STAT 602 Homework 10

Yuchi Hu

March 28, 2019

1. Question 8.4.4 pg 332

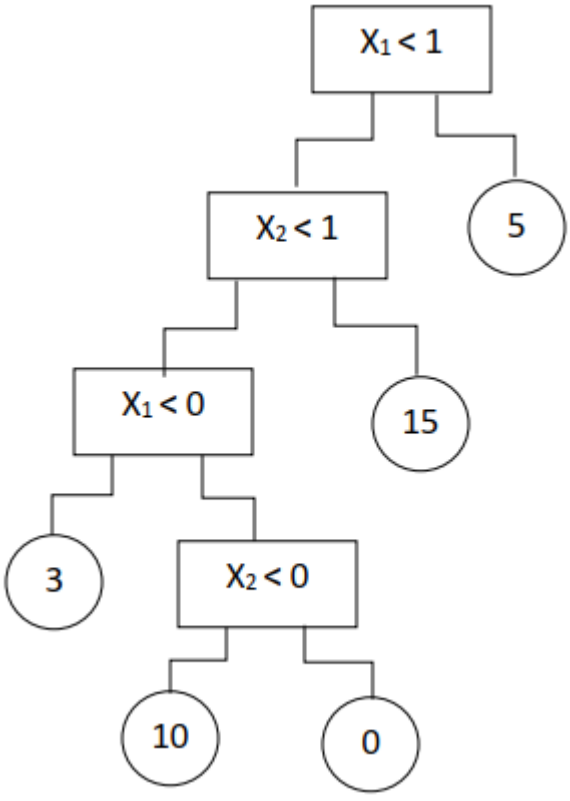
This question relates to the plots in Figure 8.12.

(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

The rectangles indicate splits, and the circles are terminal nodes. The number inside each node is the mean of the response for observations that fall there.

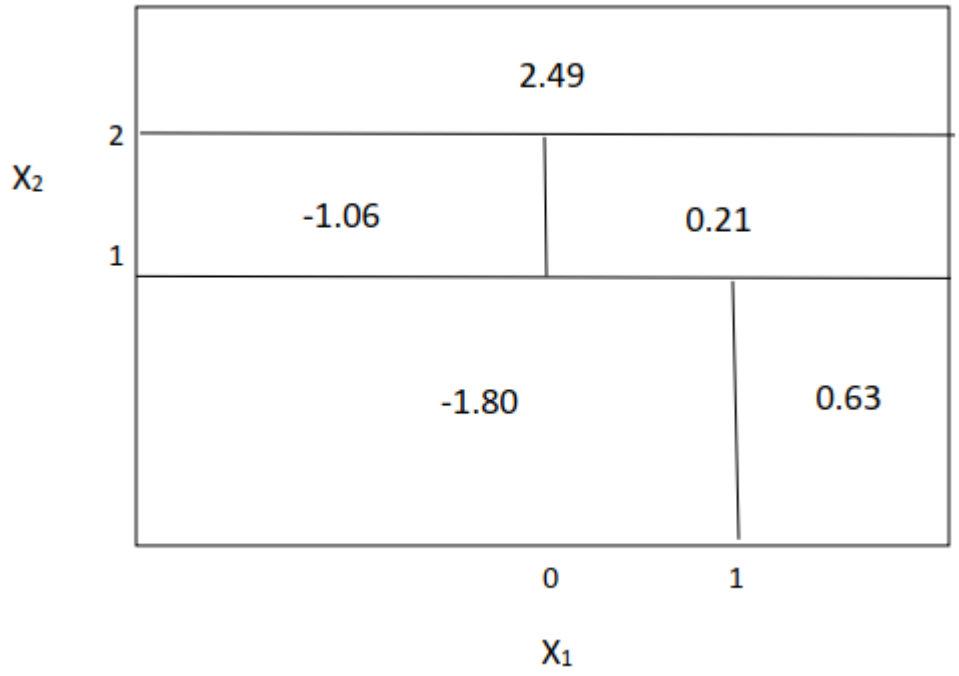


(b)

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

The predictor space is partitioned into five regions corresponding to the five terminal nodes. The number inside each region is the mean of the response for observations that fall there.



2. Question 8.4.8 pg 332

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.

Answer

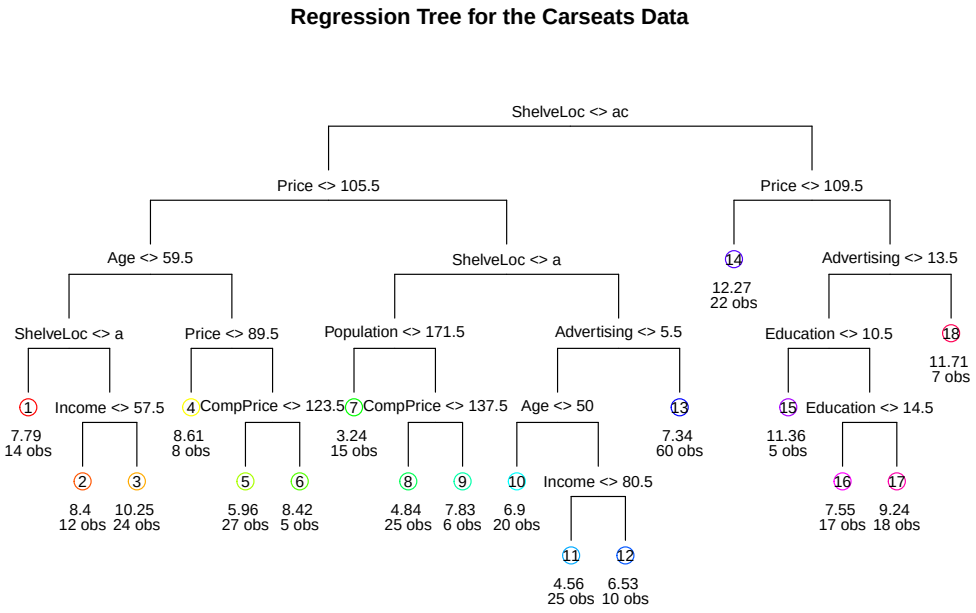
We use **sample()** with an 80:20 split to divide the **Carseats** data into a training set (320 observations) and a test set (80 observations).

