Predictive Modeling

Series 8

Exercise 8.1

Draw an example (of your own invention) of a partition of a two-dimensional feature space that could result from recursive binary splitting. Your example should contain at least five regions. Draw a decision tree corresponding to this partition. Be sure to label all aspects of your figures, including the regions $R_1, R_2, \ldots$, the cutpoints $t_1, t_2, \ldots$, and so forth.

Hint: Your result should look something like Figures 6.2 and 6.3 in the lecture notes. You may consider the built-in **R**-dataset **`mtcars`**.

Exercise 8.2

Consider the Gini index, classification error rate, and cross-entropy in a simple classification setting with two classes. Create a single plot that displays each of these quantities as a function of $\hat{p}_{m1}$. The x-axis should display $\hat{p}_{m1}$, ranging from 0 to 1, and the y-axis should display the value of the Gini index, classification error, and entropy.

Hint: In a setting with two classes, $\hat{p}_{m2} = 1 - \hat{p}_{m1}$. You could make this plot by hand, but it will be much easier to generate it in **R**.

Exercise 8.3

The abbreviation CART stands for **C**lassification **A**nd **R**egression **T**rees. So far we have only considered classification trees. In this exercise we will now focus on the regression setting.

We assume that observations $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$ are given, where $\hat{x}_i$ are the predictor values and y_i is a *numerical* response. When we partition the predictor space into M regions $R_1, \ldots, R_M$ by means of binary splitting, we predict the value of the response in the j-th region by

$$\hat{y}_{R_j} = \frac{1}{\#R_j} \sum_{i \in R_j} y_i$$

where $\#R_j$ denotes the number of data points in R_j. In order to assess the quality of the regression tree, we compute the *residual sum of squeares (RSS)*

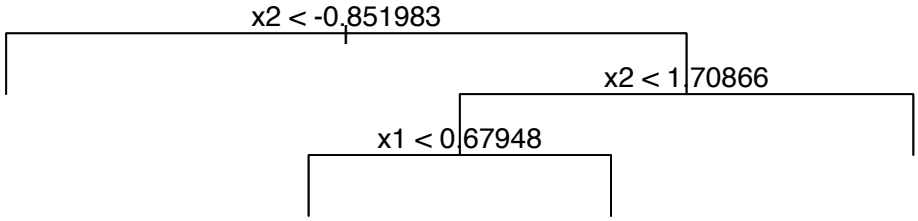
$$\sum_{j=1}^{M} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Regression trees are grown in such a way that each split leads to the greatest possible reduction in RSS.

Assume we are given the following data:

	x1	x2	y
1	-0.85	1.83	3.45
2	1.15	-0.19	4.80
3	-0.36	0.71	3.52
4	1.53	0.29	3.21
5	1.76	-1.59	13.46
6	-1.82	1.60	8.10
7	0.11	-1.02	9.88
8	1.57	-1.83	15.01
9	0.21	-0.69	7.86
10	-0.17	1.82	1.41

Here y is the numeric response and x_1 and x_2 the predictor values. Further we have a decision tree of the form



a) Sketch the predictor space, including the data points and the partition according to the tree.

b) Compute the predicted values $\hat{y}$ for each terminal node of the tree.

c) Compute the value of RSS for this tree.

Exercise 8.4

This problem involves the `OJ` data set, which is part of the `ISLR`package.It contains purchase records of 1070 of customers either buying Citrus Hill (CH) or Minute Maid (MM) Orange Juice. 17 characteristics of the customer and the product are recorded.

a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining 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?

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.

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

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?

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

h) Which tree size 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.

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

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

Exercise 8.5

We will now compare regression and classification trees by means of the `Auto` data set as already studied in Exercise 6.4. Make sure, that the auxiliary variable `mpg01` is available, i.e. a binary variable that is 1 if `mpg` is larger than its median an 0 else.

a) Compute a classification tree that predicts the variable `mpg01`. Compute the optimal tree size by invoking the `cv.tree()`-function and provide the cross-validated classification error.

b) Compute a regression tree that predicts the variable `mpg`. Again, use the `cv.tree()` function to obtain the optimal tree. How large is the RSS?

c) Construct a classifier for predicting `mpg01` based on the regression model in b) by thresholding the *predicted* value of `mpg` at the median of the training set. Compare the training error of this classifier and the one in a).

