Homework 5

Kris Hanus, Laura Glathar, Arkya Rakshit, Jace Crist, Brandon Dlugosz
University of Nebraska at Omaha

BSAD 8700 - Business Analytics
Due: February 16, 2015

ANSWER FOR 8:

(a)
```
library(MASS)
library(dplyr)

##
## Attaching package:  'dplyr'
##
## The following object is masked from 'package:MASS':
##
##     select
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(ISLR)
library(caret)

## Loading required package:  lattice
## Loading required package:  ggplot2

library(tree)
```

For some odd reason echo = FALSE is failing.

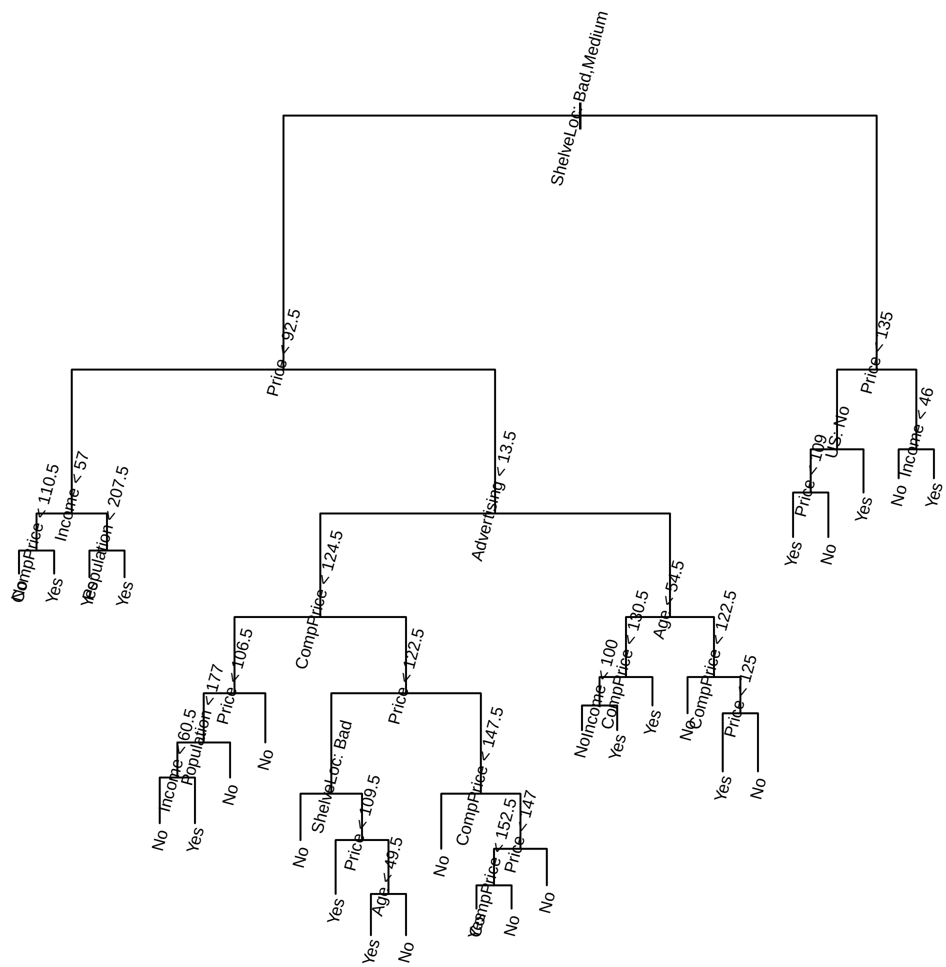
```
attach(Carseats)
High = ifelse (Sales <= 8,"No", "Yes")
Carseats = data.frame(Carseats, High)
head(Carseats)

##   Sales CompPrice Income Advertising Population Price ShelveLoc Age
## 1  9.50       138     73          11        276   120       Bad  42
## 2 11.22       111     48          16        260    83      Good  65
## 3 10.06       113     35          10        269    80    Medium  59
## 4  7.40       117    100           4        466    97    Medium  55
```

```
## 5  4.15       141     64           3        340   128       Bad  38
## 6 10.81       124    113          13        501    72       Bad  78
##   Education Urban  US High
## 1        17   Yes Yes  Yes
## 2        10   Yes Yes  Yes
## 3        12   Yes Yes  Yes
## 4        14   Yes Yes   No
## 5        13   Yes  No   No
## 6        16    No Yes  Yes

set.seed(2345)
DataIndex<-createDataPartition(Carseats$Sales, p = 0.8, list=FALSE, times=1)
train<-Carseats[DataIndex,]
Carseats.test<-Carseats[-DataIndex,]
High.test=High[-DataIndex]
dim(Carseats)

## [1] 400  12

dim(train)

## [1] 321  12

dim(Carseats.test)

## [1] 79 12
```