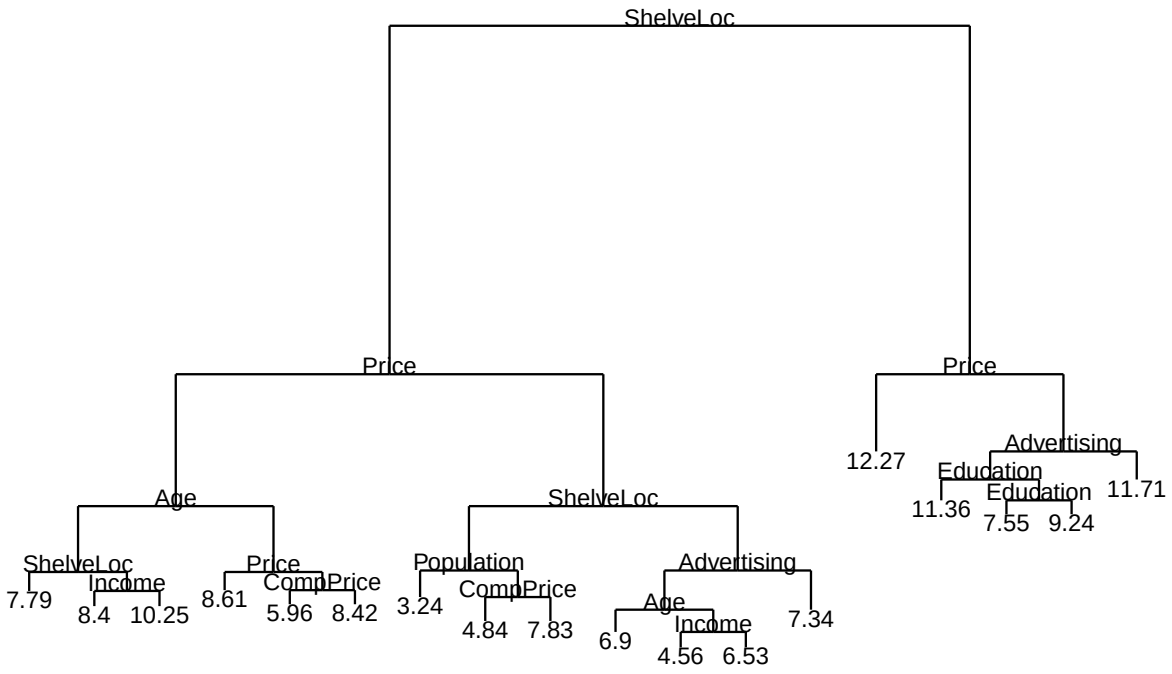
(b)

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

Answer

We fit a regression tree to the training set using **tree()** from the **tree** package. The response is *Sales*, and the predictors are the other variables. We plot the tree using **draw.tree()** from the **maptree** package. For the ggplot2 equivalent, we use the **ggdendro** package.





We see that there are 18 terminal nodes or leaves. The top split assigns observations with "Bad" or "Medium" *ShelveLoc* to the left branch (ac corresponds to the first and third factor levels of *ShelveLoc*), and then that group is further split on *Price* = 105.5 and so on. Observations with "Good" *ShelveLoc* are assigned to the right branch, and then further split on *Price* = 109.5 and so on. Thus, *ShelveLoc* (quality of the shelving location) is the most important factor in determining *Sales*. Overall, the tree segments the observations into 18 regions of predictor space.

```
## Test MSE: 5.313622
```

We obtain a test MSE of 5.314.

(c)

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

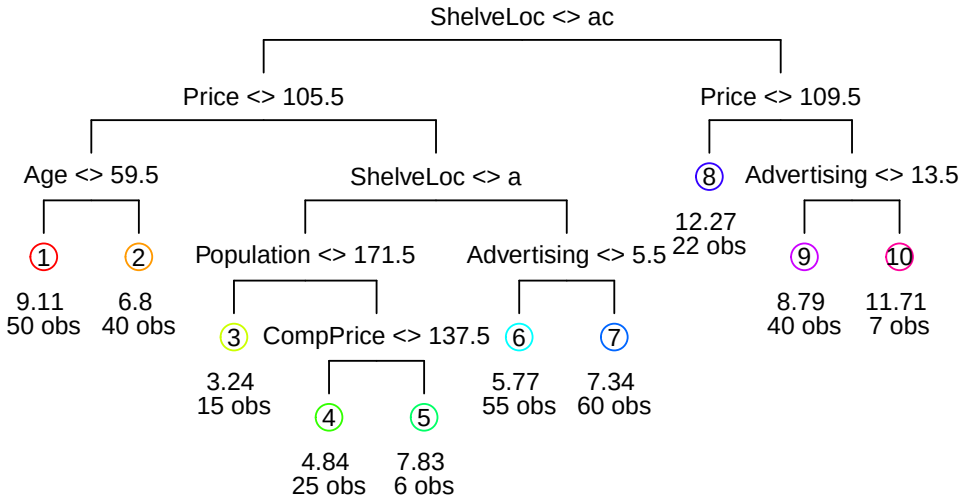
Answer

We use **cv.tree()** with 10-fold cross-validation to determine the optimal level of tree complexity. The optimal tree size corresponds to that with the lowest cross-validation error or deviance.