Result Checker

E 8.2: Hint:

$$
\begin{aligned}
E &= 1 - \max(\hat{p}_{m1}, 1 - \hat{p}_{m1}) \\
G &= 2\hat{p}_{m1}(1 - \hat{p}_{m1}) \\
D &= -\hat{p}_{m1}\log(\hat{p}_{m1}) - (1 - \hat{p}_{m1})\log(1 - \hat{p}_{m1})
\end{aligned}
$$

E 8.3: b) c)

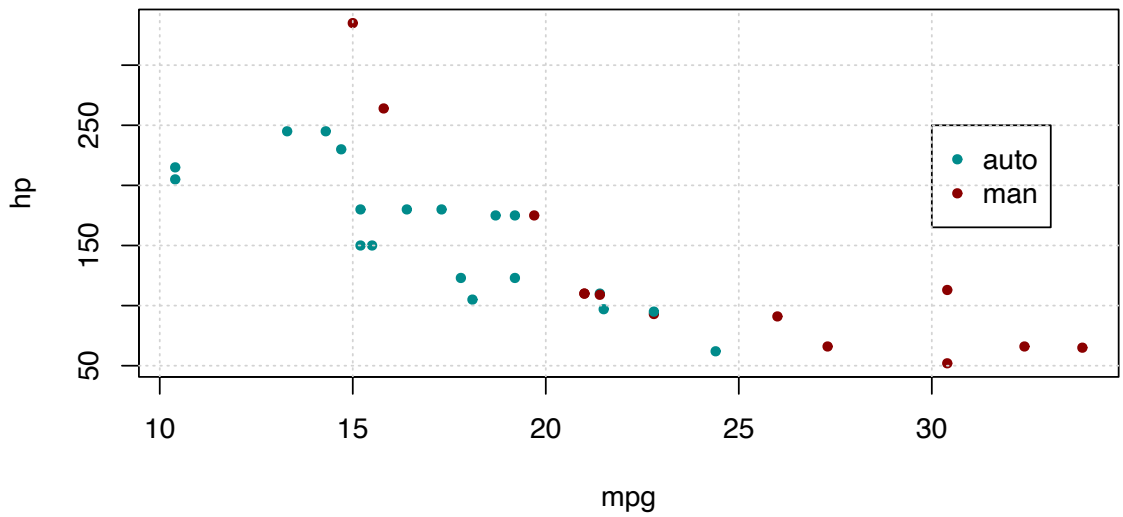
$$
\begin{aligned}
\hat{y}_{R_1} &= 12.78 \\
\hat{y}_{R_2} &= 6.49 \\
\hat{y}_{R_3} &= 4.00 \\
\hat{y}_{R_4} &= 2.43
\end{aligned}
\qquad \text{RSS} = 30.438
$$

