PREDICT422-Assignment4 Classification

Artur Mrozowski

May 21, 2017

Chapter 4 ISLR 11. In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set. (a) Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.2.3
```

```
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0
##
##      weight      acceleration        year           origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000
##  Median :2804   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :1.577
##  3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
##
##                  name
##  amc matador       :  5
##  ford pinto        :  5
##  toyota corolla    :  5
##  amc gremlin       :  4
##  amc hornet        :  4
##  chevrolet chevette:  4
##  (Other)           :365
```

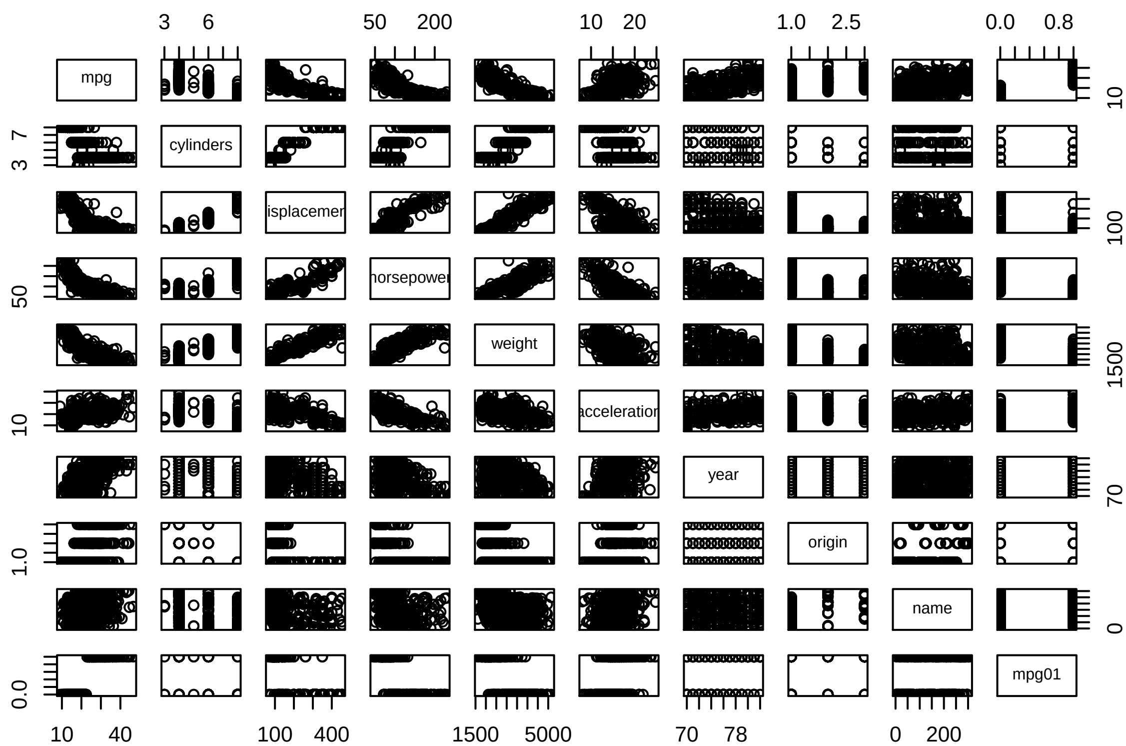
(b) Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

```
attach(Auto)
mpg01 = rep(0, length(mpg))
mpg01[mpg > median(mpg)] = 1
Auto = data.frame(Auto, mpg01)
```

```r
cor(Auto[, -9])
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
## mpg01         0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
##              acceleration       year     origin      mpg01
## mpg             0.4233285  0.5805410  0.5652088  0.8369392
## cylinders      -0.5046834 -0.3456474 -0.5689316 -0.7591939
## displacement   -0.5438005 -0.3698552 -0.6145351 -0.7534766
## horsepower     -0.6891955 -0.4163615 -0.4551715 -0.6670526
## weight         -0.4168392 -0.3091199 -0.5850054 -0.7577566
## acceleration    1.0000000  0.2903161  0.2127458  0.3468215
## year            0.2903161  1.0000000  0.1815277  0.4299042
## origin          0.2127458  0.1815277  1.0000000  0.5136984
## mpg01           0.3468215  0.4299042  0.5136984  1.0000000
```

```r
pairs(Auto)  # doesn't work well since mpg01 is 0 or 1
```



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

```r
train = (year%%2 == 0)  # if the year is even
test = !train
Auto.train = Auto[train, ]
Auto.test = Auto[test, ]
mpg01.test = mpg01[test]
```

(d) Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```r
# LDA
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.2.2
```

```r
lda.fit = lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,
    subset = train)
lda.pred = predict(lda.fit, Auto.test)
mean(lda.pred$class != mpg01.test)
```

```
## [1] 0.1263736
```

12.6% test error rate.

