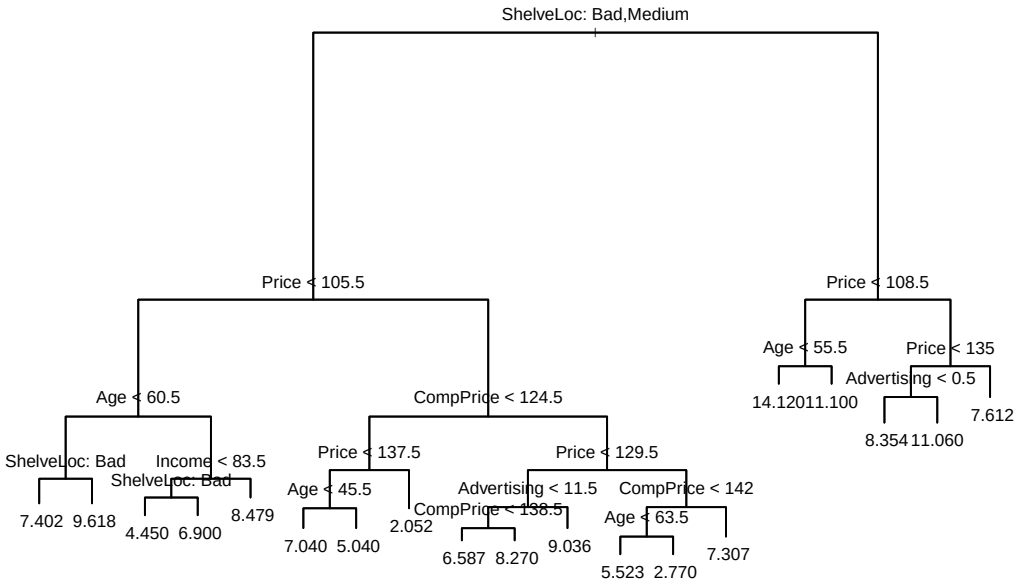
```
library(tree)
tree.carseat = tree(Sales ~ ., DFTrain)

# Here we can set the minimum number of elements at a node.
control.settings <- tree.control(minsize = 30, nobs = nrow(DFTrain))
summary(tree.carseat)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = DFTrain)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Income"      "CompPrice"
## [6] "Advertising"
## Number of terminal nodes:  19
## Residual mean deviance:  2.305 = 569.3 / 247
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.64800 -0.97940 -0.07851  0.00000  0.93590  4.57700
```

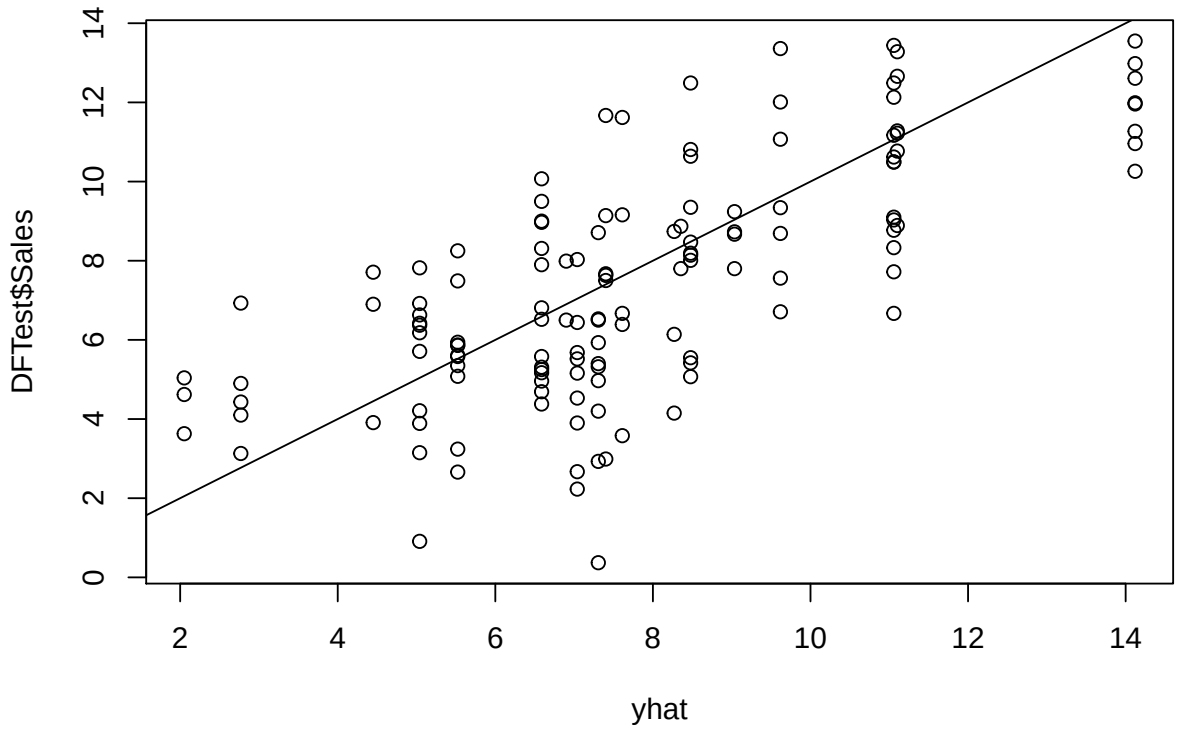
```
plot(tree.carseat, cex = 0.35)

text(tree.carseat, pretty = 0, cex = 0.6)
```