Predictive Modeling

Solutions to Series 8

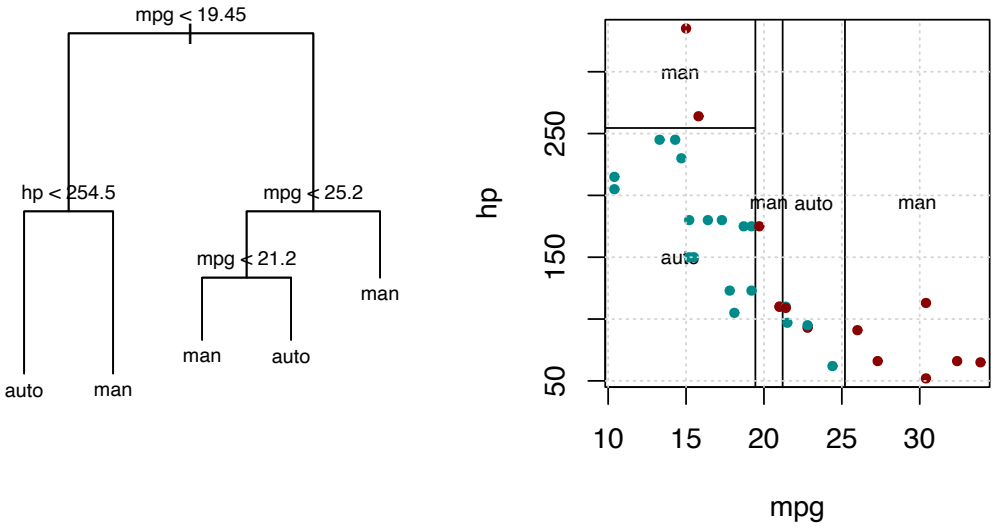
Solution 8.1 As an example we use **am** (automatic or manual transmission) from the data set **mtcars** with the two predictors **hp** and **mpg**.

```
library(tree)
label = as.integer(mtcars$am) + 1
cols = c("darkcyan", "darkred")
data <- mtcars
data["am"] <- factor(data$am, levels = c(0, 1), labels = c("auto",
    "man"))
plot(hp ~ mpg, data = data, col = cols[label], pch = 20)
legend(30, 250, legend = c("auto", "man"), col = cols, pch = 20)
grid()
```




```
par(mfrow = c(1, 2))

tc = tree.control(32, minsize = 4)
tm <- tree(am ~ mpg + hp, data = data, control = tc)
```

```r
plot(tm)
text(tm, cex = 0.7)

partition.tree(tm, cex = 0.7)
points(hp ~ mpg, data = data, col = cols[label], pch = 20)
grid()
```



Solution 8.2

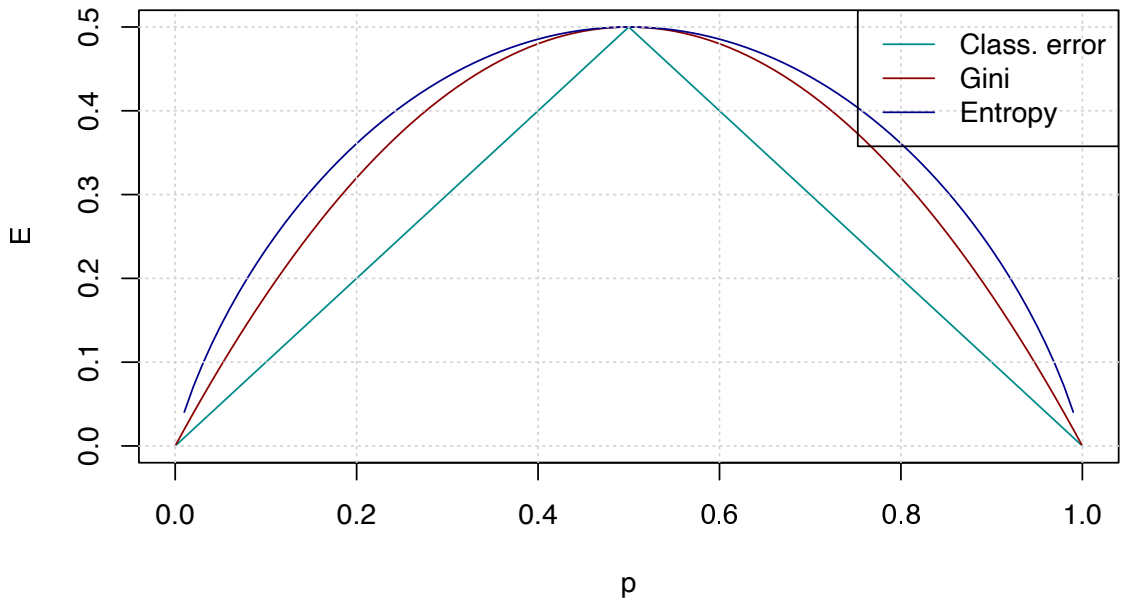
From the formulas in the script and from the relation $\hat{p}_{m2} = 1 - \hat{p}_{m1}$ we infer that

$$
\begin{aligned}
E &= 1 - \max(\hat{p}_{m1}, 1 - \hat{p}_{m1}) \\
G &= 2\hat{p}_{m1}(1 - \hat{p}_{m1}) \\
D &= -\hat{p}_{m1}\log(\hat{p}_{m1}) - (1 - \hat{p}_{m1})\log(1 - \hat{p}_{m1})
\end{aligned}
$$