(e) Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```r
# QDA
qda.fit = qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,
    subset = train)
qda.pred = predict(qda.fit, Auto.test)
mean(qda.pred$class != mpg01.test)
```

```
## [1] 0.1318681
```

13.2% test error rate.

(f) Perform logistic regression on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```r
# Logistic regression
glm.fit = glm(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,
    family = binomial, subset = train)
glm.probs = predict(glm.fit, Auto.test, type = "response")
glm.pred = rep(0, length(glm.probs))
glm.pred[glm.probs > 0.5] = 1
mean(glm.pred != mpg01.test)
```

```
## [1] 0.1208791
```

12.1% test error rate.

(g) Perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

```
library(class)
```

```
## Warning: package 'class' was built under R version 3.2.2
```

```
train.X = cbind(cylinders, weight, displacement, horsepower)[train, ]
test.X = cbind(cylinders, weight, displacement, horsepower)[test, ]
train.mpg01 = mpg01[train]
set.seed(1)
# KNN(k=1)
knn.pred = knn(train.X, test.X, train.mpg01, k = 1)
mean(knn.pred != mpg01.test)
```

```
## [1] 0.1538462
```

```
# KNN(k=10)
knn.pred = knn(train.X, test.X, train.mpg01, k = 10)
mean(knn.pred != mpg01.test)
```

```
## [1] 0.1648352
```

```
# KNN(k=100)
knn.pred = knn(train.X, test.X, train.mpg01, k = 100)
mean(knn.pred != mpg01.test)
```

```
## [1] 0.1428571
```

k=1, 15.4% test error rate. k=10, 16.5% test error rate. k=100, 14.3% test error rate. K of 100 seems to perform the best. 100 nearest neighbors.

13. Using the Boston data set, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, and KNN models using various subsets of the predictors. Describe your findings.

```
library(MASS)
summary(Boston)
```

```
##       crim                zn             indus            chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
```

```
##       nox               rm              age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00
```

```r
attach(Boston)
crime01 = rep(0, length(crim))
crime01[crim > median(crim)] = 1
Boston = data.frame(Boston, crime01)

train = 1:(dim(Boston)[1]/2)
test = (dim(Boston)[1]/2 + 1):dim(Boston)[1]
Boston.train = Boston[train, ]
Boston.test = Boston[test, ]
crime01.test = crime01[test]
```

```r
# logistic regression
glm.fit = glm(crime01 ~ . - crime01 - crim, data = Boston, family = binomial,
    subset = train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
glm.probs = predict(glm.fit, Boston.test, type = "response")
glm.pred = rep(0, length(glm.probs))
glm.pred[glm.probs > 0.5] = 1
mean(glm.pred != crime01.test)
```

```
## [1] 0.1818182
```

18.2% test error rate.

```r
glm.fit = glm(crime01 ~ . - crime01 - crim - chas - tax, data = Boston, family = binomial,
    subset = train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
glm.probs = predict(glm.fit, Boston.test, type = "response")
glm.pred = rep(0, length(glm.probs))
glm.pred[glm.probs > 0.5] = 1
mean(glm.pred != crime01.test)
```

```
## [1] 0.1857708
```

18.6% test error rate.

```r
# LDA
lda.fit = lda(crime01 ~ . - crime01 - crim, data = Boston, subset = train)
lda.pred = predict(lda.fit, Boston.test)
mean(lda.pred$class != crime01.test)
```

```
## [1] 0.1343874
```

13.4% test error rate.

```r
lda.fit = lda(crime01 ~ . - crime01 - crim - chas - tax, data = Boston, subset = train)
lda.pred = predict(lda.fit, Boston.test)
mean(lda.pred$class != crime01.test)
```

```
## [1] 0.1225296
```

12.3% test error rate.

```r
lda.fit = lda(crime01 ~ . - crime01 - crim - chas - tax - lstat - indus - age,
    data = Boston, subset = train)
lda.pred = predict(lda.fit, Boston.test)
mean(lda.pred$class != crime01.test)
```

```
## [1] 0.1185771
```