Carseatstraining is a randomly chosen training data set, and Carseatstest is a randomly chosen test data set. Both were selected by non-replacement methods. This is a better method of selection (then what the book shows) creating no bias and no sequencing issues in the data.

(b)
```
str(Carseats)

## 'data.frame': 400 obs. of  12 variables:
##  $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
##  $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
##  $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
##  $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
##  $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
##  $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
##  $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
##  $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
##  $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
##  $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
##  $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
##  $ High       : Factor w/ 2 levels "No","Yes": 2 2 2 1 1 2 1 2 1 1 ...

tree.carseats=tree(High~.-Sales, Carseats)
summary(tree.carseats)

##
## Classification tree:
```

```
## tree(formula = High ~ . - Sales, data = Carseats)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
## [6] "Advertising" "Age"         "US"
## Number of terminal nodes:  27
## Residual mean deviance:  0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400

plot(tree.carseats)
text(tree.carseats, pretty = 0,cex=0.6,srt=75)
```



$MSE = 9\%$. By forcing the model to create an optimal dataset and model selection this should eliminating error.

(c)
```
tree.carseats
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##   1) root 400 541.500 No ( 0.59000 0.41000 )
##     2) ShelveLoc: Bad,Medium 315 390.600 No ( 0.68889 0.31111 )
##       4) Price < 92.5 46  56.530 Yes ( 0.30435 0.69565 )
##         8) Income < 57 10  12.220 No ( 0.70000 0.30000 )
##          16) CompPrice < 110.5 5   0.000 No ( 1.00000 0.00000 ) *
##          17) CompPrice > 110.5 5   6.730 Yes ( 0.40000 0.60000 ) *
##         9) Income > 57 36  35.470 Yes ( 0.19444 0.80556 )
##          18) Population < 207.5 16  21.170 Yes ( 0.37500 0.62500 ) *
##          19) Population > 207.5 20   7.941 Yes ( 0.05000 0.95000 ) *
##       5) Price > 92.5 269 299.800 No ( 0.75465 0.24535 )
##        10) Advertising < 13.5 224 213.200 No ( 0.81696 0.18304 )
##          20) CompPrice < 124.5 96  44.890 No ( 0.93750 0.06250 )
##            40) Price < 106.5 38  33.150 No ( 0.84211 0.15789 )
##              80) Population < 177 12  16.300 No ( 0.58333 0.41667 )
##               160) Income < 60.5 6   0.000 No ( 1.00000 0.00000 ) *
##               161) Income > 60.5 6   5.407 Yes ( 0.16667 0.83333 ) *
##              81) Population > 177 26   8.477 No ( 0.96154 0.03846 ) *
##            41) Price > 106.5 58   0.000 No ( 1.00000 0.00000 ) *
##          21) CompPrice > 124.5 128 150.200 No ( 0.72656 0.27344 )
##            42) Price < 122.5 51  70.680 Yes ( 0.49020 0.50980 )
##              84) ShelveLoc: Bad 11   6.702 No ( 0.90909 0.09091 ) *
##              85) ShelveLoc: Medium 40  52.930 Yes ( 0.37500 0.62500 )
##               170) Price < 109.5 16   7.481 Yes ( 0.06250 0.93750 ) *
##               171) Price > 109.5 24  32.600 No ( 0.58333 0.41667 )
##                 342) Age < 49.5 13  16.050 Yes ( 0.30769 0.69231 ) *
##                 343) Age > 49.5 11   6.702 No ( 0.90909 0.09091 ) *
##            43) Price > 122.5 77  55.540 No ( 0.88312 0.11688 )
##              86) CompPrice < 147.5 58  17.400 No ( 0.96552 0.03448 ) *
##              87) CompPrice > 147.5 19  25.010 No ( 0.63158 0.36842 )
##               174) Price < 147 12  16.300 Yes ( 0.41667 0.58333 )
##                 348) CompPrice < 152.5 7   5.742 Yes ( 0.14286 0.85714 ) *
##                 349) CompPrice > 152.5 5   5.004 No ( 0.80000 0.20000 ) *
##               175) Price > 147 7   0.000 No ( 1.00000 0.00000 ) *
##        11) Advertising > 13.5 45  61.830 Yes ( 0.44444 0.55556 )
##          22) Age < 54.5 25  25.020 Yes ( 0.20000 0.80000 )
##            44) CompPrice < 130.5 14  18.250 Yes ( 0.35714 0.64286 )
##              88) Income < 100 9  12.370 No ( 0.55556 0.44444 ) *
##              89) Income > 100 5   0.000 Yes ( 0.00000 1.00000 ) *
##            45) CompPrice > 130.5 11   0.000 Yes ( 0.00000 1.00000 ) *
##          23) Age > 54.5 20  22.490 No ( 0.75000 0.25000 )
##            46) CompPrice < 122.5 10   0.000 No ( 1.00000 0.00000 ) *
##            47) CompPrice > 122.5 10  13.860 No ( 0.50000 0.50000 )
##              94) Price < 125 5   0.000 Yes ( 0.00000 1.00000 ) *
##              95) Price > 125 5   0.000 No ( 1.00000 0.00000 ) *
##     3) ShelveLoc: Good 85  90.330 Yes ( 0.22353 0.77647 )
##       6) Price < 135 68  49.260 Yes ( 0.11765 0.88235 )
##        12) US: No 17  22.070 Yes ( 0.35294 0.64706 )
##          24) Price < 109 8   0.000 Yes ( 0.00000 1.00000 ) *
##          25) Price > 109 9  11.460 No ( 0.66667 0.33333 ) *
```

```
##         13) US: Yes 51  16.880 Yes ( 0.03922 0.96078 ) *
##        7) Price > 135 17  22.070 No ( 0.64706 0.35294 )
##         14) Income < 46 6   0.000 No ( 1.00000 0.00000 ) *
##         15) Income > 46 11  15.160 Yes ( 0.45455 0.54545 ) *
```

```
set.seed (2)
tree.carseats =tree(High~.-Sales, train)
tree.pred=predict(tree.carseats, Carseats.test, type = 'class' )
table(tree.pred, High.test)
```

```
##          High.test
## tree.pred No Yes
##       No  37   8
##       Yes 12  22
```

```
(59)/79
```

```
## [1] 0.7468354
```