We see that ShelveLoc, Price and Age are the three most important variables. CompPrice is relevant for BadMedium shelf Loc. This is interesting as we expect those willing to consider all locations are those that would be conparing the prices.

```
yhat = predict(tree.carseat, newdata = DFTest)
plot(yhat, DFTest$Sales)
abline(0, 1)
```




```
MSE <- mean((yhat - DFTest$Sales)^2)
library(pander)

RSS <- sum((yhat - DFTest$Sales)^2)
TSS <- sum((DFTrain$Sales - mean(DFTest$Sales))^2)
RS2_Train <- 1 - (RSS/TSS)
```

The MSE of the training set is `4.9286046` and the R^2 of the training set is `0.6768345`

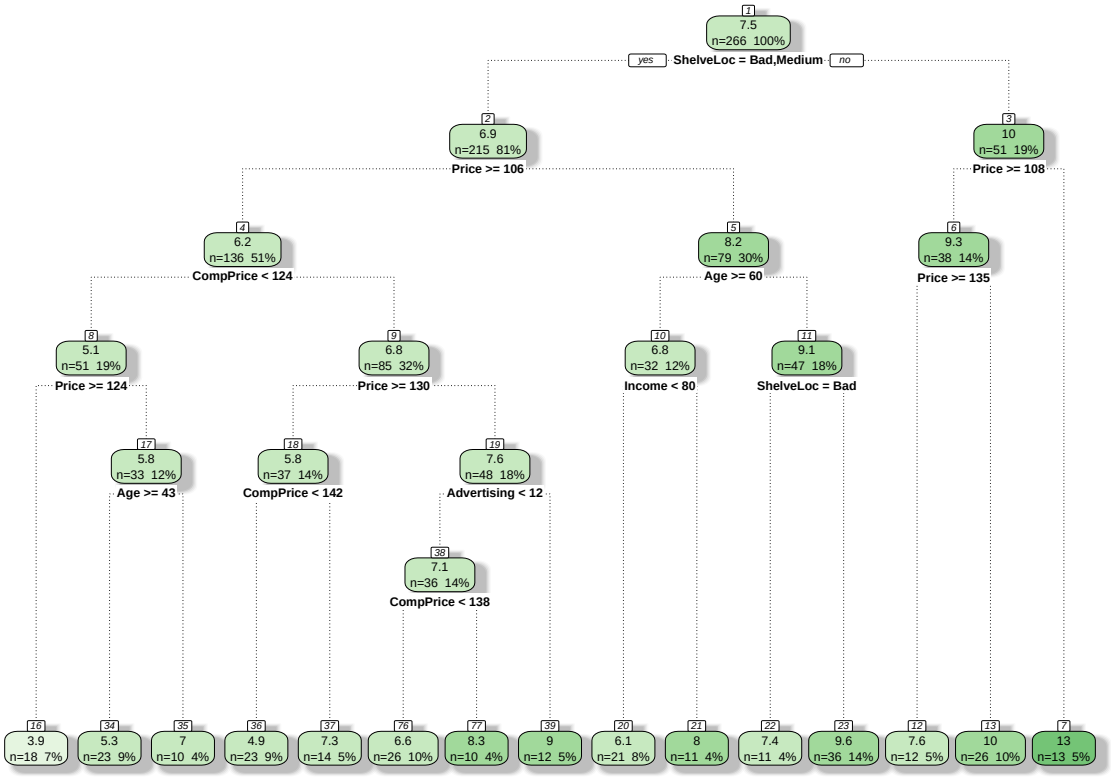
Here we take a quick look at the rpart package. The implementor of the the tree package reccomends rpart over tree.