With this we can use e.g. **R** to perform the plotting:

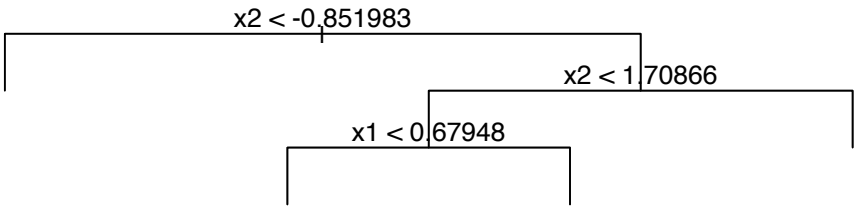
```r
p = seq(0, 1, by = 0.01)
cols = c("darkcyan", "darkred", "darkblue")
# classification error rate
E = 1 - pmax(p, 1 - p)

# Gini index
G = 2 * p * (1 - p)
```

```
# cross entropy (scaled)
D = -p * log(p) - (1 - p) * log(1 - p)
D = D * 0.5/max(D, na.rm = T)


plot(p, E, type = "l", col = cols[1])
lines(p, G, col = cols[2])
lines(p, D, col = cols[3])
grid()
legend("topright", legend = c("Class. error", "Gini", "Entropy"),
    lty = 1, col = cols)
```

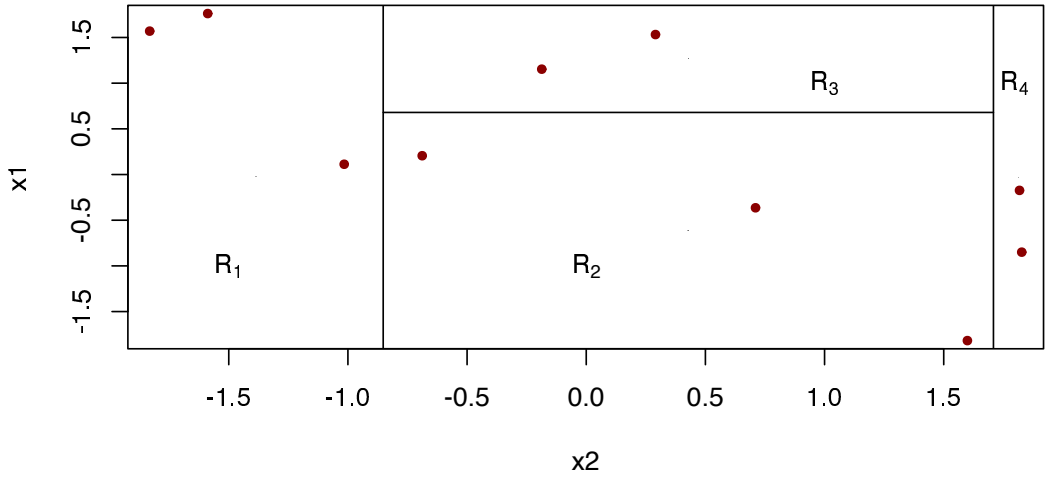


Note that we have scaled the cross-entropy to pass through the point $(0.5; 0.5)$ such that it becomes comparable to the other two error measures.

Solution 8.3



a)



b) We find that observations number 5, 7 and 8 satisfy the condition for the first terminal node R_1 (from the left). By averaging the corresponding y variables we get

$$\hat{y}_{R_1} = \frac{1}{3}(13.46 + 15.01 + 9.88) = 12.78$$

Likewise we find

$$\hat{y}_{R_2} = \frac{1}{3}(7.86 + 3.52 + 8.10) = 6.49$$
$$\hat{y}_{R_3} = \frac{1}{2}(4.8 + 3.21) = 4.00$$
$$\hat{y}_{R_4} = \frac{1}{2}(3.45 + 1.41) = 2.43$$

c) The RSS as defined above reads

$$\text{RSS} = (13.46 - 12.78)^2 + (15.01 - 12.78)^2 + \cdots + (1.41 - 2.43)^2 \approx 30.438$$

