Andrew Rothman
anr248@stanford.edu

STATS 202 – Homework #7

November 30th, 2018
SUID: 06325118

Problem #1:

Exercise 4 from section 8.4 of ITSL textbook

This question relates to the plots in Figure 8.12 below:

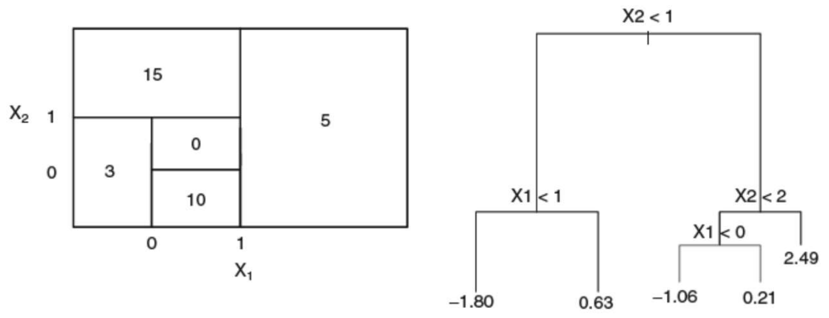
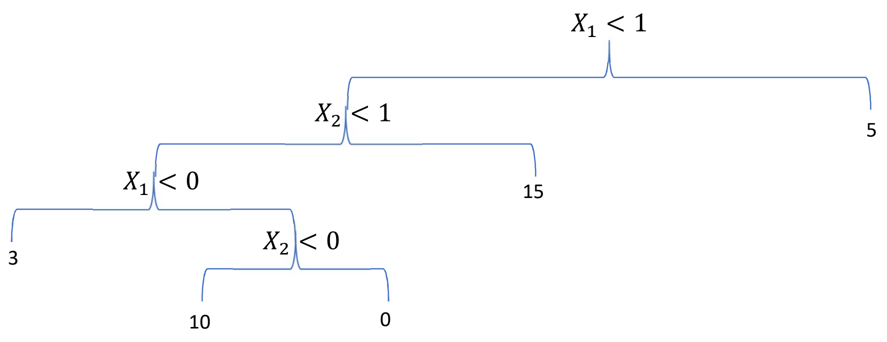


FIGURE 8.12. Left*: A partition of the predictor space corresponding to Exercise 4a.* Right*: A tree corresponding to Exercise 4b.*

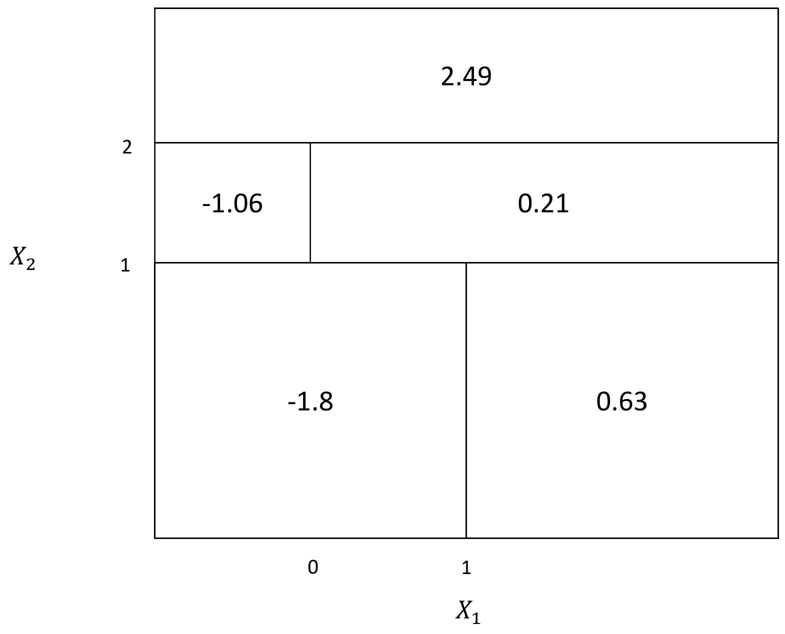
(a) Sketch the tree corresponding to the partition of the predictor space illustrated in the left-hand panel of Figure 8.12. The numbers inside the boxes indicate the mean of Y within each region.

Answer:



(b) Create a diagram similar to the left-hand panel of Figure 8.12, using the tree illustrated in the right-hand panel of the same figure. You should divide up the predictor space into the correct regions, and indicate the mean for each region.

Answer:



Problem #2:

Exercise 5 from section 8.4 of ITSL textbook

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

0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 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?

Answer:

Majority Vote Approach:
Using the majority vote approach, we first classify each of the 10 estimates into "red" or "green" classes using a cut-off to $P(Class\ is\ Red \mid X) > 0.5 = Red$

Prediction ID	**Prediction**	**Class-prediction**
1	0.1	Green
2	0.15	Green
3	0.2	Green
4	0.2	Green
5	0.55	Red
6	0.6	Red
7	0.6	Red
8	0.65	Red
9	0.7	Red
10	0.75	Red

From the table above, the majority of predictions (6 of the 10) are classified into the "red" class. Therefore based on the "majority vote" approach, the final classification class would be the "red" class.