https://stat.ethz.ch/pipermail/r-help/2005-May/070922.html

```
library(rpart)  # Popular decision tree algorithm
library(rattle)  # Fancy tree plot
library(rpart.plot)  # Enhanced tree plots
library(RColorBrewer)  # Color selection for fancy tree plot

control.settings <- rpart.control(minsplit = 30)
tree.rpart <- rpart(Sales ~ ., data = DFTrain, control = control.settings)

fancyRpartPlot(tree.rpart)
```

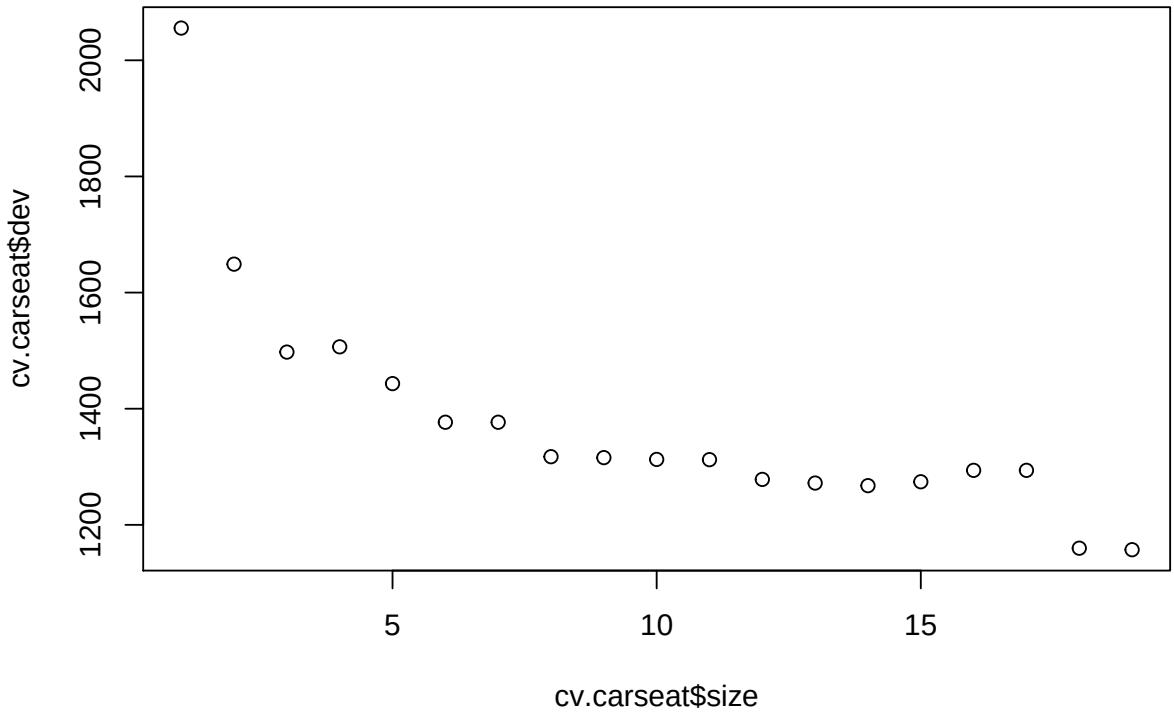


We see some differences but most of the same variables towards the top of the tree.