11.9% test error rate.

```r
# KNN
library(class)
train.X = cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black,
    lstat, medv)[train, ]
test.X = cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black,
    lstat, medv)[test, ]
train.crime01 = crime01[train]
set.seed(1)
# KNN(k=1)
knn.pred = knn(train.X, test.X, train.crime01, k = 1)
mean(knn.pred != crime01.test)
```

```
## [1] 0.458498
```

45.8% test error rate.

```r
# KNN(k=10)
knn.pred = knn(train.X, test.X, train.crime01, k = 10)
mean(knn.pred != crime01.test)
```

```
## [1] 0.1185771
```

11.1% test error rate.

```r
# KNN(k=100)
knn.pred = knn(train.X, test.X, train.crime01, k = 100)
mean(knn.pred != crime01.test)
```

```
## [1] 0.4901186
```

49.0% test error rate.

```r
# KNN(k=10) with subset of variables
train.X = cbind(zn, nox, rm, dis, rad, ptratio, black, medv)[train, ]
test.X = cbind(zn, nox, rm, dis, rad, ptratio, black, medv)[test, ]
knn.pred = knn(train.X, test.X, train.crime01, k = 10)
mean(knn.pred != crime01.test)
```

```
## [1] 0.2727273
```

28.5% test error rate.

Chapter 8 ISLR 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.

```r
library(ISLR)
attach(Carseats)
set.seed(1)

train = sample(dim(Carseats)[1], dim(Carseats)[1]/2)
Carseats.train = Carseats[train, ]
Carseats.test = Carseats[-train, ]
```

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

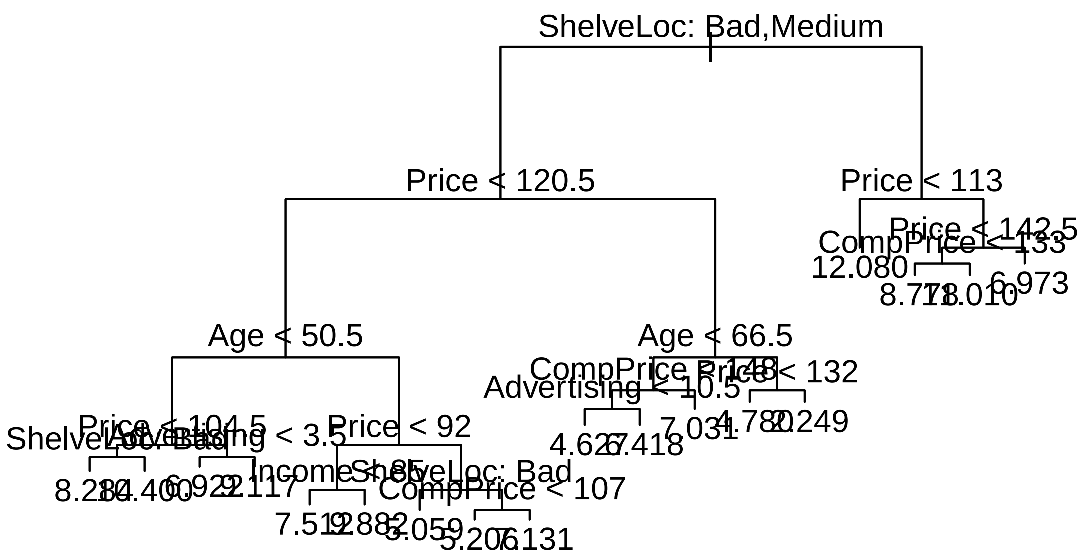
```r
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.2.5
```