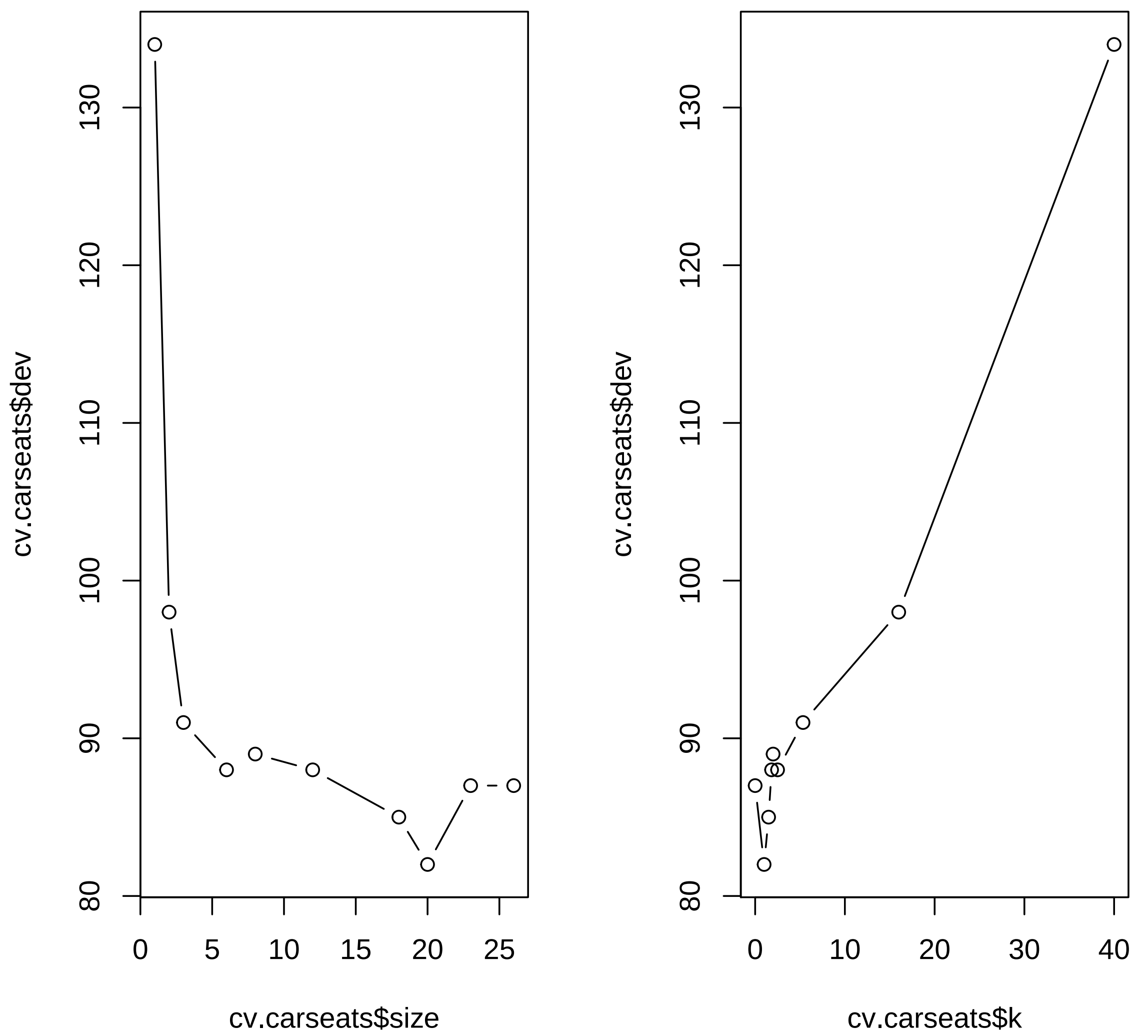
```
set.seed (3)
cv.carseats =cv.tree(tree.carseats ,FUN=prune.misclass )
names(cv.carseats )
```

```
## [1] "size"   "dev"    "k"      "method"
```