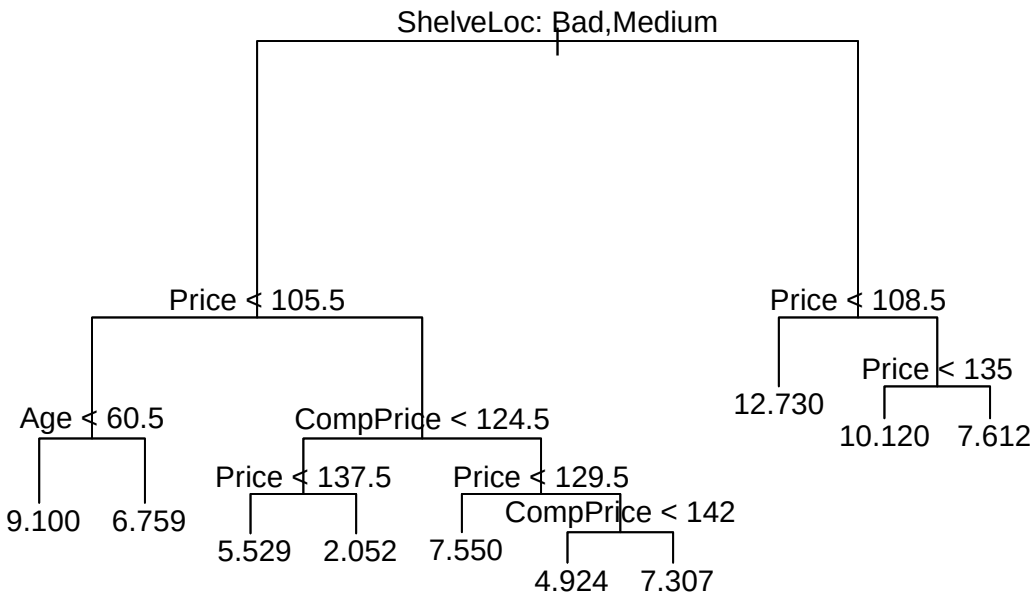
c)

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

```
cv.carseat <- cv.tree(tree.carseat)
plot(cv.carseat$size, cv.carseat$dev)
```



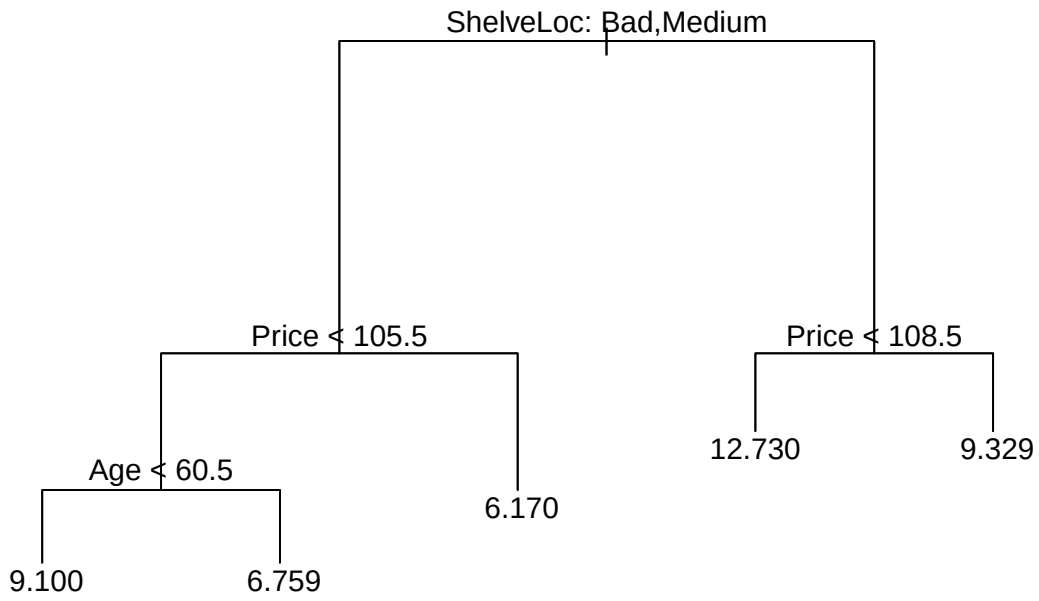

```
prune.carseat = prune.tree(tree.carseat, best = 10)
plot(prune.carseat)
text(prune.carseat, pretty = 0)
```




```
yhat = predict(prune.carseat, newdata = DFTest)
MSE_pruned_10 <- mean((yhat - DFTest$Sales)^2)
library(pander)

RSS <- sum((yhat - DFTest$Sales)^2)
TSS <- sum((DFTrain$Sales - mean(DFTest$Sales))^2)
RS2_Train_pruned_10 <- 1 - (RSS/TSS)

prune.carseat = prune.tree(tree.carseat, best = 5)
plot(prune.carseat)
text(prune.carseat, pretty = 0)
```




```
yhat = predict(prune.carseat, newdata = DFTest)

MSE_pruned_5 <- mean((yhat - DFTest$Sales)^2)
library(pander)

RSS <- sum((yhat - DFTest$Sales)^2)
TSS <- sum((DFTrain$Sales - mean(DFTest$Sales))^2)
RS2_Train_pruned_5 <- 1 - (RSS/TSS)
```

