Bruce Campbell ST-617 Homework 4

Wed Jul 20 19:52:45 2016

```
rm(list = ls())
set.seed(7)
```

Chapter 8

Problem 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.

```
library(ISLR)
attach(OJ)
train = sample(nrow(OJ), 800)
DF <- OJ
DFTrain <- DF[train, ]
DFTest <- DF[-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?

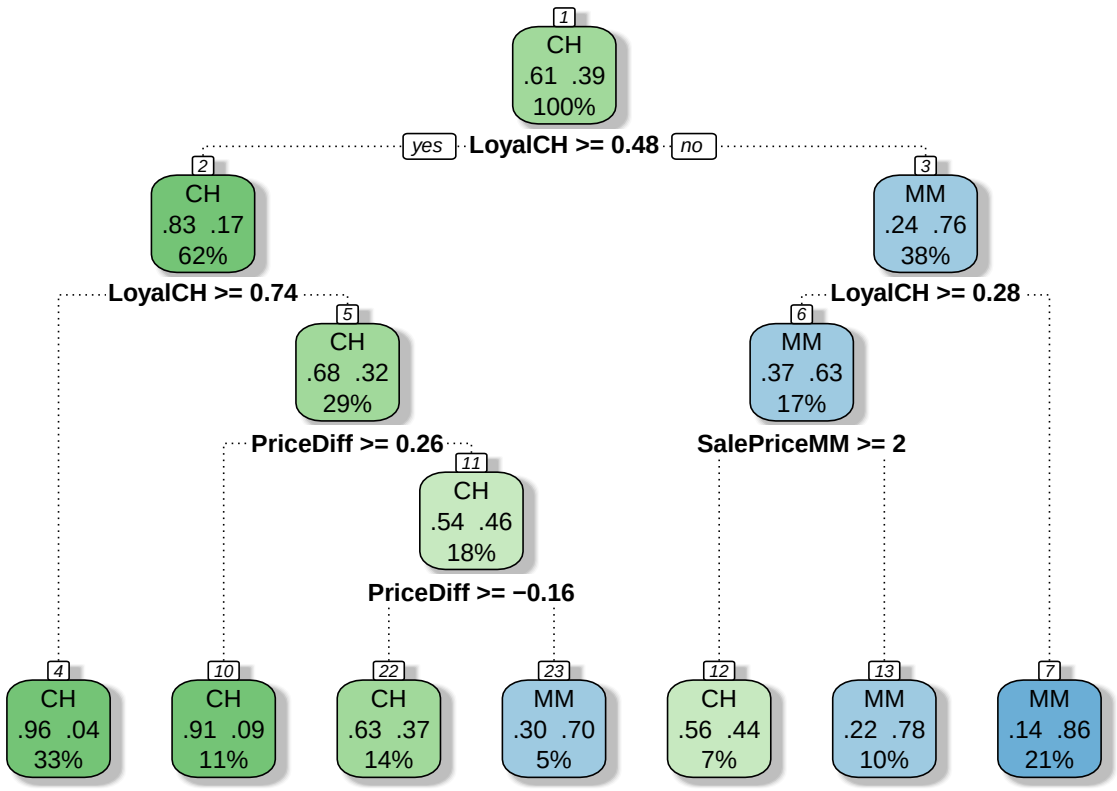
```
names(DFTrain)
```

```
##  [1] "Purchase"       "WeekofPurchase" "StoreID"        "PriceCH"
##  [5] "PriceMM"        "DiscCH"         "DiscMM"         "SpecialCH"
##  [9] "SpecialMM"      "LoyalCH"        "SalePriceMM"    "SalePriceCH"
## [13] "PriceDiff"      "Store7"         "PctDiscMM"      "PctDiscCH"
## [17] "ListPriceDiff"  "STORE"
```