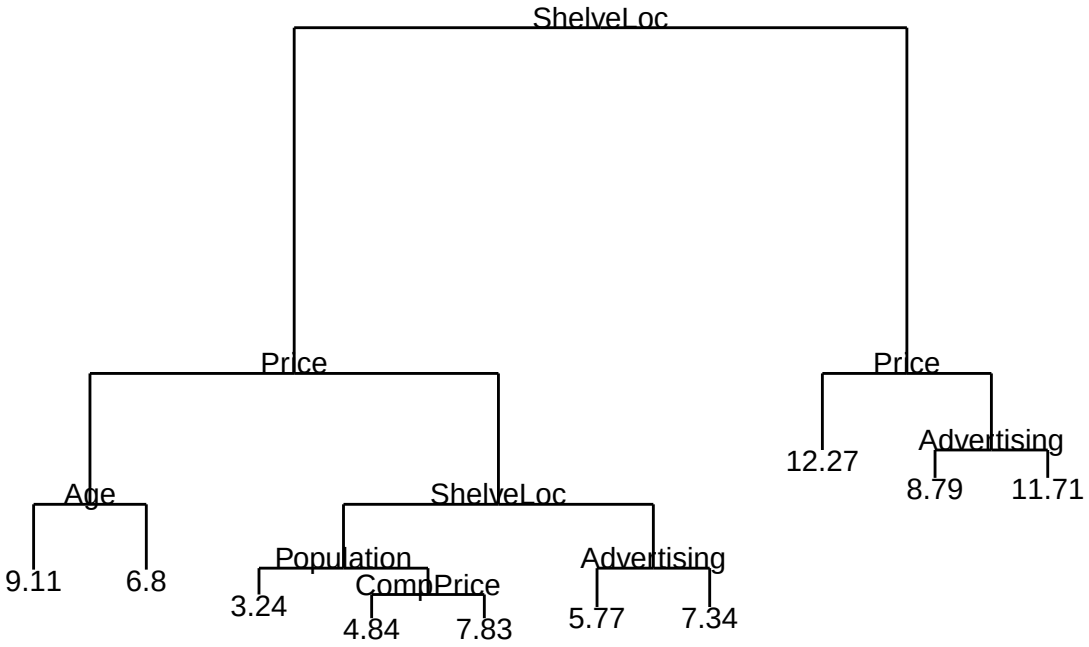
```
## Optimal size: 10
```

Thus, the optimal level of tree complexity is 10 terminal nodes. The resulting pruned tree is shown below:

Pruned Regression Tree for the Carseats Data



GGPLOT2: Pruned Regression Tree for the Carseats Data



```
## Test MSE: 5.826313
```

The test MSE is 5.826. Since the test MSE associated with the unpruned tree is 5.314, pruning the tree does not 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.

Answer

We use **randomForest()** from the **randomForest** package to perform bagging. Bagging is simply random forest with $m = p$, so **randomForest()** can be used to perform both. The argument mtry=10 indicates that all 10 predictors should be considered for each split of the tree. By default, 500 trees are grown.

```
## Test MSE: 3.343307
```

We obtain a test MSE of 3.343.

Next, we use **importance()** to view the importance of each variable.

```
##               %IncMSE IncNodePurity
## CompPrice   33.616773    258.763648
## Income      12.840362    129.436918
## Advertising 20.898163    186.548364
## Population   1.916791     88.471689
## Price       69.303801    655.116689
## ShelveLoc   76.095087    763.535790
## Age         26.169995    251.756604
## Education    4.852247     66.266026
## Urban       -2.243253     10.194433
## US           3.505185      8.547899
```

The first measure (%IncMSE) is the mean decrease of accuracy in predictions on the out of bag samples when a given variable is excluded from the model. The second measure (IncNodePurity) is the total decrease in node impurity that results from splits over that variable, averaged over all trees. We see that *ShelveLoc* and *Price* are by far the two most important variables.

(e)

Use random forests to analyze this 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

Similar to bagging, We use **randomForest()** to perform random forests. In random forests, only a random subset of the predictors is considered for each split; by default, **randomForest()** uses $m \approx p/3$ when building a random forest of regression trees. In this case, $m = 3$ since there are 10 predictors.

```
## Test MSE: 3.664894
```

We obtain a test MSE of 3.665. Since the test MSE associated with bagging is 3.343, random forest does not improve upon bagging in this case.

Next, we use **importance()** to view the importance of each variable.

```
##                %IncMSE IncNodePurity
## CompPrice   19.05258509     232.57957
## Income       6.87023239     182.54457
## Advertising 16.22552178     197.38918
## Population   0.03747007     156.73154
## Price       47.83887648     549.37534
## ShelveLoc   49.78541607     583.46746
## Age         19.52223922     282.07163
## Education    0.94823487     110.68676
## Urban       -2.08923223      19.73004
## US           3.84087343      27.63182
```

Again, we see that *ShelveLoc* and *Price* are by far the two most important variables.

3. Question 8.4.9 pg 334

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

We use **sample()** to divide the **OJ** data into a training set (800 observations) and a test set (270 observations).

(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

We fit a classification tree to the training set with *Purchase* as the response and the other variables as predictors. We use **summary()** to produce summary statistics about the tree.

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = train)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.771 = 610.6 / 792
## Misclassification error rate: 0.1662 = 133 / 800
```

We see that only two out of the 17 predictors were used in tree construction: *LoyalCH* and *PriceDiff*. The training error rate is 0.1662. The tree has 8 terminal nodes.

(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

We type in the name of the tree object in order to get a detailed text ouput.

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1073.00 CH ( 0.60500 0.39500 )
##    2) LoyalCH < 0.50395 348  411.90 MM ( 0.27874 0.72126 )
##      4) LoyalCH < 0.276142 163  121.40 MM ( 0.12270 0.87730 )
##        8) LoyalCH < 0.051325 60   10.17 MM ( 0.01667 0.98333 ) *
##        9) LoyalCH > 0.051325 103   98.49 MM ( 0.18447 0.81553 ) *
##      5) LoyalCH > 0.276142 185  251.20 MM ( 0.41622 0.58378 )
##       10) PriceDiff < 0.05 76   72.61 MM ( 0.18421 0.81579 ) *
##       11) PriceDiff > 0.05 109  148.40 CH ( 0.57798 0.42202 ) *
##    3) LoyalCH > 0.50395 452  372.30 CH ( 0.85619 0.14381 )
##      6) LoyalCH < 0.764572 191  223.70 CH ( 0.72775 0.27225 )
##       12) PriceDiff < 0.265 114  154.50 CH ( 0.58772 0.41228 )
##         24) PriceDiff < -0.165 34   42.81 MM ( 0.32353 0.67647 ) *
##         25) PriceDiff > -0.165 80   97.74 CH ( 0.70000 0.30000 ) *
##       13) PriceDiff > 0.265 77   37.01 CH ( 0.93506 0.06494 ) *
##      7) LoyalCH > 0.764572 261  103.30 CH ( 0.95019 0.04981 ) *
```