The cross validation results suggest that we try pruning at 5 and 10 terminal nodes.

```
MSE_DF <- data.frame(model = c("MSE Full", "MSE Prune 5", "MSE Prune 10"), c(MSE,
    MSE_pruned_5, MSE_pruned_10))
pander(MSE_DF)
```

model	c.MSE..MSE_pruned_5..MSE_pruned_10.
MSE Full	4.929
MSE Prune 5	4.736
MSE Prune 10	4.611

```
RSQ_DF <- data.frame(model = c("RSQ Full", "RSQ Prune 5", "RSQ Prune 10"), c(RS2_Train,
    RS2_Train_pruned_5, RS2_Train_pruned_10))
```

```
pander(RSQ_DF)
```

model	c.RS2_Train..RS2_Train_pruned_5..RS2_Train_pruned_10.
RSQ Full	0.6768
RSQ Prune 5	0.6895
RSQ Prune 10	0.6977

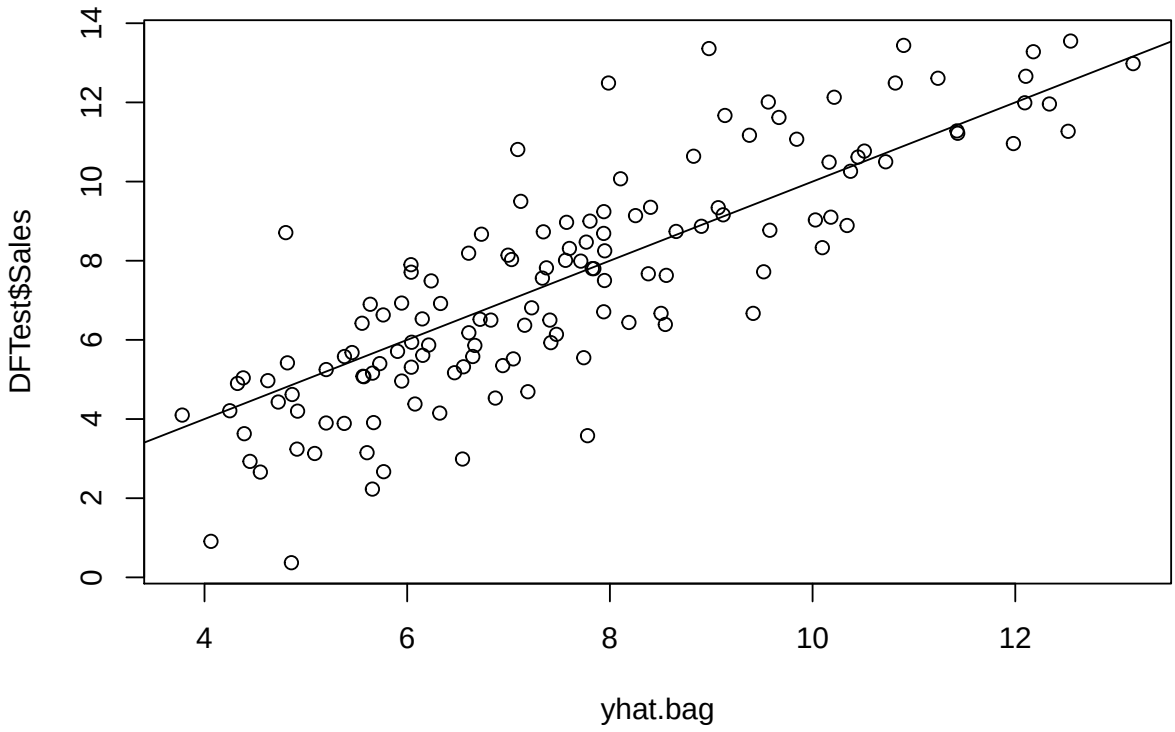
d)

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

```
library(randomForest)
bag.carseat = randomForest(Sales ~ ., data = DFTrain, mtry = 11, importance = TRUE)
bag.carseat
```