Average Probability Approach:
The average probability of the 10 estimates is:

$$\frac{0.1 + 0.15 + 0.2 + 0.2 + 0.55 + 0.6 + 0.6 + 0.65 + 0.7 + 0.75}{10} = 0.45$$

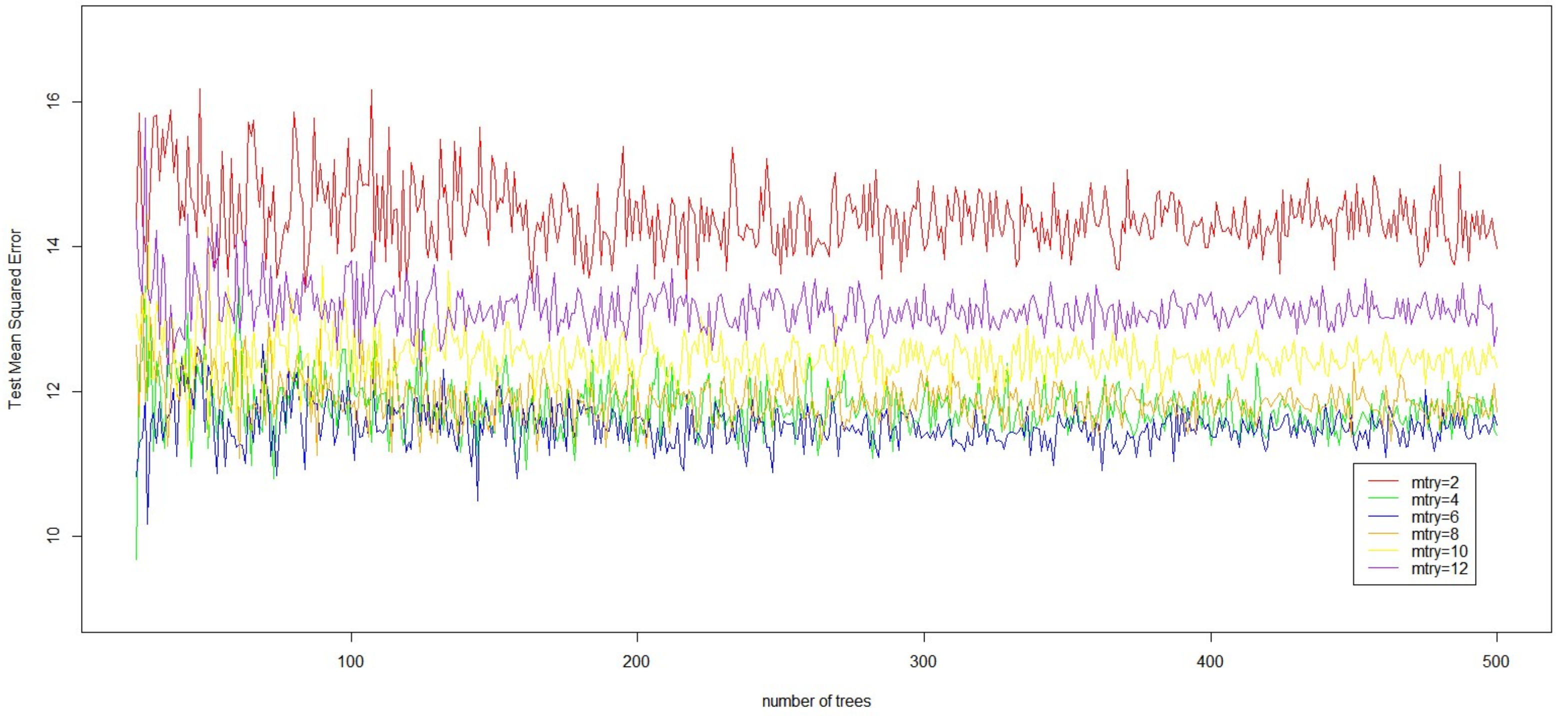
Given the average probability is 0.45 (which is ≤0.5), the final classification class would be the "green" class.

Problem #3:

Exercise 7 from section 8.4 of ITSL textbook

In the lab, we applied random forests to the “Boston” dataset 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 and ntree. You can model your plot after Figure 8.10. Describe the results obtained.

Answer:



Splitting the Boston dataset into equal sized training and testing sets (253 observations each), I ran a random forest to predict outcome “medv” with the number of candidate variables at each split-node (“mtry” parameter) set to 2,4,6,8,10, and 12 with the number of bootstrapped trees in each random forest set to 25 through 500 increased in one integer increments. The resulting test MSE was recovered for each model and is plotted above. As we can see above, as we begin to increase the mtry parameter starting at 2 the test MSE generally decreases. As we increase the mtry parameter from 6 forward the test MSE generally increases again. For each trend line with the “mtry” parameter set constant, the MSE begins to level-out around a constant variance as the number of trees increases. The trend line with mtry=6 appears to fairly consistently have the lowest test MSE as the number of trees in the forest increases. As the number of trees fit in each forest approaches 500, for the trend line with mtry=6 the MSE fluctuates slightly around test MSE≈11.5.