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

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Advertising" "Income"
## [6] "CompPrice"
## Number of terminal nodes:  18
## Residual mean deviance:  2.36 = 429.5 / 182
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.2570 -1.0360  0.1024  0.0000  0.9301  3.9130
```

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



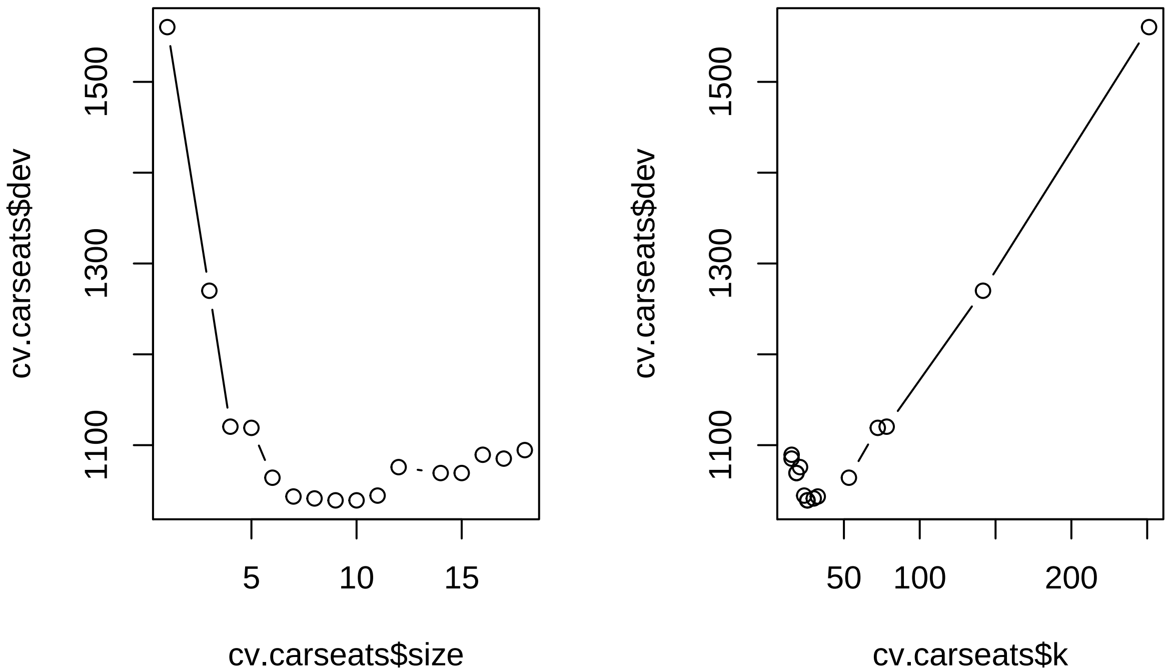

```r
pred.carseats = predict(tree.carseats, Carseats.test)
mean((Carseats.test$Sales - pred.carseats)^2)
```

```
## [1] 4.148897
```

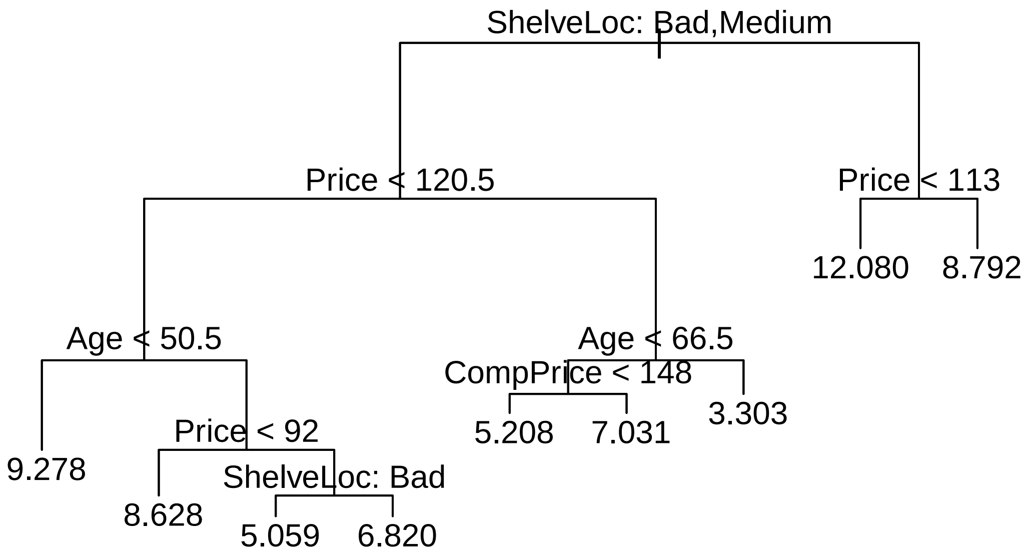
The test MSE is about 4.15

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




```
# Best size = 9
pruned.carseats = prune.tree(tree.carseats, best = 9)
par(mfrow = c(1, 1))
plot(pruned.carseats)
text(pruned.carseats, pretty = 0)
```