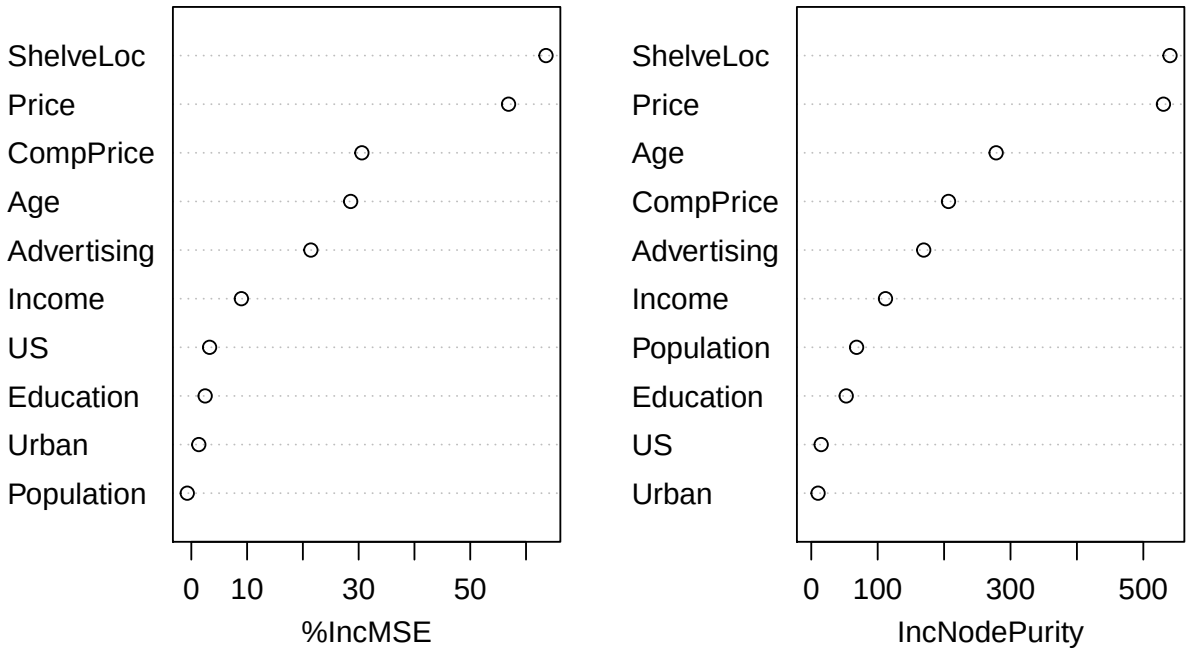
```
##
## Call:
##  randomForest(formula = Sales ~ ., data = DFTrain, mtry = 11,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 2.499298
##                     % Var explained: 67.43
```

```
yhat.bag = predict(bag.carseat, newdata = DFTest)
plot(yhat.bag, DFTest$Sales)
abline(0, 1)
```




```
MSE_Bagging <- mean((yhat.bag - DFTest$Sales)^2)


varImpPlot(bag.carseat)
```



The MSE for bagged model is `2.5398888` and is greatly reduced from that of the single regression tree MSE of `4.9286046`. We also note that the most important variables are ShelveLoc, Price, Age, CompPrice, and Advertising