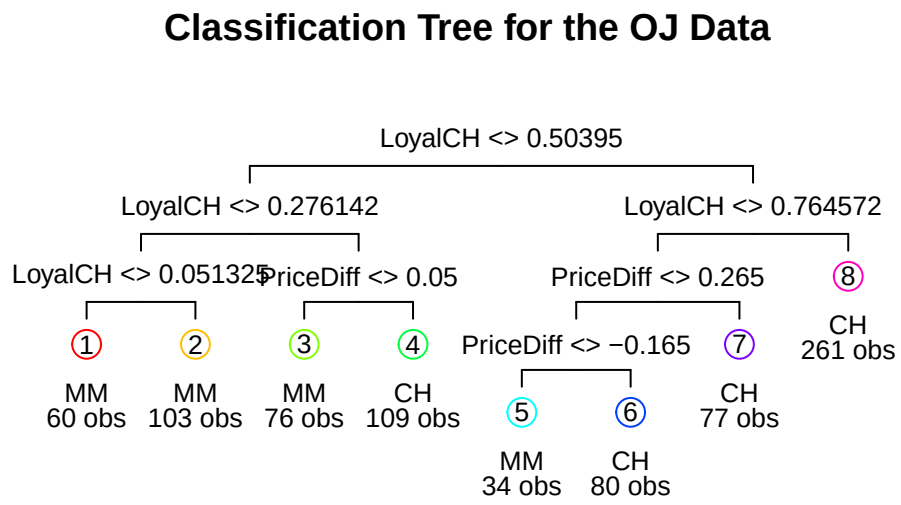
Branches that lead to terminal nodes are indicated using asterisks. If we look at node 8, we see that the split criterion is $LoyalCH < 0.0513$. The number of observations in that branch is 60, the deviance is 10.17, and the overall prediction for the branch is "MM". Finally, the fraction of observations in that branch that take on values of "CH" and "MM" are 0.0167 and 0.983, respectively.

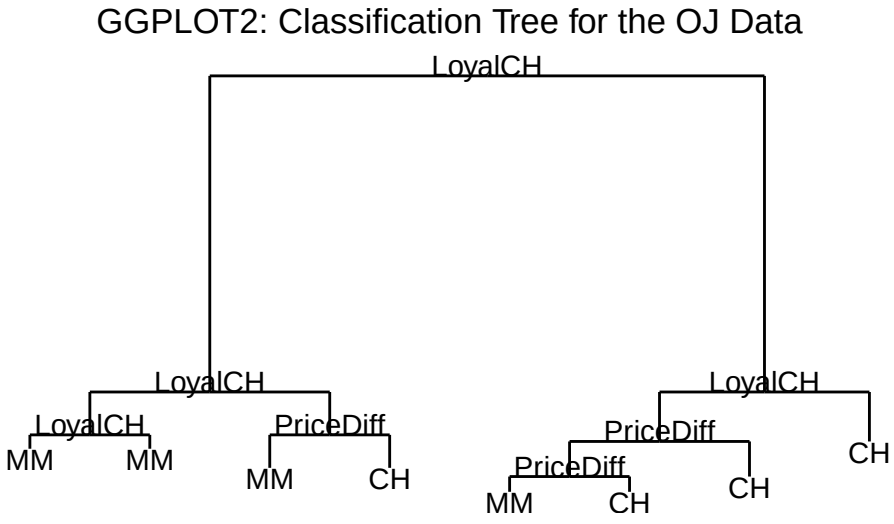
(d)

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

Answer

The plot of the tree is shown below:





We see that there are 8 terminal nodes or leaves. Only *LoyalCH* and *PriceDiff* were used in tree construction. The top three splits occur on *LoyalCH*; thus, *LoyalCH* is the most important factor in determining *Purchase*. Overall, the tree segments the observations into 8 regions of predictor space.

(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

We predict the response on the test data and produce a confusion matrix comparing the test labels to the predicted test labels.

```
##                True class
## Predicted class  CH  MM
##              CH 153  26
##              MM  16  75
```

```
## Test error rate: 0.1555556
```

The test error rate is 0.156.

(f)