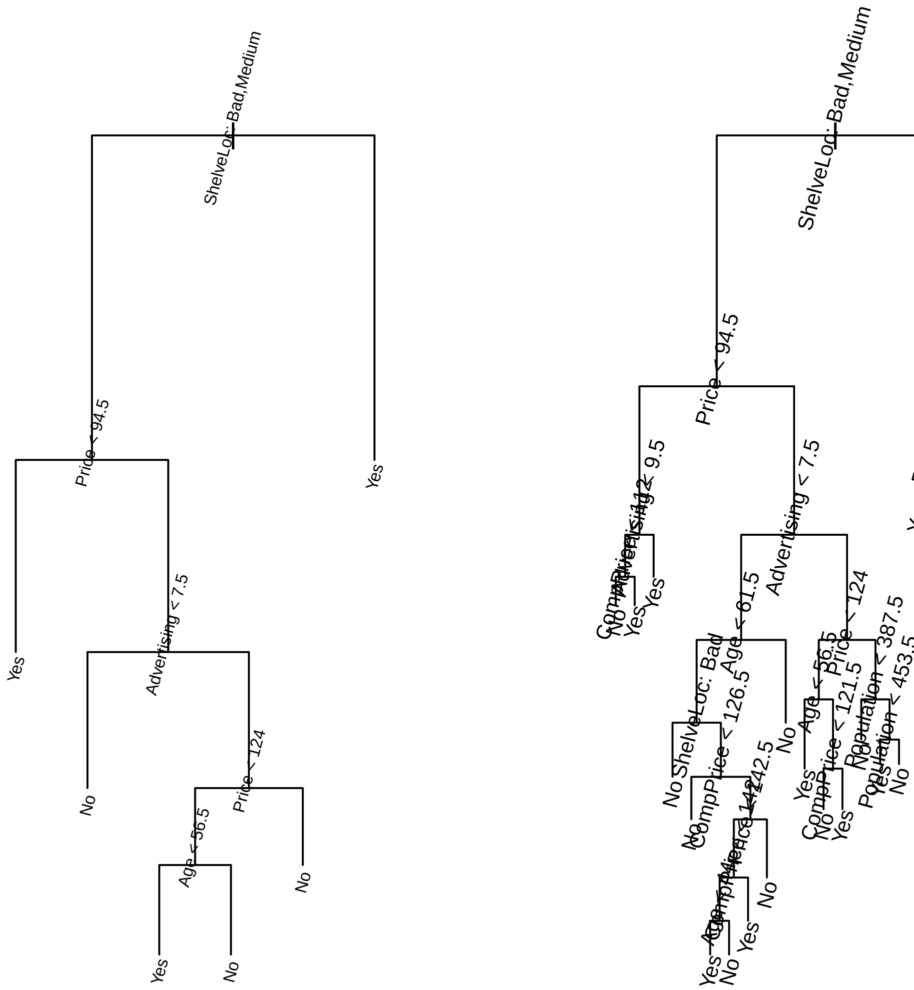
```
cv.carseats
```

```
## $size
##  [1] 26 23 20 18 12  8  6  3  2  1
##
## $dev
##  [1]  87  87  82  85  88  89  88  91  98 134
##
## $k
##  [1]      -Inf  0.000000  1.000000  1.500000  1.833333  2.000000  2.500000
##  [8]  5.333333 16.000000 40.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

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




```
prune.carseats =prune.misclass (tree.carseats ,best =6)
plot(prune.carseats )
text(prune.carseats ,pretty =0,cex=0.6,srt=75)

tree.pred=predict (prune.carseats , Carseats.test ,type="class")
table(tree.pred ,High.test)

##           High.test
## tree.pred No Yes
##       No  36  10
##       Yes 13  20

prune.carseats =prune.misclass (tree.carseats ,best =20)
plot(prune.carseats )
text(prune.carseats ,pretty =0,cex=0.75,srt=75)
```



```r
tree.pred=predict (prune.carseats , Carseats.test ,type="class")
table(tree.pred ,High.test)

##          High.test
## tree.pred No Yes
##       No  37  10
##       Yes 12  20
```

With pruning, my results from part b it seems that the fitted model to the training data is predicting with a 75% accuracy. This is 3.5% better then the books method. Pruning my model is only pushing around the error. It is not improving the model. However, using bagging methods instead might be the better method.

ANSWER FOR 9:

(a)
```
tail(OJ, 3)