R-code:

```
#################
## Problem #3 ##
#################
library(randomForest)
library(MASS)
data(Boston)
set.seed(1)
train <- sample(1:nrow(Boston), nrow(Boston)/2)
boston.test <- Boston [-train ,"medv"]
df <- matrix(NA, nrow=476, ncol=13)

for(num_var in 1:13){
  for(num_tree in 25:500){
    print(num_tree)
    rf <- randomForest(medv~., data=Boston, subset=train, mtry=num_var, ntree=num_tree)
    yhat <- predict(rf, newdata=Boston[-train,])
    df[num_tree-24, num_var] <- mean((yhat - boston.test)^2)
    rm(rf, yhat)
  }
}

df <- data.frame(df)
colnames(df) <- c("v1", "v2", "v3", "v4", "v5", "v6", "v7", "v8", "v9", "v10", "v11", "v12",
"v13")
df$num_trees <- c(25:500)
summary(df)

plot(df$num_trees, df$v2, type='l', col="red", lwd=1, xlab="number of trees", ylab="Test Mean
Squared Error", ylim=c(9,17))
lines(df$num_trees, df$v4, type='l', col="green", lwd=1)
lines(df$num_trees, df$v6, type='l', col="blue", lwd=1)
lines(df$num_trees, df$v8, type='l', col="orange", lwd=1)
lines(df$num_trees, df$v10, type='l', col="yellow", lwd=1)
lines(df$num_trees, df$v12, type='l', col="purple", lwd=1)
legend(x=450,y=11, c("mtry=2", "mtry=4", "mtry=6", "mtry=8", "mtry=10", "mtry=12"),
lty=c(1,1,1,1,1,1), col=c("red","green","blue","orange","yellow","purple"))



plot(df$num_trees, df$v1, type='l', col="red", lwd=1, xlab="number of trees", ylab="Mean Squared
Error", ylim=c(9,26))
lines(df$num_trees, df$v2, type='l', col="green", lwd=1)
lines(df$num_trees, df$v3, type='l', col="blue", lwd=1)
lines(df$num_trees, df$v4, type='l', col="orange", lwd=1)
lines(df$num_trees, df$v5, type='l', col="yellow", lwd=1)
lines(df$num_trees, df$v6, type='l', col="purple", lwd=1)
lines(df$num_trees, df$v7, type='l', col="brown", lwd=1)
legend(x=450,y=20.5, c("mtry=1", "mtry=2", "mtry=3", "mtry=4", "mtry=5", "mtry=6", "mtry=7"),
lty=c(1,1,1,1,1,1,1), col=c("red","green","blue","orange","yellow","purple","brown"))
```

Problem #4:

Exercise 8 from section 8.4 of ITSL textbook

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

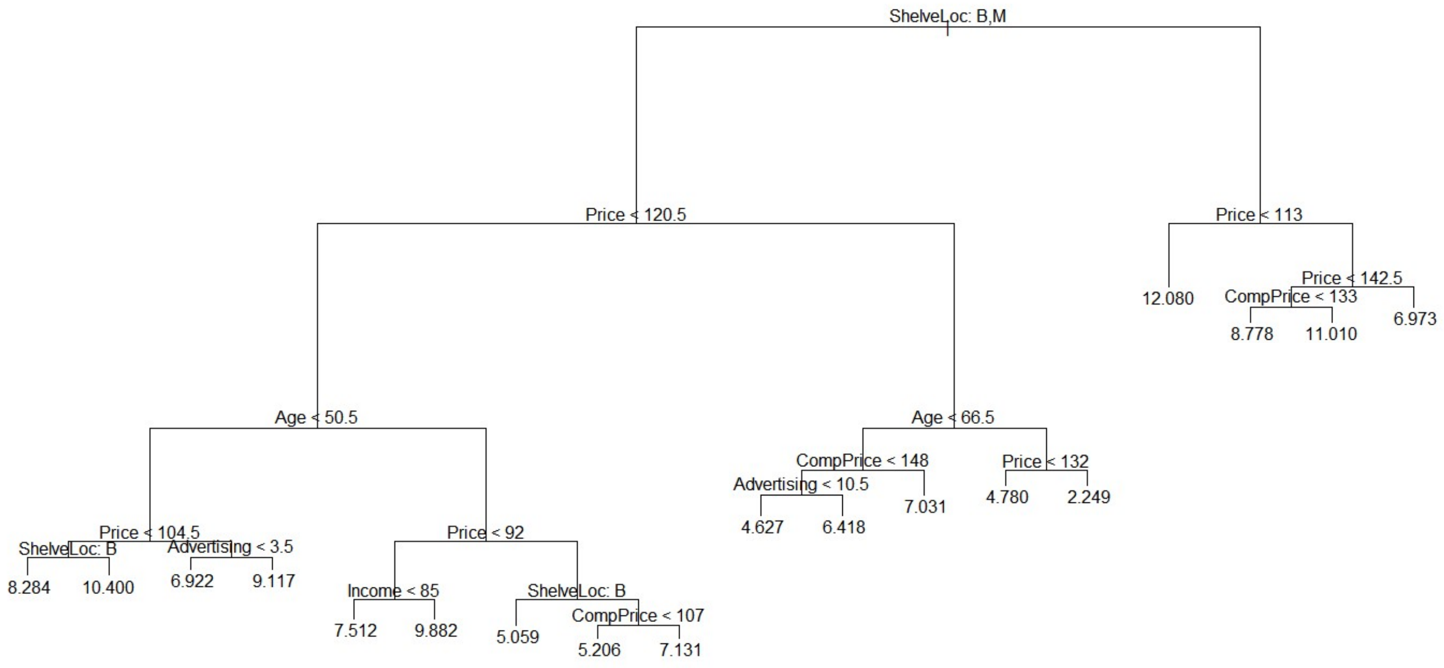
(a) Split the data set into a training set and test set.