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

Answer

We apply **cv.tree()** to the training set in order to determine the optimal tree size. By default, 10-fold CV is performed.

```
## $size
## [1] 8 7 4 2 1
##
## $dev
## [1] 159 159 170 170 316
##
## $k
## [1]  -Inf   0.0   4.0   8.5 154.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

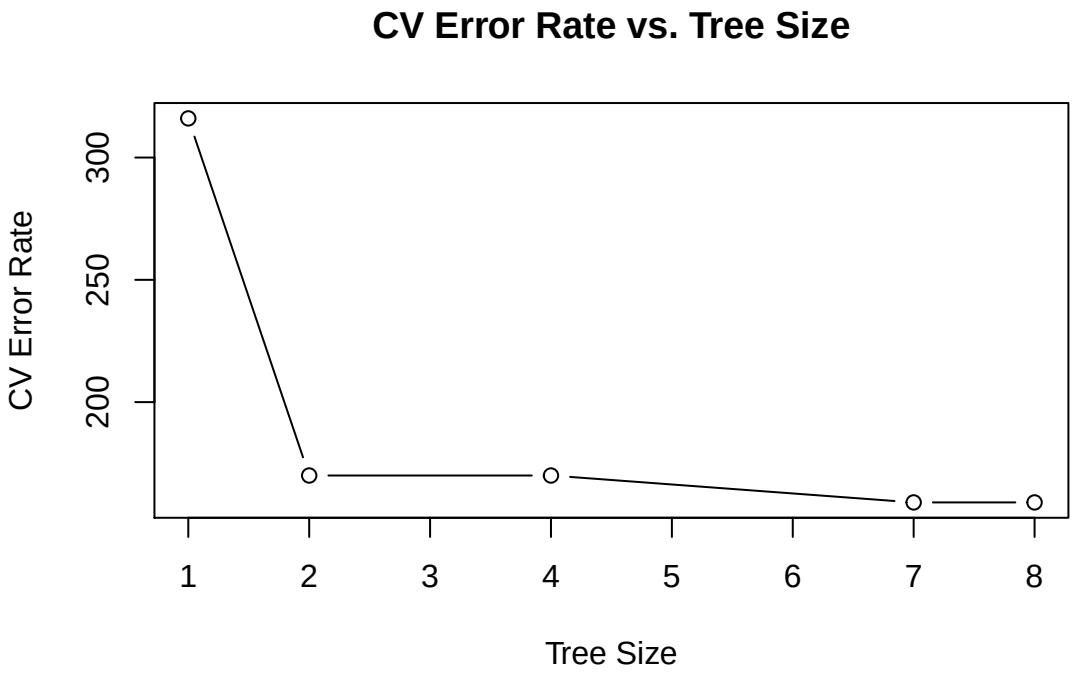
"dev" corresponds to the cross-validation error rate. We see that the trees with 8 and 7 terminal nodes are tied for the lowest CV error rate, with 159 CV errors each. We choose the smaller of the two (7 terminal nodes) as the optimal tree size.

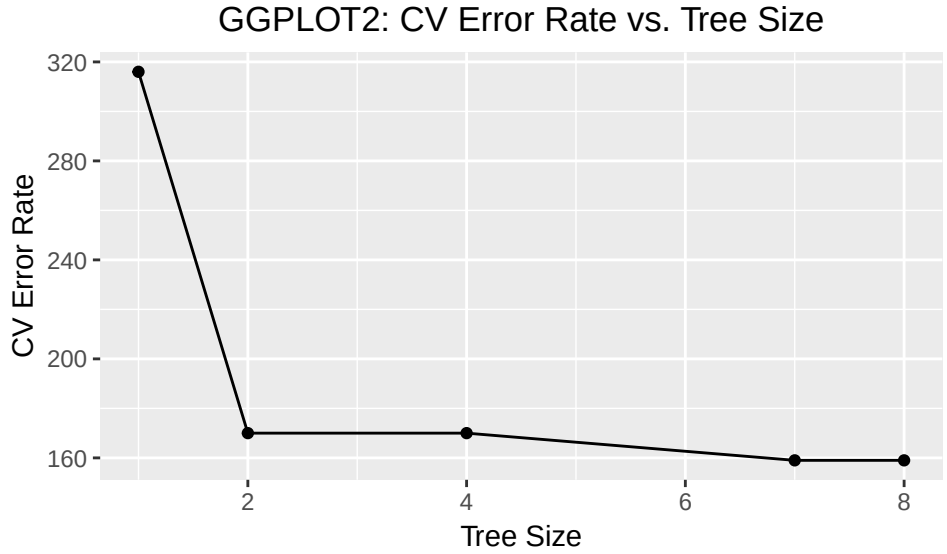
(g)

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

Answer

We 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?

Answer

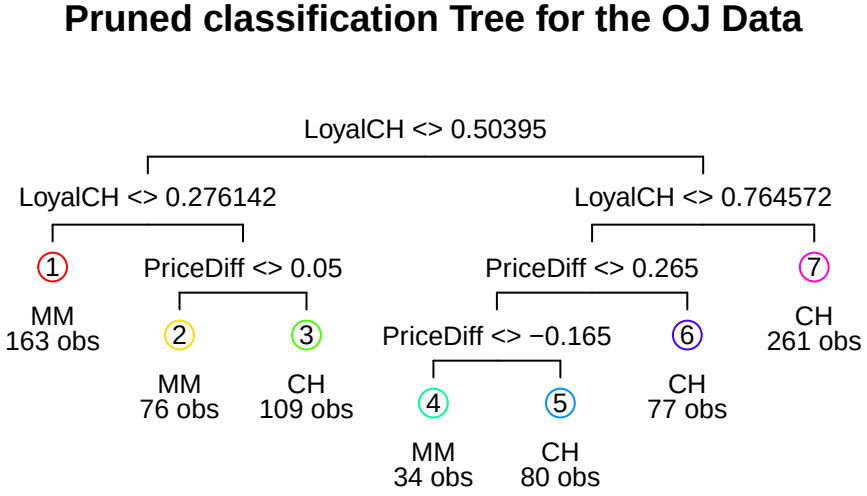
The tree with 7 or 8 terminal nodes corresponds to the lowest cross-validated classification error rate.

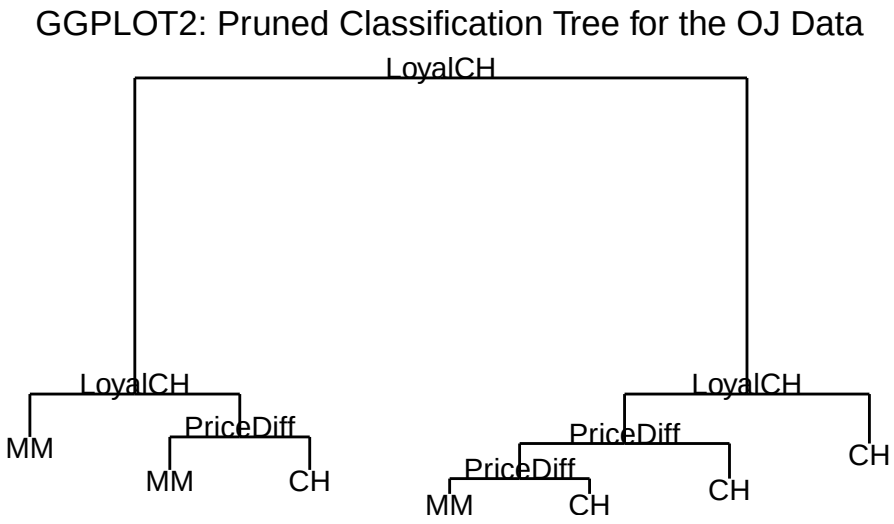
(i)

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.

Answer

The plot of the pruned tree with 7 terminal nodes is shown below:





(j)

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

Answer

The summary of the unpruned tree is shown below:

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = train)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.771 = 610.6 / 792
## Misclassification error rate: 0.1662 = 133 / 800
```