```
library(tree)

control.settings <- tree.control(minsize = 30, nobs = nrow(DFTrain))
tree.oj = tree(Purchase ~ ., DFTrain, control = control.settings, split = "deviance")

summary(tree.oj)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = DFTrain, control = control.settings,
##     split = "deviance")
## Variables actually used in tree construction:
## [1] "LoyalCH"     "SalePriceMM" "PriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7597 = 601.6 / 792
## Misclassification error rate: 0.1788 = 143 / 800
```

```
## We notice that few of the variables are included in the tree despite
## changing the control settings.  We check with rpart to make sure this is
## the case.  rpart does respond to changes in the control setting.
library(rpart)  # Popular decision tree algorithm
library(rattle)  # Fancy tree plot
library(rpart.plot)  # Enhanced tree plots
library(RColorBrewer)  # Color selection for fancy tree plot

control.settings <- rpart.control(minsplit = 100)
tree.rpart <- rpart(Purchase ~ ., data = DFTrain, control = control.settings)

fancyRpartPlot(tree.rpart)
```



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.

```
tree.oj
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1073.00 CH ( 0.60625 0.39375 )
##    2) LoyalCH < 0.482389 303  334.60 MM ( 0.24092 0.75908 )
##      4) LoyalCH < 0.051325 62   10.24 MM ( 0.01613 0.98387 ) *
##      5) LoyalCH > 0.051325 241  293.90 MM ( 0.29876 0.70124 )
##       10) SalePriceMM < 2.04 137  133.10 MM ( 0.18978 0.81022 ) *
##       11) SalePriceMM > 2.04 104  142.80 MM ( 0.44231 0.55769 ) *
##    3) LoyalCH > 0.482389 497  454.80 CH ( 0.82897 0.17103 )
##      6) LoyalCH < 0.764572 250  308.70 CH ( 0.69200 0.30800 )
##       12) PriceDiff < 0.265 157  215.30 CH ( 0.56051 0.43949 ) *
##       13) PriceDiff > 0.265 93   54.54 CH ( 0.91398 0.08602 ) *
##      7) LoyalCH > 0.764572 247   70.62 CH ( 0.96761 0.03239 )
##       14) PriceDiff < -0.39 8   10.59 CH ( 0.62500 0.37500 ) *
##       15) PriceDiff > -0.39 239   48.56 CH ( 0.97908 0.02092 ) *
```

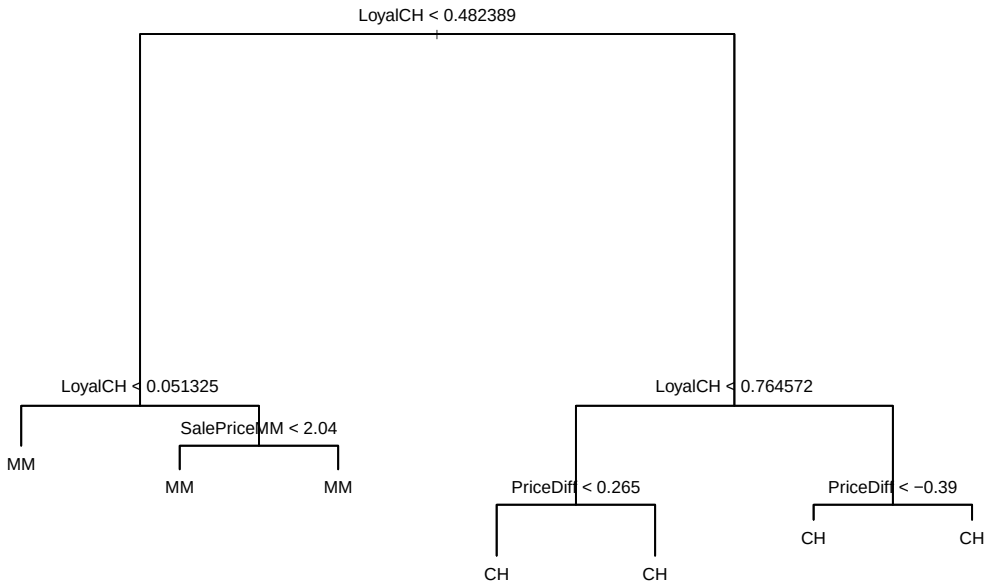
LoyalCH $>$ 0.482389 LoyalCH $<$ 0.764572 PriceDiff $<$ 0.265 PriceDiff $>$ -0.165

is an interesting node. It tells us that there may be a price difference at which a loyal customer my switch brands.

d)

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