Answer:

R-code:
```
train_id <- sample(1:nrow(Carseats), nrow(Carseats)/2)
train_df <- Carseats[train_id,]
test_df <- Carseats[-train_id,]
```

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

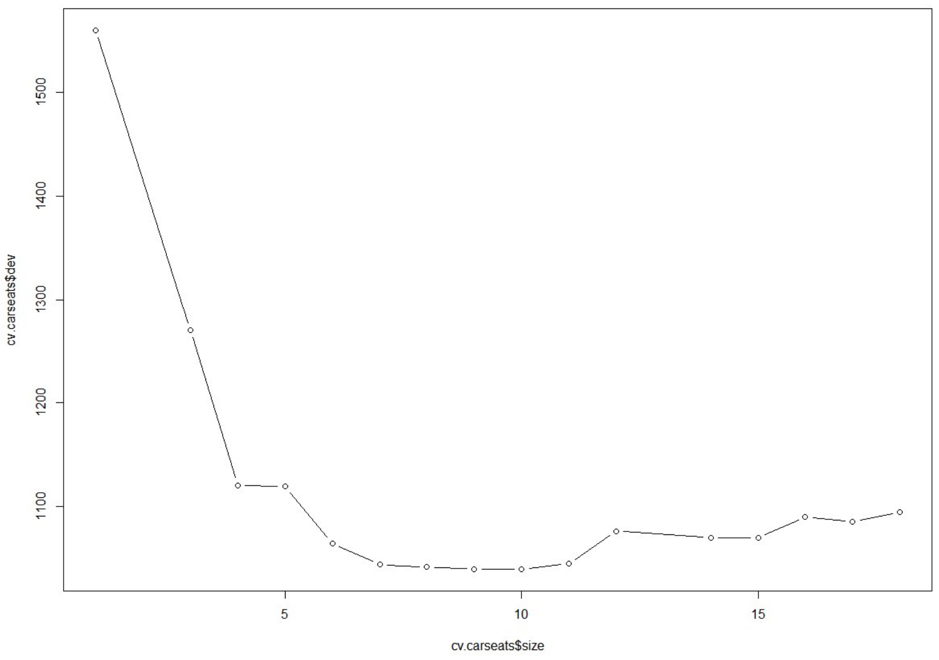
Answer:



Looking at the plotted tree above, the tree has a maximum depth of six levels from the root-node. Additionally, the tree has a total of 18 terminal nodes. The test MSE was estimated to be 4.148897.

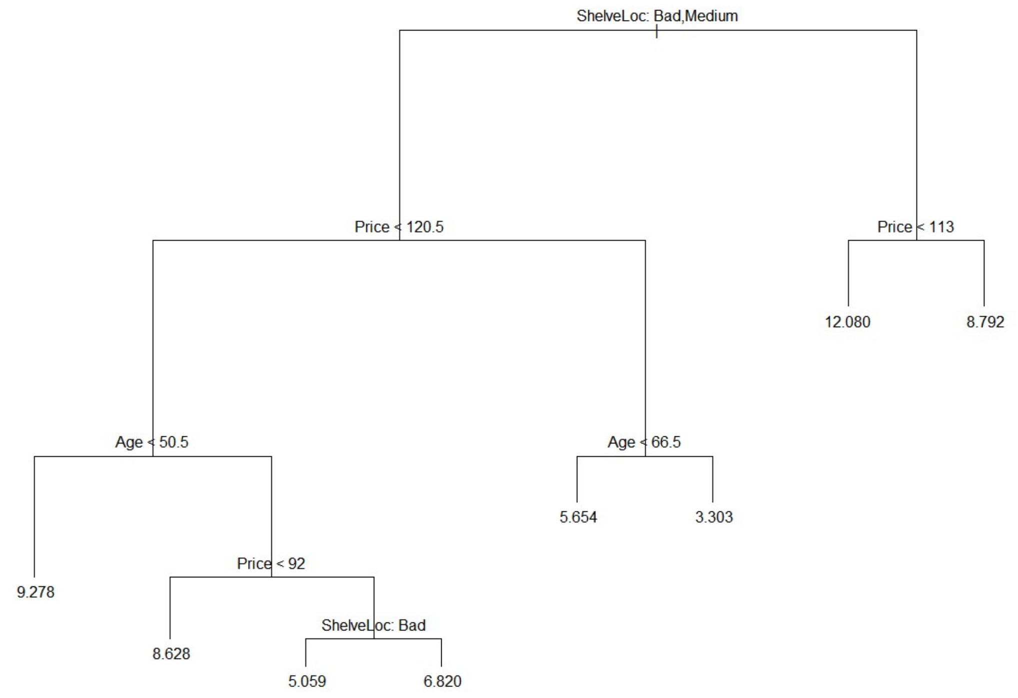
(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

Answer:
To prune the tree, we need to decide what the optimal tree size is. The "cv.tree" function was applied on the regression tree fit in part (b) to perform cross-validation to determine the optimal tree complexity. We then plot the cross-validated test error as a function of the size of the tree. The plot is shown below:



From the plot above, the cross-validation analysis on the training data revealed that the size of the tree with the lowest cross-validation error was with size (i.e. number of terminal nodes) equal to 8 (Cross-validated deviance= 1039.212).