##      Purchase WeekofPurchase StoreID PriceCH PriceMM DiscCH DiscMM
## 1068       MM            257       7    1.86    2.18      0   0.00
## 1069       CH            261       7    1.86    2.13      0   0.24
## 1070       CH            270       1    1.86    2.18      0   0.00
##      SpecialCH SpecialMM  LoyalCH SalePriceMM SalePriceCH PriceDiff Store7
## 1068         0         0 0.736206        2.18        1.86      0.32    Yes
## 1069         0         0 0.588965        1.89        1.86      0.03    Yes
## 1070         0         0 0.671172        2.18        1.86      0.32     No
##      PctDiscMM PctDiscCH ListPriceDiff STORE
## 1068  0.000000         0          0.32     0
## 1069  0.112676         0          0.27     0
## 1070  0.000000         0          0.32     1

800/1070

## [1] 0.7476636

attach(OJ)
set.seed(5432)
DataIndex2<-createDataPartition(OJ$Purchase, p = 0.747, list=FALSE, times=1)
train2<-OJ[DataIndex2,]
OJ.test<-OJ[-DataIndex2,]
dim(train2)

## [1] 800  18

dim(OJ.test)

## [1] 270  18
```

We decided to use a random sample because it will get rid of any bias or sequencing unknown to us in our sampling.

(b)
```
tree.OJ=tree(Purchase~.,train2)
summary(tree.OJ)