```r
pred.pruned = predict(pruned.carseats, Carseats.test)
mean((Carseats.test$Sales - pred.pruned)^2)
```

```
## [1] 4.993124
```

Pruning the tree in this case increases the test MSE to 4.99

(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.

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.3
```

```
## randomForest 4.6-12
```

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

```r
bag.carseats = randomForest(Sales ~ ., data = Carseats.train, mtry = 10, ntree = 500,
    importance = T)
bag.pred = predict(bag.carseats, Carseats.test)
mean((Carseats.test$Sales - bag.pred)^2)
```

```
## [1] 2.604369
```

```r
importance(bag.carseats)
```

```
##                %IncMSE IncNodePurity
## CompPrice   14.4124562    133.731797
## Income       6.5147532     74.346961
## Advertising 15.7607104    117.822651
## Population   0.6031237     60.227867
## Price       57.8206926    514.802084
## ShelveLoc   43.0486065    319.117972
## Age         19.8789659    192.880596
## Education    2.9319161     39.490093
## Urban       -3.1300102      8.695529
## US           7.6298722     15.723975
```

Bagging improves the test MSE to 2.58 We also see that Price, ShelveLoc and Age are three most important predictors of Sale.

(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.

```r
rf.carseats = randomForest(Sales ~ ., data = Carseats.train, mtry = 5, ntree = 500,
    importance = T)
rf.pred = predict(rf.carseats, Carseats.test)
mean((Carseats.test$Sales - rf.pred)^2)
```

```
## [1] 2.802383
```

```r
importance(rf.carseats)
```

```
##                %IncMSE IncNodePurity
## CompPrice   12.0259791     124.81403
## Income       5.5542673     106.15418
## Advertising 12.0466048     136.15204
## Population   0.3136897      81.68162
## Price       45.9639857     457.15711
## ShelveLoc   36.2789679     271.76488
## Age         20.8537727     196.72182
## Education    2.9005332      54.16980
## Urban       -0.6888196      11.86848
## US           6.9739759      23.64075
```

In this case, random forest worsens the MSE on test set to 2.87 Changing mm varies test MSE between 2.6 to 33. We again see that PricePrice, ShelveLocS and AgeA are three most important predictors of Sale.

9. 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.

```r
library(ISLR)
attach(OJ)
set.seed(1013)

train = sample(dim(OJ)[1], 800)
OJ.train = OJ[train, ]
OJ.test = OJ[-train, ]
```

(b) Fit a tree to the training data, with Purchase as the response and the other variables except for Buy 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?

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

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ.train)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  7
## Residual mean deviance:  0.7517 = 596.1 / 793
## Misclassification error rate: 0.155 = 124 / 800
```

The tree only uses two variables: LoyalCH and PriceDiff. It has 7 terminal nodes. Training error rate (misclassification error) for the tree is 0.155

(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.