Solution 8.4

a)
```
require(ISLR)
set.seed(123)
idx.train = sample.int(nrow(OJ), size = 800)
train.set = OJ[idx.train, ]
test.set = OJ[-idx.train, ]
```

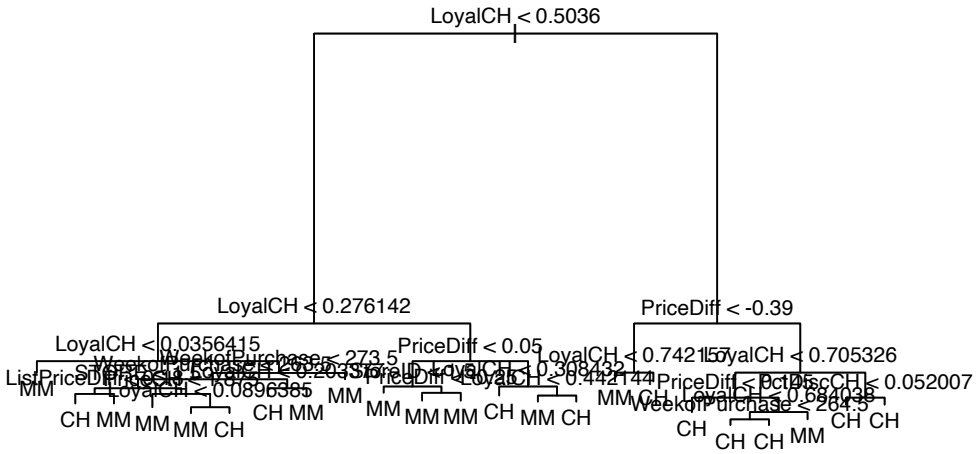
b)
```
require(tree)
tc = tree.control(nobs = 800, mincut = 5,
                  minsize = 10, mindev = 0.005)
tree.model = tree(Purchase ~ ., data = train.set,
                  control = tc)

##
## Classification tree:
## tree(formula = Purchase ~ ., data = train.set, control = tc)
## Variables actually used in tree construction:
## [1] "LoyalCH"        "WeekofPurchase" "STORE"
## [4] "ListPriceDiff"  "PriceCH"        "PriceDiff"
## [7] "StoreID"        "PctDiscCH"
## Number of terminal nodes:  23
## Residual mean deviance:  0.6237 = 484.6 / 777
## Misclassification error rate: 0.14 = 112 / 800

err=(s$misclass[1])/(s$misclass[2])
```

As we can see, the tree with standard settings has 23 terminal nodes. The training error rate is 0.14.

c)
```
tree.model
```

The output is a bit lengthy and hence omitted. The terminal nodes are indicated by a star. Node 8 is a terminal node with the value **MM**. The probability for **MM** is very high (approx. 0.986).

d)
```
plot(tree.model)
text(tree.model, cex = 0.8)
```



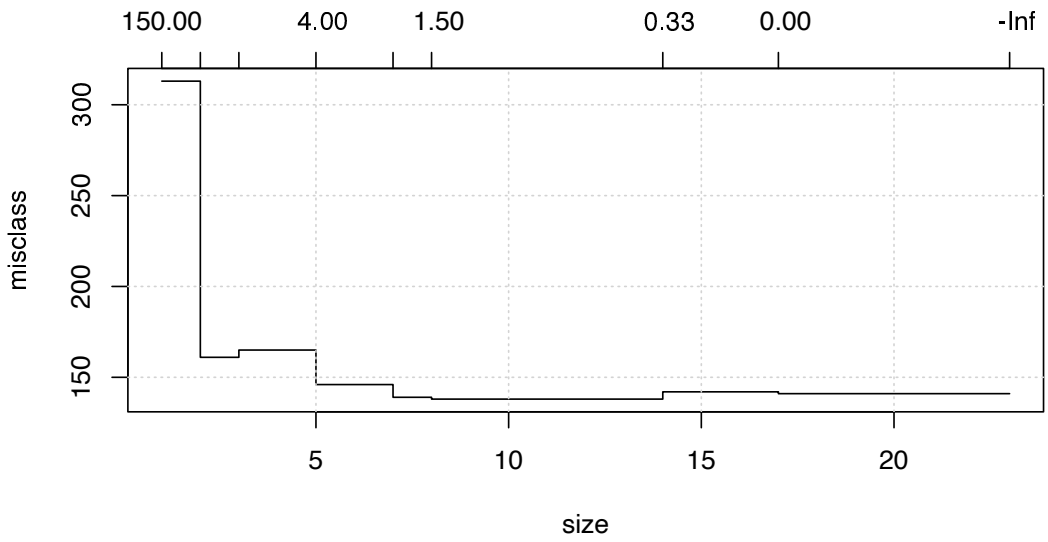
It becomes obvious from the tree, that the customer brand loyality for CH is a variable of great importance when explaining the purchase behaviour.