```
plot(tree.oj, cex = 0.35)

text(tree.oj, pretty = 0, cex = 0.6)
```



We see quite clearly that brand loyalty is the dominant variable in predicting which brand is purchased.

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.probs = predict(tree.oj, newdata = DFTest)
# oj.probs is a matrix with names columns the first being probability of CH
# the second probability of MM
oj.pred = rep("CH ", nrow(DFTest))
oj.pred[oj.probs[, 1] > 0.5] = " CH"
TB <- table(oj.pred, DFTest$Purchase)
library(pander)
pander(TB)
```

	CH	MM
CH	147	25
CH	21	77

```
ACC_Tree = (TB[1] + TB[4])/length(DFTest$Purchase)
```

The accuracy of the tree classifyer is `0.8296296`

f)

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

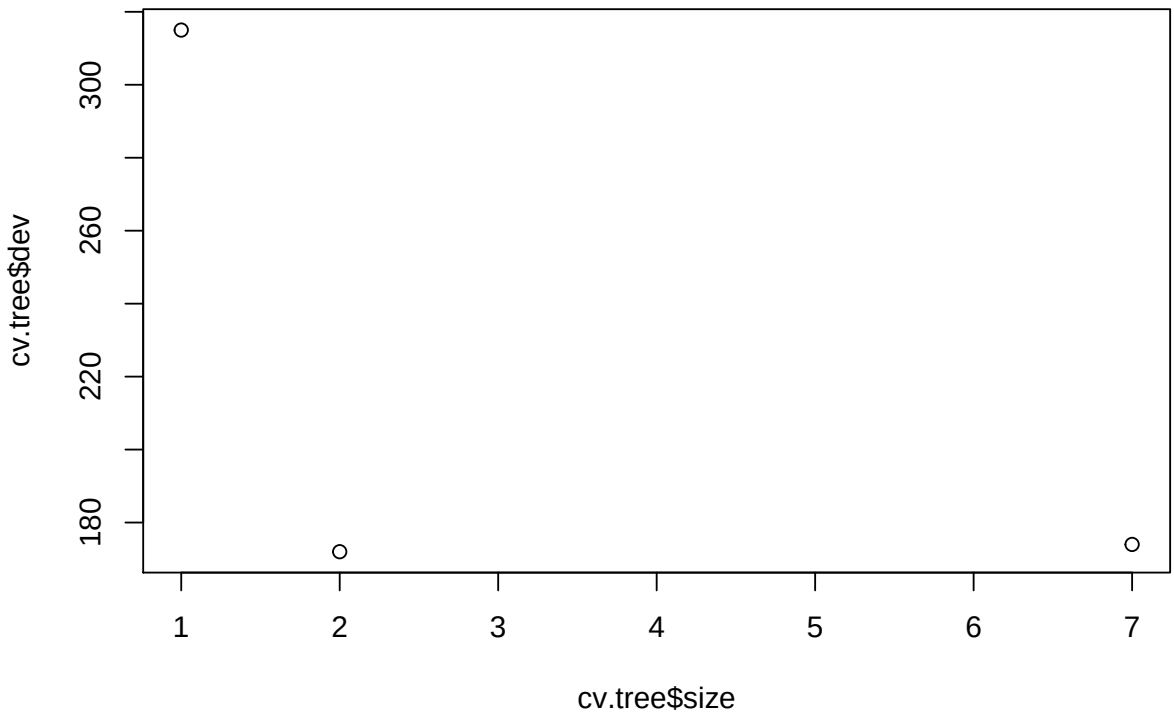
```
cv.tree <- cv.tree(tree.oj, FUN = prune.misclass)
cv.tree
```

```
## $size
## [1] 7 2 1
##
## $dev
## [1] 174 172 315
##
## $k
## [1] -Inf    0  157
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

g)

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