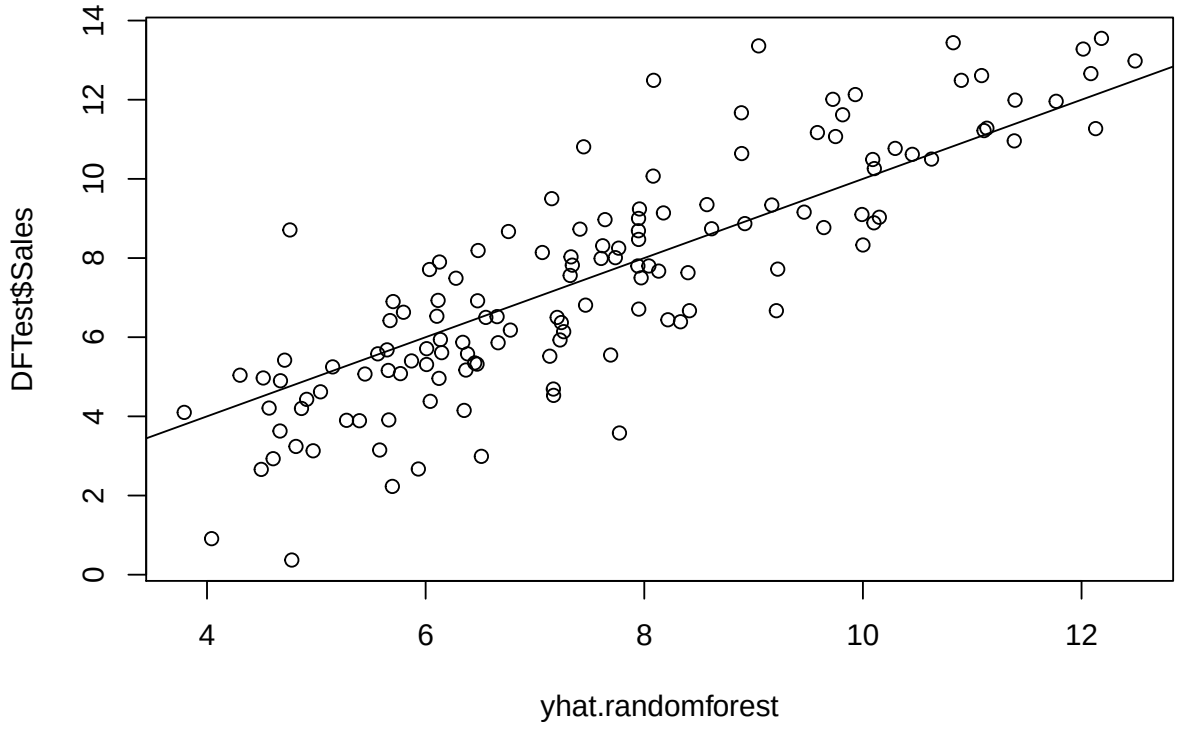
e)

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

```
randomforest.carseat = randomForest(Sales ~ ., data = DFTrain, mtry = 7, importance = TRUE)
randomforest.carseat
```

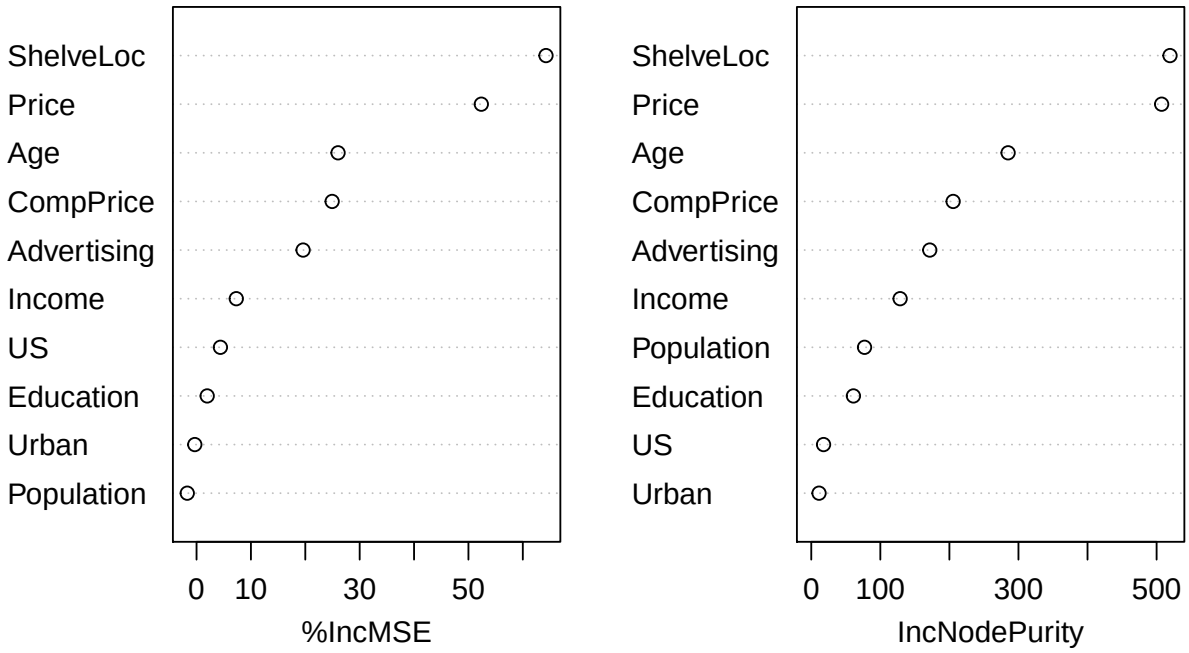
```
##
## Call:
##  randomForest(formula = Sales ~ ., data = DFTrain, mtry = 7, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##           Mean of squared residuals: 2.554792
##                     % Var explained: 66.7
```

```
yhat.randomforest = predict(randomforest.carseat, newdata = DFTest)
plot(yhat.randomforest, DFTest$Sales)
abline(0, 1)
```