We then fit the best possible tree (lowest training MSE) on the entire training set that has 8 terminal nodes. The resulting tree with 8 terminal nodes is shown below:



The tree above was then fit to the testing data, resulting in a test MSE= 5.09085. Comparing the test MSE on the pruned tree to the test MSE from part (b) (test MSE=4.148897), we can see pruning the tree did not improve the test MSE but rather increased 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.

Answer:
For the bagging approach, a total of 1000 bootstrapped trees were fit. The resulting test MSE obtained from fitting the resulting bagging model on the testing data was test MSE= 2.572379. We can see that this test MSE is quite an improvement over both the pruned tree (test MSE=5.09085) and the naively fit single regression tree (test MSE=4.148897).

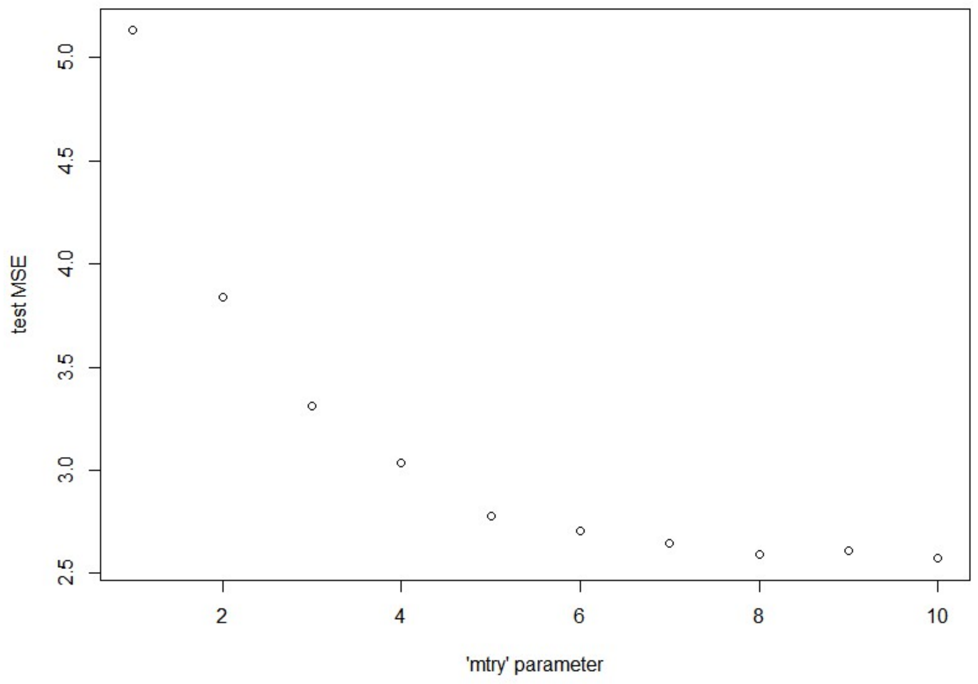
The output of the "importance" function on the resulting bagging model is shown below:

```
> importance(bag)
            IncNodePurity
CompPrice      135.264013
Income          78.893873
Advertising    125.490578
Population      61.265876
Price          505.614387
ShelveLoc      320.127726
Age            194.010348
Education       40.940231
Urban            9.175221
US              15.347388
```

From the output above, we can see that the variable "Price" is "most important" resulting in the largest mean decrease in training RSS, followed by the variable "ShelveLoc" being "second most important". The variable "Urban" had the lowest mean decrease in training RSS and would be considered least important.

(e) Use random forests to analyze the data. What test MSE 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.

Answer:
10 different Random Forest models were fit, increasing the "mtry" parameter from 1 to 10 in one unit increments. For each Random Forest model, 1000 boostrapped trees were fit. The 10 Random Forest models fit on the training data were then tested on the test data and a test MSE was recovered. The test MSE for the 10 models is shown below:



Looking at the plot above, as we vary the “mtry” parameter from 1 to 10 the test MSE initially decreases quite sharply, and begins to level out around “mtry”=5. With “mtry” set between 6 and 10 there isn’t a substantial amount of change in the test MSE.

The Random Forest model with the lowest test MSE was with the “mtry” parameter set to 10 (test MSE=2.571475). This corresponds to taking a “random subset” of size 10 of the 10 possible predictors at each node-split in each tree, and choosing the variable and split value that most minimizes the training MSE. With the “mtry” parameter set to 10, this actually corresponds to the bagging model in part (d) (choosing the variable and split value that most minimizes the training MSE among ALL of the possible 10 predictors).

Even though the best possible Random Forest model corresponds to the bagging model, just so we don't repeat describing the same results from part (d) let us instead explore the results of the Random Forest model with the second-best test MSE. This would be the Random Forest with "mtry" set to 8 (test MSE=2.594034). For the Random Forest model with mtry=8, the "importance" function produced the following output:

```
            IncNodePurity
CompPrice      132.592355
Income          82.805990
Advertising    128.739247
Population      65.283982
Price          491.137416
ShelveLoc      308.149711
Age            197.544958
Education       43.752915
Urban            9.827567
US              17.355601
```

Similar to the results of the "importance" analysis of the bagging model, for the Random Forest model the variable "Price" is "most important" resulting in the largest mean decrease in training RSS, followed by the variable "ShelveLoc" being "second most important". The variable "Urban" had the lowest mean decrease in training RSS and would be considered least important.