The training error rate of the unpruned tree is 0.1662.

The summary of the pruned tree is shown below:

```
##
## Classification tree:
## snip.tree(tree = OJ.tree, nodes = 4L)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  7
## Residual mean deviance:  0.786 = 623.3 / 793
## Misclassification error rate: 0.1662 = 133 / 800
```

The training error rate of the pruned tree is 0.1662. Thus, the training error rates of the pruned and unpruned trees are equal.

(k)

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

Answer

The confusion matrices of the unpruned and pruned trees are shown below:

```
##                 True class
## Predicted class  CH  MM
##              CH 153  26
##              MM  16  75
```

```
## Test error rate of unpruned tree: 0.1555556
```

```
##                 True class
## Predicted class  CH  MM
##              CH 153  26
##              MM  16  75
```

```
## Test error rate of pruned tree: 0.1555556
```

We see that the confusion matrices and test error rates of the unpruned (8 terminal nodes) and pruned (7 terminal nodes) trees are identical. Both test error rates are 0.156.

4. Question 8.4.10 pg 334

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.

Answer

We remove the observations for whom the salary information is unknown, and then log-transform the salaries. There are 59 observations with unknown salary information.

(b)

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

Answer

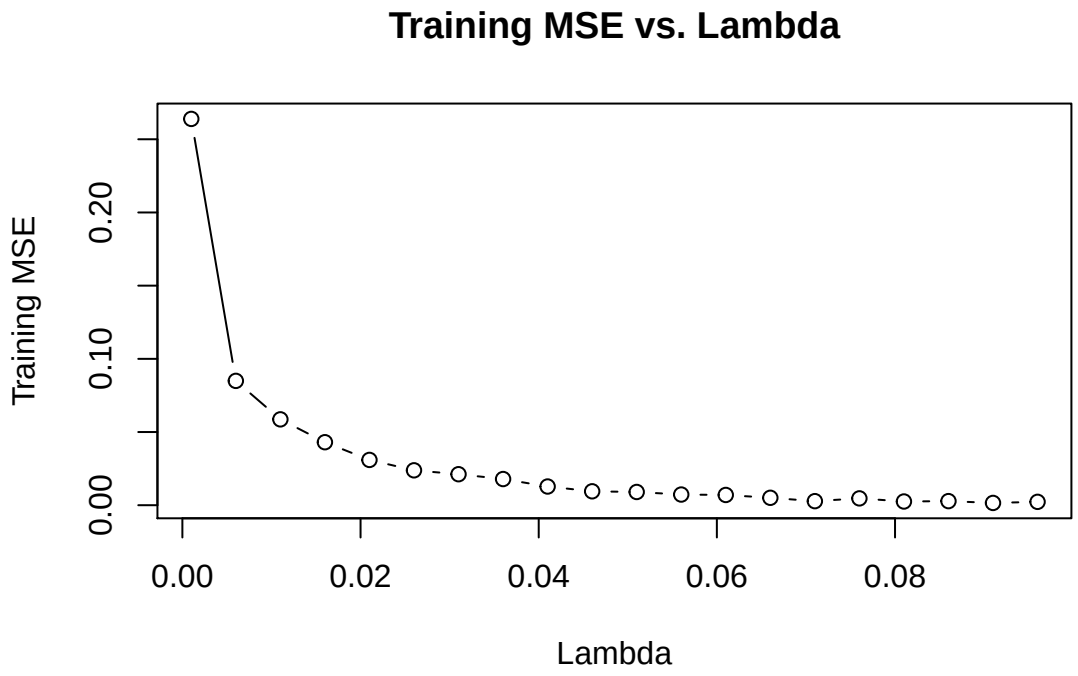
We create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

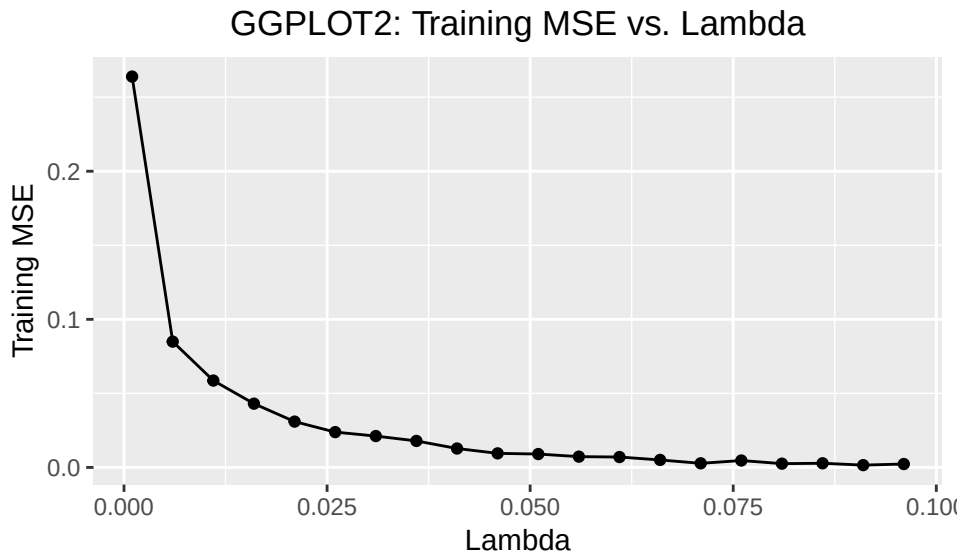
(c)

Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

Answer

We perform boosting on the training set with 1,000 trees for shrinkage values from 0.001 to 0.1, incrementing by 0.005. Then, we produce a plot with shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.