```
plot(cv.tree$size, cv.tree$dev)
```




```
optimal_tree_size <- cv.tree$size[which.min(cv.tree$dev)]
```

h)

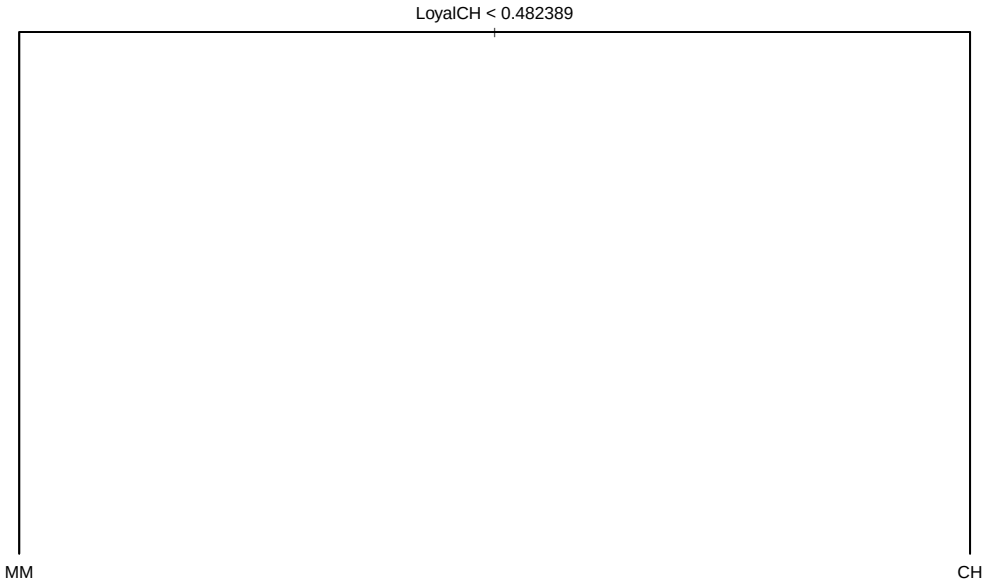
Which tree size corresponds to the lowest cross-validated classification error rate?

The optimal tree size based on the cross validation error rate is 2

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.

```
prune.oj <- prune.misclass(tree.oj, best = optimal_tree_size)

plot(prune.oj, cex = 0.35)
text(prune.oj, pretty = 0, cex = 0.6)
```



j)

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

```
summ_tree.oj <- summary(tree.oj)
train_err_tree.oj <- summ_tree.oj$misclass[1]/summ_tree.oj$misclass[2]

summ_prune.oj <- summary(prune.oj)
train_err_prune.oj <- summ_prune.oj$misclass[1]/summ_prune.oj$misclass[2]

pander(data.frame(tree = c("tree.oj", "prune.tree"), training_error = c(train_err_tree.oj,
    train_err_prune.oj)))
```

tree	training_error
tree.oj	0.1787
prune.tree	0.1975

We have the same error.

k)

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

```
oj.probs = predict(prune.oj, newdata = DFTest)
# oj.probs is a matrix with names columns the first being probability of CH
# the second probability of MM
oj.pred = rep("CH ", nrow(DFTest))
oj.pred[oj.probs[, 1] > 0.5] = " CH"
TB <- table(oj.pred, DFTest$Purchase)
library(pander)

ACC_Prune = (TB[1] + TB[4])/length(DFTest$Purchase)

pander(data.frame(tree = c("tree.oj", "prune.tree"), test_error = c(ACC_Tree,
    ACC_Prune)))
```

tree	test_error
tree.oj	0.8296
prune.tree	0.8296

We have the same error.

This is suspicious. As noted above the tree call is returning the same tree regardless of the control setting. It is notable that this tree size is close to the same as that suggested by the cross validation routine. We'd like to debug and understand this further.