__R-code:__
```
################
## Problem #4 ##
################
set.seed(1)
library(tree)
library(randomForest)
library(ISLR)
data(Carseats)
summary(Carseats)

############
## Part A ##
############
train_id <- sample(1:nrow(Carseats), nrow(Carseats)/2)
train_df <- Carseats[train_id,]
test_df <- Carseats[-train_id,]

############
## Part B ##
############
tree.carseats <- tree(Sales~., data=train_df)
summary(tree.carseats)

plot(tree.carseats)
text(tree.carseats, pretty=0.95)

yhat <- predict(tree.carseats, newdata=test_df)
MSE <- mean((yhat - test_df$Sales)^2)

############
## Part C ##
############
```

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

prune.carseats <- prune.tree(tree.carseats, best=8)
plot(prune.carseats)
text(prune.carseats, pretty=0)

yhat_prune <- predict(prune.carseats, newdata=test_df)
MSE_prune <- mean((yhat_prune - test_df$Sales)^2)

############
## Part D ##
############
bag <- randomForest(Sales~., data=train_df, mtry=10, ntree=1000)
importance(bag)
yhat_bag <- predict(bag, newdata=test_df)
MSE_bag <- mean((yhat_bag - test_df$Sales)^2)

############
## Part E ##
############
MSE_vector <- rep(NA, 10)

for(i in 1:10){
  rf <- randomForest(Sales~., data=train_df, mtry=i, ntree=1000)

  if(i==8){
    print(importance(rf))
  }
  yhat_rf <- predict(rf, newdata=test_df)
  MSE_vector[i] <- mean((yhat_rf - test_df$Sales)^2)
  rm(rf, yhat_rf)
}
MSE_vector
mtry <- c(1:10)

plot(x=mtry, y=MSE_vector, xlab="'mtry' parameter", ylab="test MSE")
```

Problem #5:

Exercise 9 from section 8.4 of ITSL textbook

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.

Answer:

R-code:

```
train_id <- sample(1:nrow(OJ), 800)
train_df <- OJ[train_id,]
test_df <- OJ[-train_id,]
```

(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?

Answer:

The output of the "summary()" function is below:

```
Classification tree:
tree(formula = Purchase ~ ., data = train_df)
Variables actually used in tree construction:
[1] "LoyalCH"       "PriceDiff"     "SpecialCH"     "ListPriceDiff"
Number of terminal nodes:  8
Residual mean deviance:  0.7305 = 578.6 / 792
Misclassification error rate: 0.165 = 132 / 800
```

We can see above that of the 17 possible predictors, the constructed tree uses only 4 of these predictors (variables "LovalCH", "PriceDiff", "SpecialCH", and "ListPriceDiff"). The constructed tree has a residual mean deviance of 0.7305, a total of 8 terminal nodes and has a training error (misclassification rate) of 16.5%.

(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.

Answer:

The R output of printing the tree object is below:

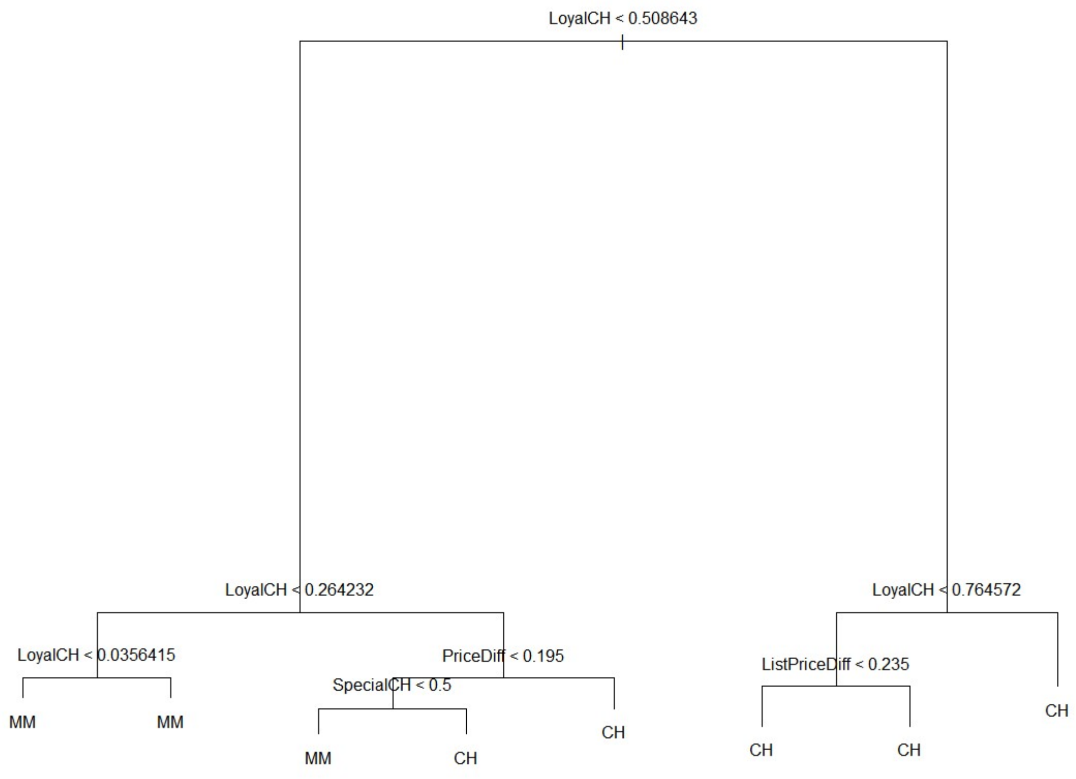
```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 800 1064.00 CH ( 0.61750 0.38250 )
   2) LoyalCH < 0.508643 350  409.30 MM ( 0.27143 0.72857 )
     4) LoyalCH < 0.264232 166  122.10 MM ( 0.12048 0.87952 )
       8) LoyalCH < 0.0356415 57   10.07 MM ( 0.01754 0.98246 ) *
       9) LoyalCH > 0.0356415 109  100.90 MM ( 0.17431 0.82569 ) *
     5) LoyalCH > 0.264232 184  248.80 MM ( 0.40761 0.59239 )
      10) PriceDiff < 0.195 83   91.66 MM ( 0.24096 0.75904 )
        20) SpecialCH < 0.5 70   60.89 MM ( 0.15714 0.84286 ) *
        21) SpecialCH > 0.5 13   16.05 CH ( 0.69231 0.30769 ) *
      11) PriceDiff > 0.195 101  139.20 CH ( 0.54455 0.45545 ) *
   3) LoyalCH > 0.508643 450  318.10 CH ( 0.88667 0.11333 )
     6) LoyalCH < 0.764572 172  188.90 CH ( 0.76163 0.23837 )
      12) ListPriceDiff < 0.235 70   95.61 CH ( 0.57143 0.42857 ) *
      13) ListPriceDiff > 0.235 102   69.76 CH ( 0.89216 0.10784 ) *
     7) LoyalCH > 0.764572 278   86.14 CH ( 0.96403 0.03597 ) *
```

Let us choose terminal node “8” to interpret in the output above. The split criterion for observations ending at terminal node “8” is based on the variable “LoyalCH” being <0.0356415. The number of observations ending at node “8” are 57 observations with an overall deviance of 10.07 and a “majority vote” prediction for this node being outcome “MM”. In the training data, approximately 1.8% of the 57 observations ending at node “8” have outcome “CH” with the remaining 98.2% having outcome “MM”.

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

Answer:



From the tree above, we can see the predictor "LoyalCH" is a very important predictor of the outcome "Purchase" given it is the variable split at the root-node. In fact, the next two split nodes are also based on the "LoyalCH" variable, also indicating the importance of "LoyalCH" in predicting "Purchase".

(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?

Answer:

Predicting the response on the test data, the resulting confusion matrix is shown below:

		Actual "Purchase"	
		CH	**MM**
Predicted "Purchase"	**CH**	147	49
	MM	12	62

$$Test\ error\ rate = \frac{(49+12)}{(147+12+49+62)} \approx 0.2259$$

From the confusion matrix and the calculation above, the test error rate is approximately 22.59%.

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

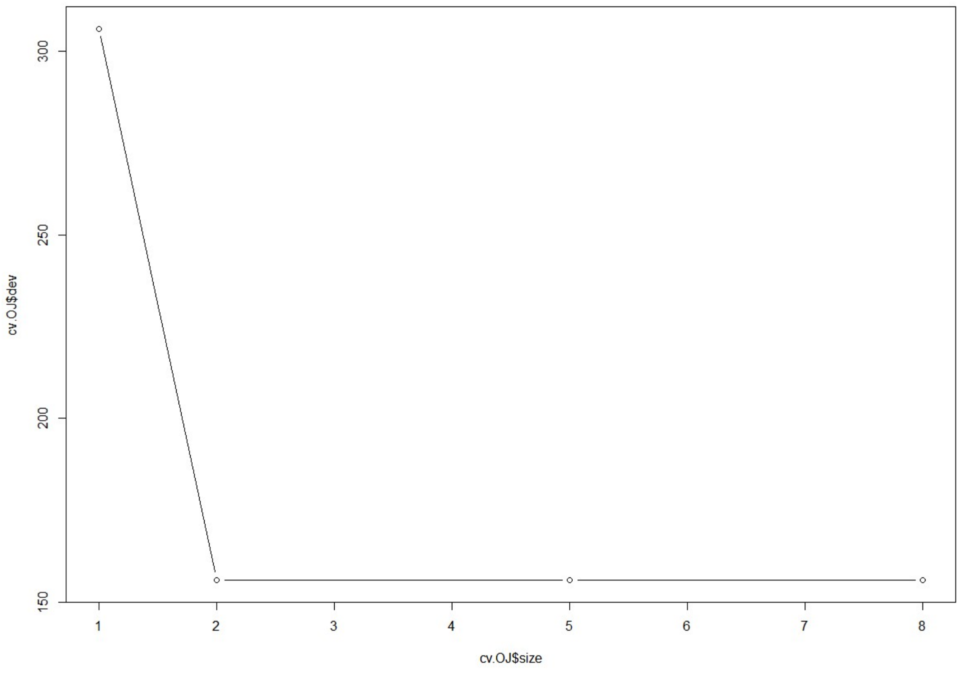
Answer:

R-code:

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

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

Answer:
The resulting plot is below:

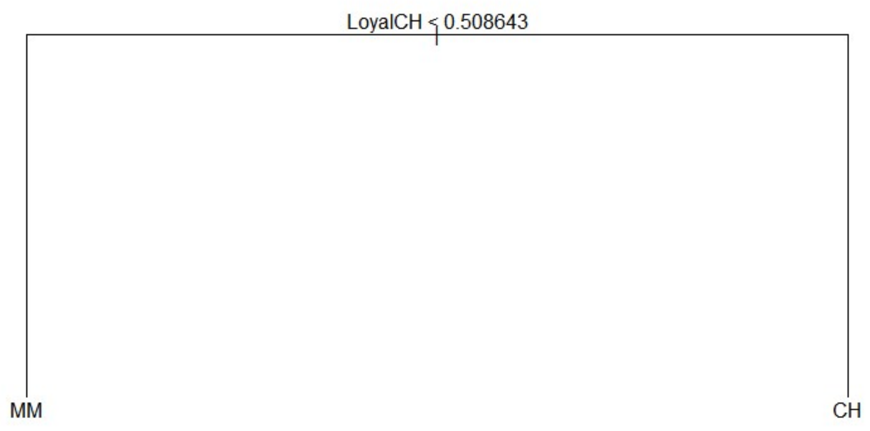


(h) Which tree size corresponds to the lowest cross-validated classification error rate?

Answer:
The lowest cross-validated deviance is 156. From the plot in part (g) we can see there are in fact several trees that share this lowest cross-validated deviance (trees with size 2 through 8). Therefore we will choose the least complex tree with this lowest deviance, which would be a tree of size 2 (i.e. two terminal nodes).

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

Answer:
The plot of the resulting pruned tree with two terminal nodes is shown below:



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

Answer:
The confusion matrix for the training error for the unpruned tree is:

		Actual "Purchase"	
		CH	**MM**
Predicted "Purchase"	**CH**	463	101
	MM	31	205

With training error rate:

$$\frac{(101+31)}{(463+31+101+205)} = 0.165$$

The confusion matrix for the training error for the pruned tree is:

Predicted "Purchase"		Actual "Purchase" CH	Actual "Purchase" MM
	CH	399	51
	MM	95	255

With training error rate:

$$\frac{(95+51)}{(399+95+51+255)} = 0.1825$$

The training error rate for the unpruned tree was 16.5%, with the training error rate for the pruned tree being 18.25%, making the training error rate for the pruned tree higher.

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

Answer:
The confusion matrix for the testing error for the unpruned tree is:

Predicted "Purchase"		Actual "Purchase" CH	Actual "Purchase" MM
	CH	147	49
	MM	12	62

With testing error rate:

$$\frac{(12+49)}{(147+49+12+62)} \approx 0.2259$$

The confusion matrix for the testing error for the pruned tree is:

Predicted "Purchase"		Actual "Purchase" CH	Actual "Purchase" MM
	CH	119	30
	MM	40	81

With testing error rate:

$$\frac{(40+30)}{(119+30+40+81)} \approx 0.2593$$

The testing error rate for the unpruned tree was 22.59%, with the testing error rate for the pruned tree being 25.93%, making the testing error rate for the pruned tree higher.

R-code:
```
################
## Problem #5 ##
################
set.seed(1)
library(tree)
library(randomForest)
library(ISLR)
data(OJ)
summary(OJ)
OJ$Purchase <- as.factor(OJ$Purchase)