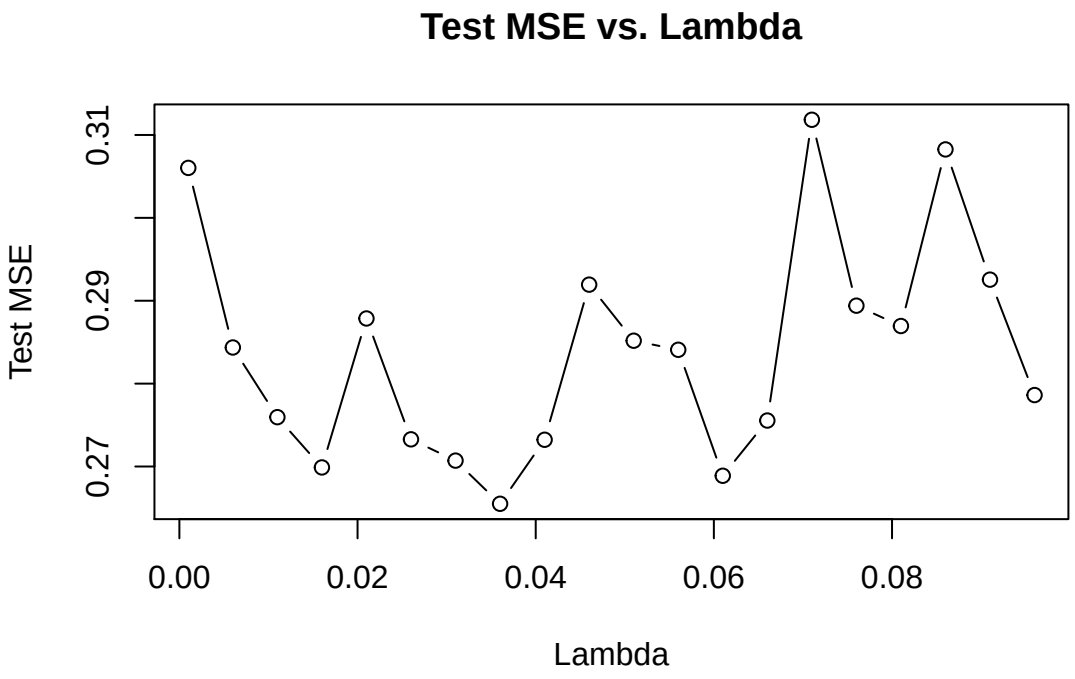


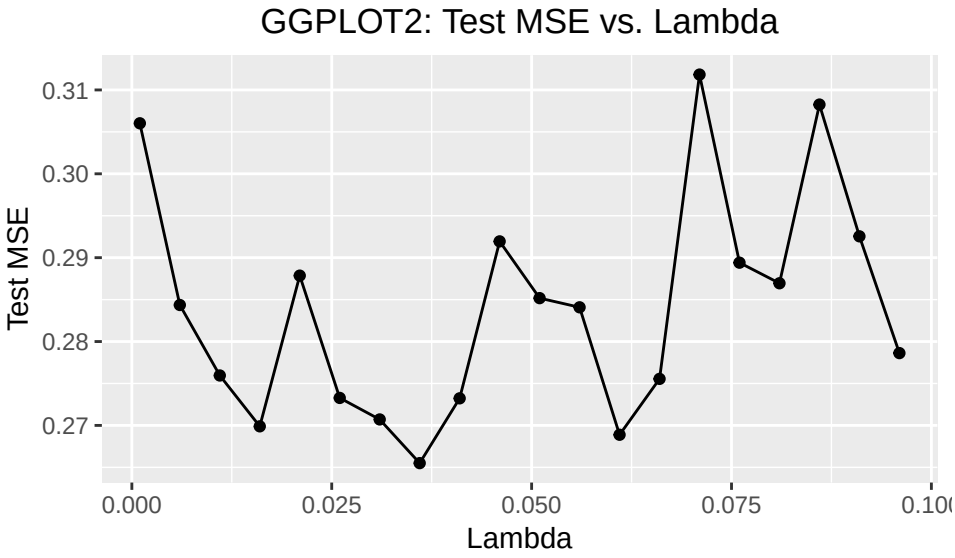
(d)

Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

Answer

We produce a plot with shrinkage values on the x-axis and the corresponding test set MSE on the y-axis. The shrinkage values range from 0.001 to 0.1, incrementing by 0.005 (same as part(c)).





(e)

Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in Chapters 3 and 6.

Answer

First, we need to find the lowest test MSE of boosting and the associated λ.

```
## Lowest test MSE of boosting: 0.2655034
```

```
## Best lambda for boosting: 0.036
```

The lowest test MSE of boosting is 0.266, and the associated λ is 0.036.

Next, we calculate the test MSE of linear regression (chapter 3) and ridge regression (chapter 6).

```
## Test MSE of linear regression: 0.4917959
```

```
## Test MSE of ridge regression: 0.4531049
```

The test MSE's of linear and ridge regressions are 0.492 and 0.453, respectively. Thus, the test MSE of boosting (0.266) is lower than the test MSE's of linear and ridge regressions.