e)
```
pred.class = predict(tree.model, newdata = test.set,
                     type = "class")
cf = table(test.set$Purchase, pred.class)
addmargins(cf)

##       pred.class
##         CH  MM Sum
##    CH  142  24 166
##    MM   25  79 104
##    Sum 167 103 270

test.error.rate <- 49/270
test.error.rate

## [1] 0.1814815
```

f)
```
tree.cv = cv.tree(tree.model, FUN = prune.misclass, K = 5)
opt.alpha = tree.cv$k[which.min(tree.cv$dev)]
opt.alpha

## [1] 1.5

opt.size = tree.cv$size[which.min(tree.cv$dev)]
```

```
opt.size

## [1] 8
```

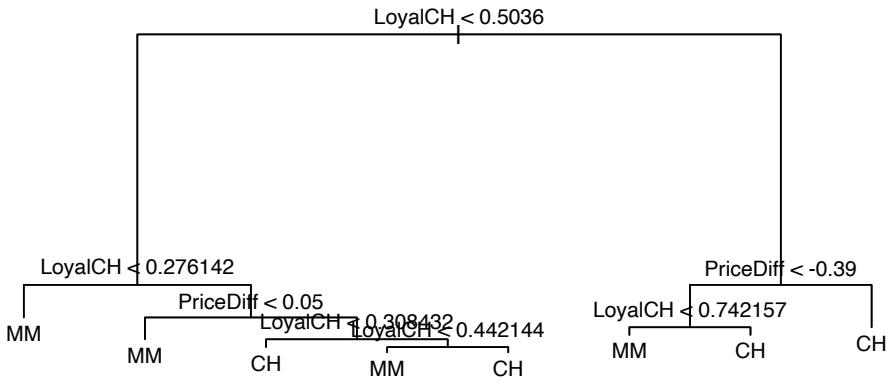
g)
```
plot(tree.cv)
grid()
```



h) From the plot in g) we see that the optimal tree size of is the same as calculated in f).

i) There are two possibilities to do this, either using the optimal size or the optimal alpha.

```
pruned.model <- prune.misclass(tree.model, k = opt.alpha)
# prune.model <- prune.misclass(tree.model,best =
# opt.size)

plot(pruned.model)
text(pruned.model, cex = 0.8)
```



j)
```
summary(tree.model)

##
## Classification tree:
## tree(formula = Purchase ~ ., data = train.set, control = tc)
## Variables actually used in tree construction:
## [1] "LoyalCH"        "WeekofPurchase" "STORE"
## [4] "ListPriceDiff"  "PriceCH"        "PriceDiff"
## [7] "StoreID"        "PctDiscCH"
## Number of terminal nodes:  23
## Residual mean deviance:  0.6237 = 484.6 / 777
## Misclassification error rate: 0.14 = 112 / 800

summary(pruned.model)

##
## Classification tree:
## snip.tree(tree = tree.model, nodes = c(10L, 7L, 4L))
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7978 = 631.9 / 792
## Misclassification error rate: 0.1525 = 122 / 800
```