##
## Classification tree:
## tree(formula = Purchase ~ ., data = train2)
## Variables actually used in tree construction:
## [1] "LoyalCH"     "SalePriceMM" "SpecialCH"   "PriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7258 = 574.8 / 792
## Misclassification error rate: 0.165 = 132 / 800
```

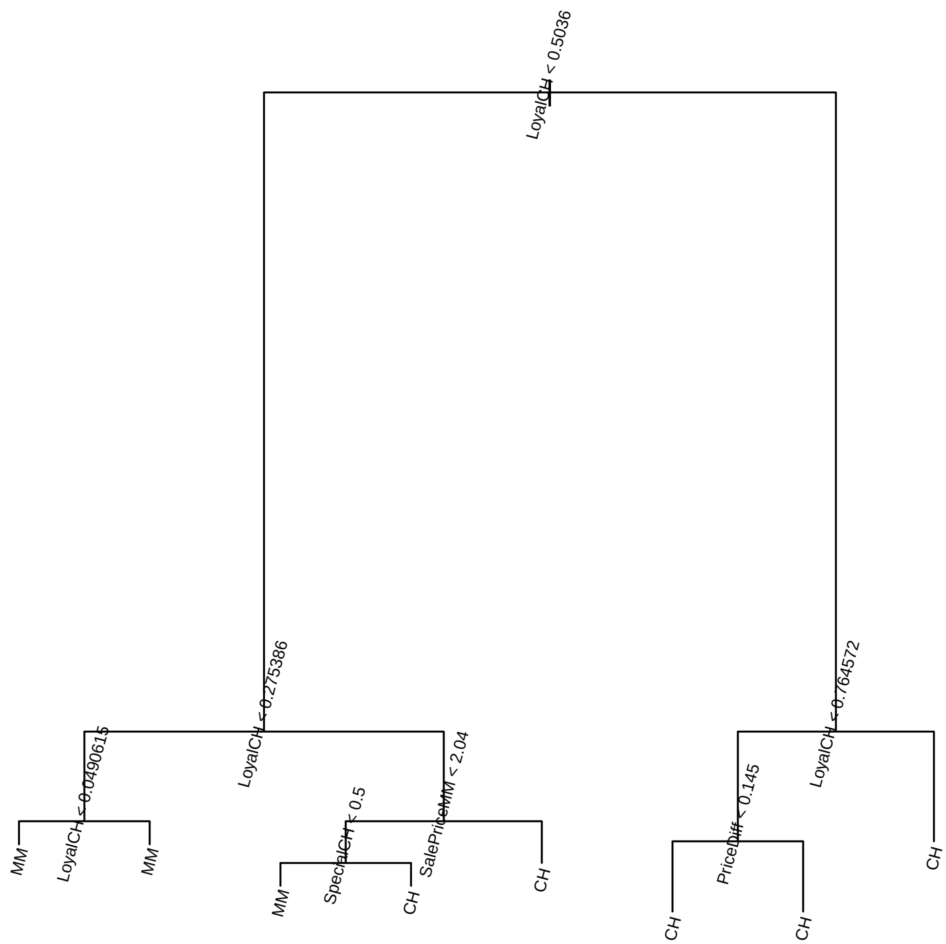
There were 8 terminal nodes with a $MSE = 16.25\%$.

(c)
```r
tree.OJ
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1070.00 CH ( 0.61000 0.39000 )
##    2) LoyalCH < 0.5036 353  415.10 MM ( 0.27479 0.72521 )
##      4) LoyalCH < 0.275386 172  123.60 MM ( 0.11628 0.88372 )
##        8) LoyalCH < 0.0490615 60   10.17 MM ( 0.01667 0.98333 ) *
##        9) LoyalCH > 0.0490615 112  102.00 MM ( 0.16964 0.83036 ) *
##      5) LoyalCH > 0.275386 181  246.90 MM ( 0.42541 0.57459 )
##       10) SalePriceMM < 2.04 94  108.90 MM ( 0.26596 0.73404 )
##         20) SpecialCH < 0.5 78   76.37 MM ( 0.19231 0.80769 ) *
##         21) SpecialCH > 0.5 16   21.17 CH ( 0.62500 0.37500 ) *
##       11) SalePriceMM > 2.04 87  117.30 CH ( 0.59770 0.40230 ) *
##    3) LoyalCH > 0.5036 447  337.30 CH ( 0.87472 0.12528 )
##      6) LoyalCH < 0.764572 193  218.70 CH ( 0.74611 0.25389 )
##       12) PriceDiff < 0.145 77  106.60 CH ( 0.51948 0.48052 ) *
##       13) PriceDiff > 0.145 116   77.16 CH ( 0.89655 0.10345 ) *
##      7) LoyalCH > 0.764572 254   64.09 CH ( 0.97244 0.02756 ) *
```