```r
oj.tree
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1075.00 CH ( 0.60250 0.39750 )
##    2) LoyalCH < 0.5036 359  422.80 MM ( 0.27577 0.72423 )
##      4) LoyalCH < 0.276142 170  119.10 MM ( 0.11176 0.88824 ) *
##      5) LoyalCH > 0.276142 189  257.50 MM ( 0.42328 0.57672 )
##       10) PriceDiff < 0.05 79   76.79 MM ( 0.18987 0.81013 ) *
##       11) PriceDiff > 0.05 110  148.80 CH ( 0.59091 0.40909 ) *
##    3) LoyalCH > 0.5036 441  343.30 CH ( 0.86848 0.13152 )
##      6) LoyalCH < 0.764572 186  210.30 CH ( 0.74731 0.25269 )
##       12) PriceDiff < -0.165 29   34.16 MM ( 0.27586 0.72414 ) *
##       13) PriceDiff > -0.165 157  140.90 CH ( 0.83439 0.16561 )
##         26) PriceDiff < 0.265 82   95.37 CH ( 0.73171 0.26829 ) *
##         27) PriceDiff > 0.265 75   31.23 CH ( 0.94667 0.05333 ) *
##      7) LoyalCH > 0.764572 255   90.67 CH ( 0.95686 0.04314 ) *
```

Let's pick terminal node labeled "10)". The splitting variable at this node is PriceDiffPriceDiff. The splitting value of this node is 0.05. There are 79 points in the subtree below this node. The deviance for all points contained in region below this node is 80. A * in the line denotes that this is in fact a terminal node. The prediction at this node is Sales = MM. About 19% points in this node have CH as value of Sales. Remaining 81% points have MM as value of Sales.

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

LoyalCH is the most important variable of the tree, in fact top 3 nodes contain LoyalCH. If LoyalCH<0.27, the tree predicts MM. If LoyalCH>0.76, the tree predicts CH. For intermediate values of LoyalCH, the decision also depends on the value of 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?

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

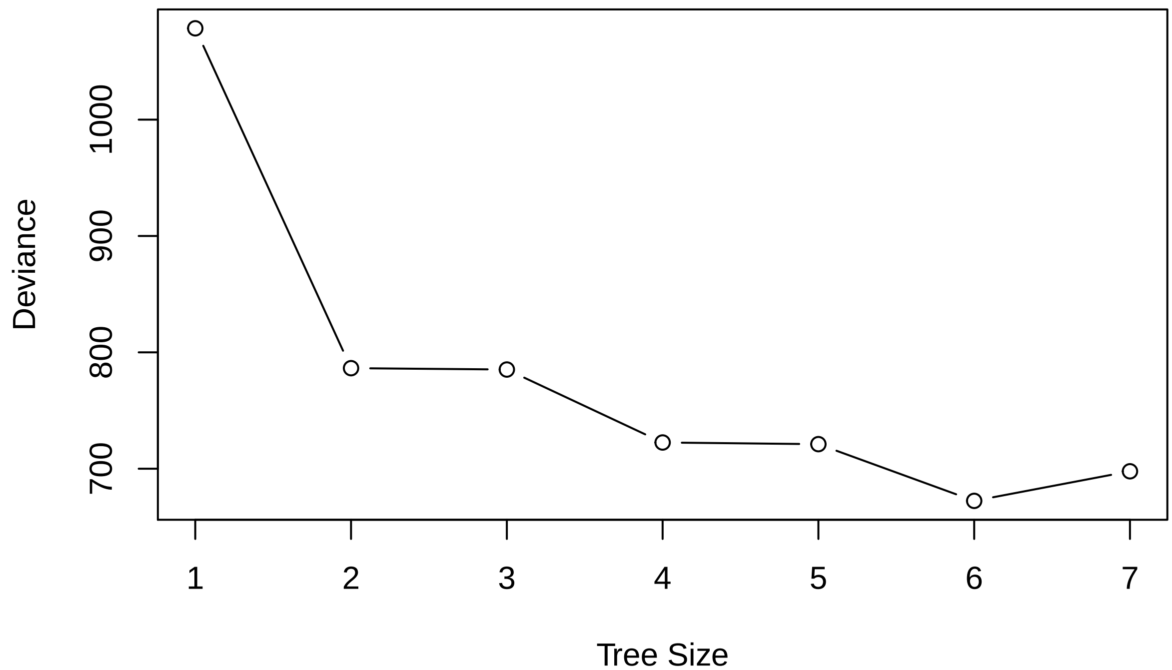
```
##     oj.pred
##       CH  MM
##   CH 152  19
##   MM  32  67
```

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

```
cv.oj = cv.tree(oj.tree, FUN = prune.tree)
```

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

```
plot(cv.oj$size, cv.oj$dev, type = "b", xlab = "Tree Size", ylab = "Deviance")
```



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

Size of 6 gives lowest cross-validation error.

(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.

```
oj.pruned = prune.tree(oj.tree, best = 6)
```

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

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

```
##
## Classification tree:
## snip.tree(tree = oj.tree, nodes = 13L)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  6
## Residual mean deviance:  0.7689 = 610.5 / 794
## Misclassification error rate: 0.155 = 124 / 800
```

Misclassification error of pruned tree is exactly same as that of original tree - 0.155.

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

```r
pred.unpruned = predict(oj.tree, OJ.test, type = "class")
misclass.unpruned = sum(OJ.test$Purchase != pred.unpruned)
misclass.unpruned/length(pred.unpruned)
```

```
## [1] 0.1888889
```

```r
pred.pruned = predict(oj.pruned, OJ.test, type = "class")
misclass.pruned = sum(OJ.test$Purchase != pred.pruned)
misclass.pruned/length(pred.pruned)
```

```
## [1] 0.1888889
```

Pruned and unpruned trees have same test error rate of 0.189.