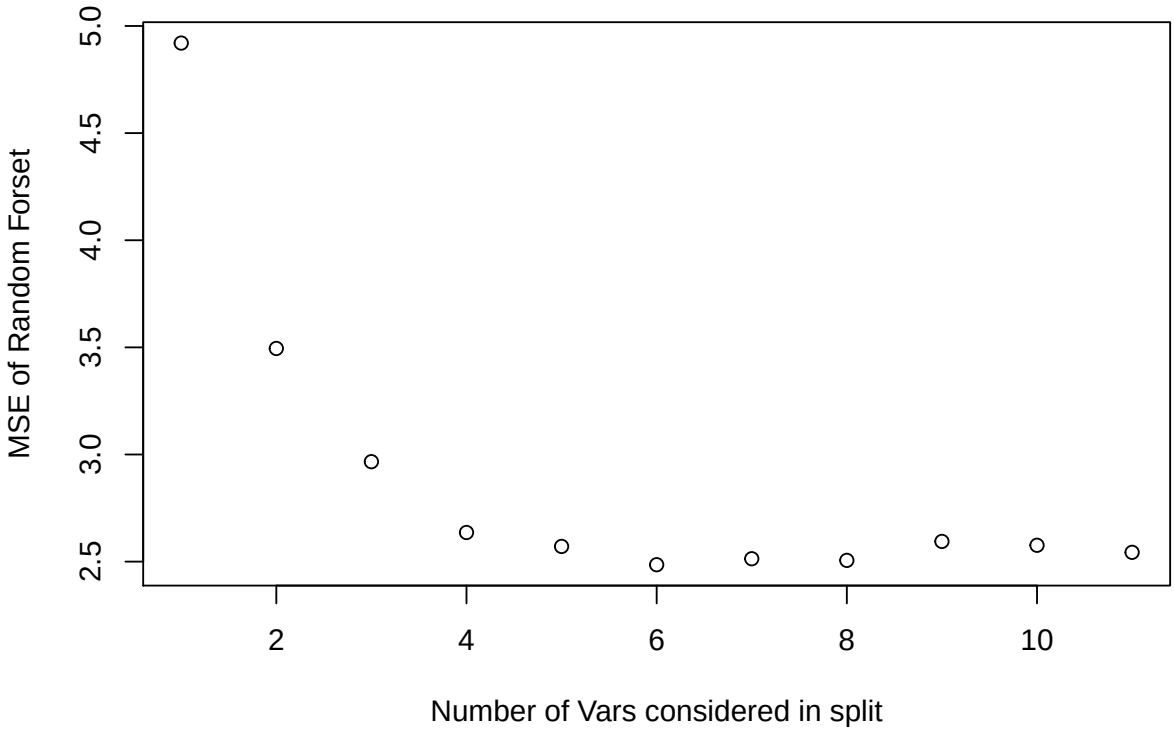

```
MSE_RandomForest <- mean((yhat.randomforest - DFTest$Sales)^2)

varImpPlot(randomforest.carseat)
```



As expected the random forest MSE (`2.4646252`) is lower than the bagging MSE (`2.5398888`)

```
mse_vev <- matrix(data = NA, nrow = 11, ncol = 1)
for (i in 1:11) {
    randomforest.carseat = randomForest(Sales ~ ., data = DFTrain, mtry = i,
        importance = TRUE)
    yhat.randomforest = predict(randomforest.carseat, newdata = DFTest)
    mse_vev[i] <- mean((yhat.randomforest - DFTest$Sales)^2)
}
plot(1:11, mse_vev, xlab = "Number of Vars considered in split", ylab = "MSE of Random Forset")
```




```
which.min(mse_vev)
```

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

The bagging importance plot has served us well. We chose an m of 7 based on that plot, and the plot above of the MSE by m confirms that 7 was the best choice. As expected we see the MSE goes down rapidly as we add variables, and then levels off.