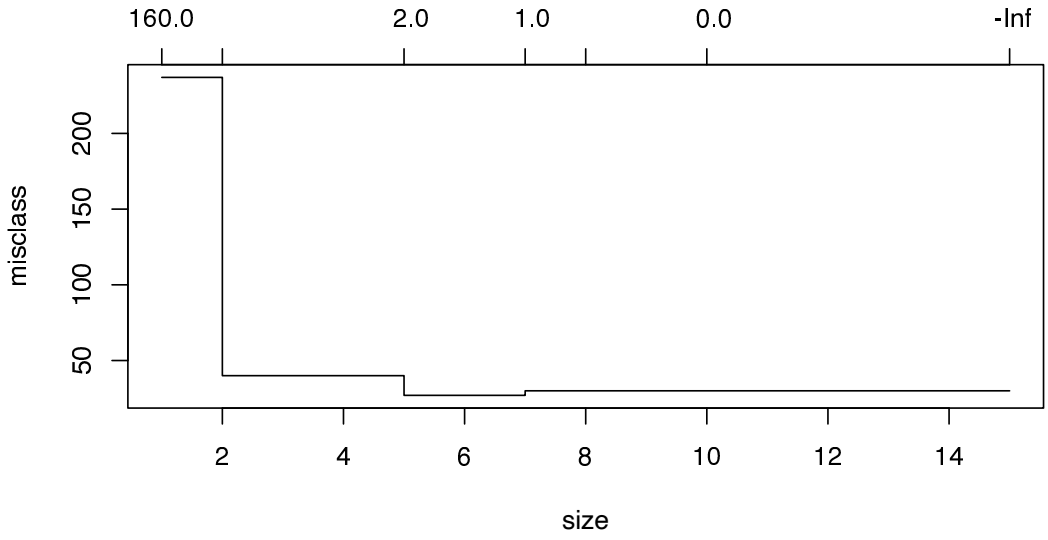
We see that the training error of the original model is slightly smaller than the training error for the pruned model.

k)
```
#from c) we know:
test.error.rate

## [1] 0.1814815

#pruned model
pred.classnewprune <- predict(pruned.model,
                              newdata = test.set,
                              type = "class")
cfnewprune = table(test.set$Purchase,pred.classnewprune)
addmargins(cfnewprune)

##      pred.classnewprune
##        CH  MM Sum
##   CH  142  24 166
##   MM   24  80 104
##   Sum 166 104 270

prune.test.error <- 48/270
prune.test.error

## [1] 0.1777778
```

Solution 8.5

```
require(ISLR)
# additional column mpg01 and remove mpg
AutoC <- Auto[, -which(names(Auto) == "name")]
AutoC$mpg01 <- as.factor(AutoC$mpg > median(AutoC$mpg))
AutoC <- AutoC[, -which(names(AutoC) == "mpg")]
```

a)
```
# tree
require(tree)
tree.model <- tree(mpg01 ~ ., data = AutoC)
tree.cv <- cv.tree(tree.model, method = "misclass", K = 10)
plot(tree.cv)
```

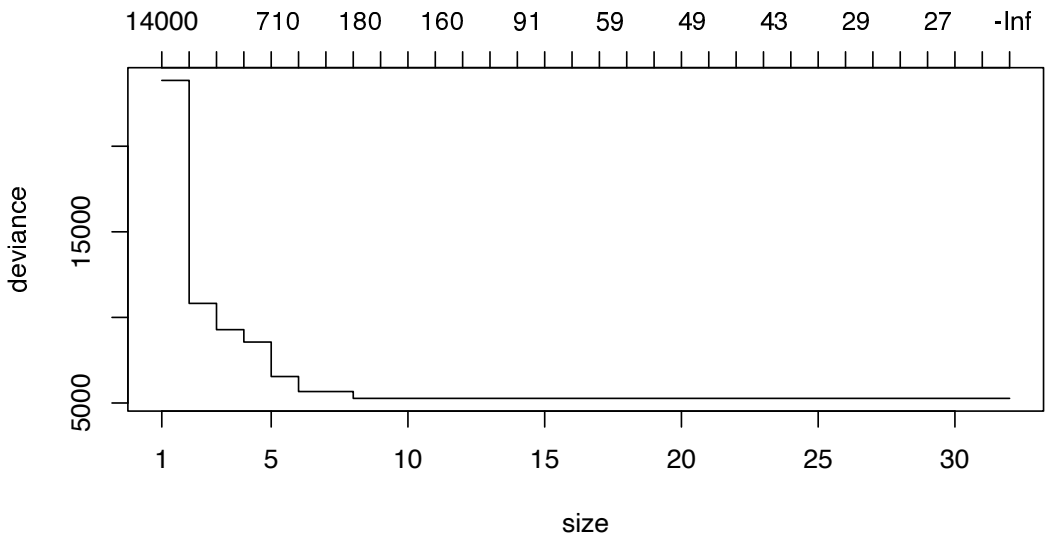



```
min.dev = min(tree.cv$dev)

idx.min = which(min.dev == tree.cv$dev)
opt.size = min(tree.cv$size[idx.min])
opt.size

## [1] 5

opt.alpha = tree.cv$k[which(tree.cv$size == opt.size)]
opt.alpha

## [1] 2

pruned.cla.model <- prune.tree(tree.model, k = opt.alpha)
```