############
## Part A ##
############
train_id <- sample(1:nrow(OJ), 800)
train_df <- OJ[train_id,]
test_df <- OJ[-train_id,]

############
## Part B ##
############
tree.OJ <- tree(Purchase~., data=train_df)
summary(tree.OJ)
yhat_train <- predict(tree.OJ, newdata=train_df, type="class")
table(yhat_train, train_df$Purchase)

############
## Part C ##
############
tree.OJ

############
## Part D ##
############
plot(tree.OJ)
text(tree.OJ, pretty=0)

############
## Part E ##
############
yhat_test <- predict(tree.OJ, newdata=test_df, type="class")
table(yhat_test, test_df$Purchase)

############
## Part F ##
############
cv.OJ <- cv.tree(tree.OJ, FUN=prune.misclass)
cv.OJ

############
## Part G ##
############
plot(cv.OJ$size, cv.OJ$dev, type="b")

############
## Part I ##
############
prune.OJ <- prune.tree(tree.OJ, best=2)
plot(prune.OJ)
text(prune.OJ, pretty=0)

############
## Part J ##
############
```

```
yhat_full_training <- predict(tree.OJ, newdata=train_df, type="class")
table(yhat_full_training, train_df$Purchase)

yhat_pruned_training <- predict(prune.OJ, newdata=train_df, type="class")
table(yhat_pruned_training, train_df$Purchase)

############
## Part K ##
############
yhat_full_test <- predict(tree.OJ, newdata=test_df, type="class")
table(yhat_full_test, test_df$Purchase)
yhat_pruned_test <- predict(prune.OJ, newdata=test_df, type="class")
table(yhat_pruned_test, test_df$Purchase)
```