(f)

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

Answer

We perform boosting on the training set with 1,000 trees and the shrinkage value (0.036) associated with the lowest test MSE (see part (e)). The boosted model summary outputs relative influence statstics:

```
##                 var    rel.inf
## CAtBat       CAtBat 24.9635612
## CRuns         CRuns  8.7207780
## CWalks       CWalks  8.5774407
## CRBI           CRBI  8.2180907
## PutOuts     PutOuts  6.1463234
## Walks         Walks  5.4399444
## Years         Years  5.2985551
## Hits           Hits  4.2401197
## CHits         CHits  4.2211724
## AtBat         AtBat  4.0169454
## CHmRun       CHmRun  4.0015007
## Assists     Assists  3.9072781
## RBI             RBI  3.2382176
## HmRun         HmRun  2.7396689
## Runs           Runs  2.3929729
## Errors       Errors  2.2914449
## NewLeague NewLeague  0.5889925
## Division   Division  0.5817370
## League       League  0.4152564
```

We see that *CAtBat* (number of career at bats) is by far the most important variable in predicting *Salary*.

(g)

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

Answer

We apply bagging to the training set.

```
## Test MSE: 0.2314244
```

The test MSE is 0.231. This is even better than the test MSE of boosting (0.266).

5.

In the past couple of homework assignments you have used different classification methods to analyze the dataset you chose. For this homework, use tree-based classification methods (tree,bagging, randomforest, boosting) to model your data. Find the test error using any/all of methods (VSA, K-fold CV). Compare the results you obtained with the result from previous homework. Did the results improve? (Use the table you previously made to compare results)

Answer

Background Review

I chose the Adult dataset from the UC Irvine Machine Learning Repository. This dataset was extracted from the 1994 Census Bureau database. The task is to classify a person into one of two income groups (>50K or <=50K) based on the predictors using various classification methods and to compare these methods' performances. The full dataset is not provided, but rather separate training and test sets are. There are a total of 48,842 observations (32,561 observations in the training set and 16,281 observations in the test set, i.e. 2/3 training and 1/3 test). There are 15 total variables (response + 14 predictors).

The following numeric variables were chosen as the final predictors through variable selection: *age*, *education_num*, *capital_gain*, *capital_loss*, and *hours_per_week*. To compare the classification methods, we used the validation set approach (VSA), 5-fold and 10-fold cross-validations to estimate the test error for the models.

Summary of Previous Classification Methods

Table 1: Test Error Rate Comparison for the Classification Methods

Method	VSA	5-Fold CV	10-Fold CV
Logistic Regression	0.1872	0.1303	0.1303
LDA	0.1981	0.1986	0.1986
QDA	0.2036	0.2039	0.2038
KNN (K=sqrt(n))	0.1723	0.1698	0.1683
MclustDA	0.2672	0.2526	0.2504
MclustDA with EDDA	0.2037	0.2039	0.204

Table 1 shows the VSA, 5-fold, and 10-fold CV test errors associated with each model. Note that I omitted the **rpart** classification tree since we will be using the **tree** package later.

5-fold or 10-fold CV are more reliable than VSA since their test error rate estimates suffer neither from high bias nor high variance, while the estimate using VSA is highly variable. Logistic regression performed the best since it has the lowest 5-fold and 10-fold CV test errors, although KNN did have the best VSA test error. On the other hand, MclustDA by far performed the worst for all three cross-validation methods.

Tree-based Classification Methods

We now use tree-based classification methods to model the data.

Classification Tree

We fit a classification tree to the training set using **tree()** from the **tree** package.

```
## VSA Test Error: 0.1842639
```

```
## 5-Fold CV Test Error: 0.1911883
```

```
## 10-Fold CV Test Error: 0.1898983
```

Bagging

We use **randomForest()** from the **randomForest** package to perform bagging. Bagging is simply random forest with $m = p$, so **randomForest()** can be used to perform both. The argument mtry=5 indicates that all 5 predictors should be considered for each split of the tree. By default, 500 trees are grown.

```
## VSA Test Error: 0.1760334
```

```
## 5-Fold CV Test Error: 0.1742149
```

```
## 10-Fold CV Test Error: 0.173007
```

Random Forests

Similar to bagging, We use **randomForest()** to perform random forests. In random forests, only a random subset of the predictors is considered for each split; by default, **randomForest()** uses $m \approx \sqrt{p}$ when building a random forest of classification trees. In this case, $m = 2$ since there are 5 predictors.

```
## VSA Test Error: 0.1606781
```

```
## 5-Fold CV Test Error: 0.1609682
```

```
## 10-Fold CV Test Error: 0.1600673
```

Boosting

We use **gbm()** from the **gbm** package to perform boosting on the training set with 500 trees and the default shrinkage value: 0.1. First, the response (*income*) must be converted to 0-1 outcomes in order to fit boosted classification trees. In this case, incomes of <=50K are converted to 0's, and incomes of >50K are converted to 1's.

```
## VSA Test Error: 0.1563786
```

```
## 5-Fold CV Test Error: 0.1569348
```

```
## 10-Fold CV Test Error: 0.1563819
```

Summary of Classification Methods Including Tree-Based Methods

Table 2: Test Error Rate Comparison for the Classification Methods

Method	VSA	5-Fold CV	10-Fold CV
Logistic Regression	0.1872	0.1303	0.1303
LDA	0.1981	0.1986	0.1986
QDA	0.2036	0.2039	0.2038
KNN (K=sqrt(n))	0.1723	0.1698	0.1683
MclustDA	0.2672	0.2526	0.2504
MclustDA with EDDA	0.2037	0.2039	0.204
Classification Tree	0.1843	0.1912	0.1899
Bagging	0.176	0.1742	0.173
Random Forests	0.1607	0.161	0.1601
Boosting	0.1564	0.1569	0.1564

Table 2 shows the VSA, 5-fold, and 10-fold CV test errors associated with each model including the tree-based methods.

We see that bagging, random forests, and boosting sequentially improve upon the single classification tree; however, logistic regression is still the best in terms of 5-fold and 10-fold CV test errors.