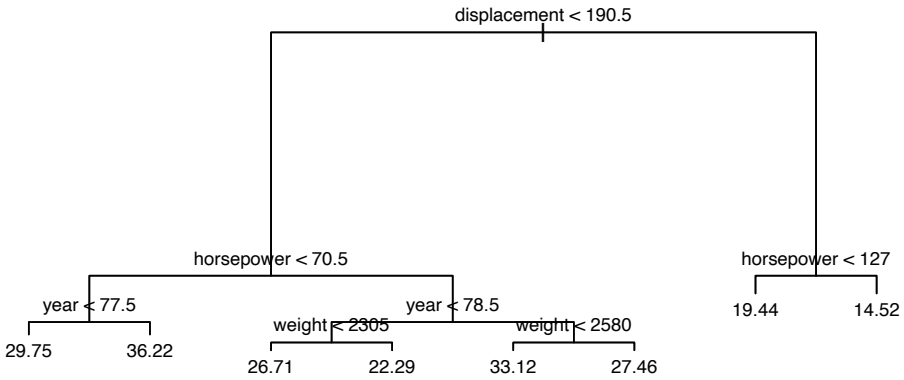
The minimum number of missclassifications in the sequence of pruned trees is 27, i.e. the missclassification rate is 6.89 %.

b)
```
AutoR <- Auto[, -which(names(Auto)=="name")]
tc = tree.control(nobs = nrow(AutoR), mindev = 1e-3)
reg.tree.model <- tree(mpg~.,data = AutoR, control = tc)

reg.tree.cv <- cv.tree(reg.tree.model, K=5)
min.rss = min(reg.tree.cv$dev)
min.rss

## [1] 5269.667
```

```
plot(reg.tree.cv)
```




```
idx.min =  which(min.rss == reg.tree.cv$dev)
opt.size = min(reg.tree.cv$size[idx.min])
opt.size

## [1] 8

opt.alpha = reg.tree.cv$k[which(reg.tree.cv$
                                  size==opt.size)]
opt.alpha

## [1] 176.9088

pruned.reg.model  <- prune.tree(reg.tree.model,
                                k = opt.alpha)
plot(pruned.reg.model)
text(pruned.reg.model, cex = 0.7)
```



The optimal tree has 8 nodes with an RSS of 5269.67.

c) We predict the classes for the regression based classifier by thresholding

```
pred.reg = as.factor(predict(pruned.reg.model)
                     > median(AutoR$mpg))
addmargins(table(pred.reg, AutoC$mpg01))

##
## pred.reg FALSE TRUE Sum
##    FALSE   192   33 225
##    TRUE      4  163 167
##    Sum     196  196 392

pred.cla = predict(pruned.cla.model, type="class")
addmargins(table(pred.cla, AutoC$mpg01))

##
## pred.cla FALSE TRUE Sum
##    FALSE   186    7 193
##    TRUE     10  189 199
##    Sum     196  196 392
```

We see that the classification tree has a better accuracy than the regression tree with subsequent thresholding, although cars with **mpg** < median(**mpg**) are predicited more reliably with the second method. One reason for the smaller overall classification error could be that the classification tree is teached in order to

give the best possible separation if the two classes whereas the regression tree tries to optimize the acutal prediction of the variable `mpg`. The regression tree is hence not so much focused on the critical area of values where thresholding takes place (i.e. near median(`mpg`)) but tries to give a good estimate in the whole range of `mpg`.