We picked terminal node begining after leaf 10. It seems that 76.37% of customers will pick Minute Maid OJ unless there is a significant discount offered for Citrus Hill.

(d)
```r
plot(tree.OJ)
text(tree.OJ,pretty =0,cex=0.6,srt=75)
```



There seems to be a focus on the type of customer loyalty for CH based on deals and discounts. Also, in general customers will go will Minute Maid unless there is a savings.

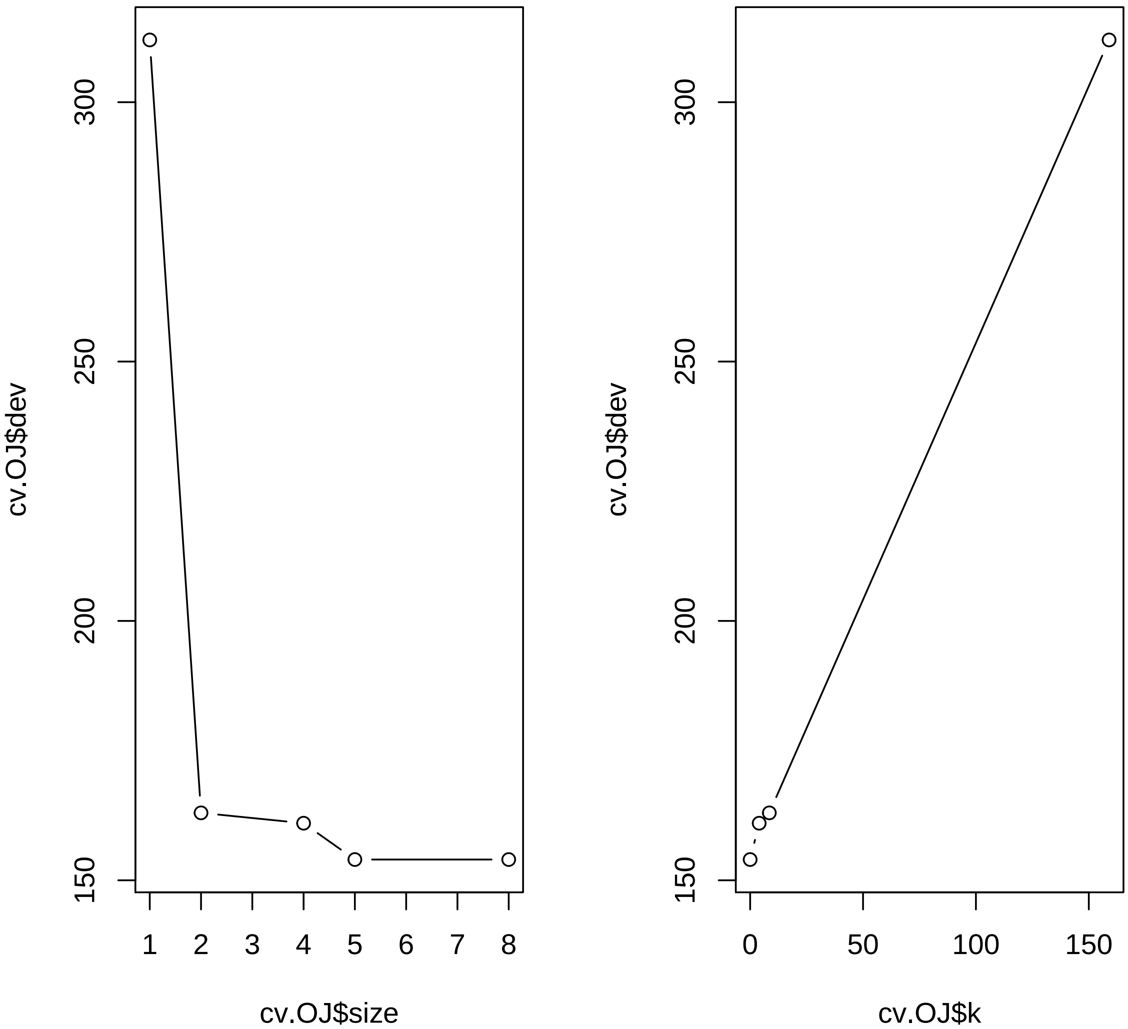
(e)
```
tree.pred2=predict(tree.OJ, OJ.test, type = 'class' )
table(tree.pred2,OJ.test$Purchase)

##
## tree.pred2  CH  MM
##         CH 154  42
##         MM  11  63

(154+63)/270

## [1] 0.8037037
```

We are able to predict with an 80.4% accuracy.

(f)
```
set.seed (3)
cv.OJ =cv.tree(tree.OJ ,FUN=prune.misclass )
names(cv.OJ)

## [1] "size"   "dev"    "k"      "method"

cv.OJ

## $size
## [1] 8 5 4 2 1
##
## $dev
## [1] 154 154 161 163 312
##
## $k
## [1]  -Inf   0.0   4.0   8.5 159.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

The optimal tree size would have 4 terminal nodes. Dev is the lowest when size is 4.

(g)
```
par(mfrow =c(1,2))
plot(cv.OJ$size ,cv.OJ$dev ,type="b")
plot(cv.OJ$k ,cv.OJ$dev ,type="b")
```



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

(h) As stated in part f, the optimal tree size would have 4 terminal nodes.