DSI_06- HW7 pg 363

Julia Gallucci

2023-03-05

9. This problem involves the OJ data set which is part of the ISLR2 package.

```
library(ISLR2)
attach(OJ)
head(OJ)
```

```
##   Purchase WeekofPurchase StoreID PriceCH PriceMM DiscCH DiscMM SpecialCH
## 1       CH            237       1    1.75    1.99   0.00    0.0         0
## 2       CH            239       1    1.75    1.99   0.00    0.3         0
## 3       CH            245       1    1.86    2.09   0.17    0.0         0
## 4       MM            227       1    1.69    1.69   0.00    0.0         0
## 5       CH            228       7    1.69    1.69   0.00    0.0         0
## 6       CH            230       7    1.69    1.99   0.00    0.0         0
##   SpecialMM  LoyalCH SalePriceMM SalePriceCH PriceDiff Store7 PctDiscMM
## 1         0 0.500000        1.99        1.75      0.24     No  0.000000
## 2         1 0.600000        1.69        1.75     -0.06     No  0.150754
## 3         0 0.680000        2.09        1.69      0.40     No  0.000000
## 4         0 0.400000        1.69        1.69      0.00     No  0.000000
## 5         0 0.956535        1.69        1.69      0.00    Yes  0.000000
## 6         1 0.965228        1.99        1.69      0.30    Yes  0.000000
##   PctDiscCH ListPriceDiff STORE
## 1  0.000000          0.24     1
## 2  0.000000          0.24     1
## 3  0.091398          0.23     1
## 4  0.000000          0.00     1
## 5  0.000000          0.00     0
## 6  0.000000          0.30     0
```

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

```
set.seed(123)
train = sample(1: nrow(OJ), 800) #from 1 to 1070, random sample size = 800
test = -train

train_set = OJ[train,]
test_set = OJ[test,]
```

(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?

```
library(tree)
set.seed(123)
str(OJ$Purchase) #factor w/ 2 levels "CH", "MM"
```

```
##  Factor w/ 2 levels "CH","MM": 1 1 1 2 1 1 1 1 1 1 ...
```

```
contrasts(OJ$Purchase) #CH = 0, MM = 1
```

```
##    MM
## CH  0
## MM  1
```

```
tree.fit <- tree(Purchase ~ ., data = OJ, subset = train)
summary(tree.fit)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ, subset = train)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7625 = 603.9 / 792
## Misclassification error rate: 0.165 = 132 / 800
```

Training error rate of ~0.165 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.

```
tree.fit
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1071.00 CH ( 0.60875 0.39125 )
##    2) LoyalCH < 0.5036 350  415.10 MM ( 0.28000 0.72000 )
##      4) LoyalCH < 0.276142 170  131.00 MM ( 0.12941 0.87059 )
##        8) LoyalCH < 0.0356415 56   10.03 MM ( 0.01786 0.98214 ) *
##        9) LoyalCH > 0.0356415 114  108.90 MM ( 0.18421 0.81579 ) *
##      5) LoyalCH > 0.276142 180  245.20 MM ( 0.42222 0.57778 )
##       10) PriceDiff < 0.05 74   74.61 MM ( 0.20270 0.79730 ) *
##       11) PriceDiff > 0.05 106  144.50 CH ( 0.57547 0.42453 ) *
##    3) LoyalCH > 0.5036 450  357.10 CH ( 0.86444 0.13556 )
##      6) PriceDiff < -0.39 27   32.82 MM ( 0.29630 0.70370 ) *
##      7) PriceDiff > -0.39 423  273.70 CH ( 0.90071 0.09929 )
##       14) LoyalCH < 0.705326 130  135.50 CH ( 0.78462 0.21538 )
##         28) PriceDiff < 0.145 43   58.47 CH ( 0.58140 0.41860 ) *
##         29) PriceDiff > 0.145 87   62.07 CH ( 0.88506 0.11494 ) *
##       15) LoyalCH > 0.705326 293  112.50 CH ( 0.95222 0.04778 ) *
```

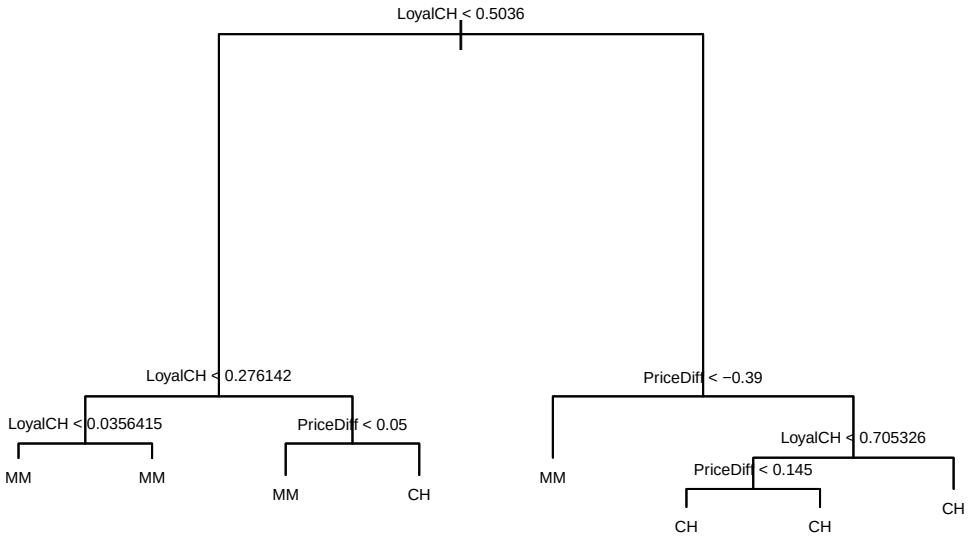
- node: a unique number for the node in the tree

- split: the equation used to branch at the node
- n: the number of observations following the left-side of the branch
- deviance: the deviance associated with that branch
- yval: predicted value at the node
- (yprob): the proportion of values in that branch that are absent and present *: an asterisk next to a node indicates it is terminal i.e, terminal node LoyalCH < 0.0356415 56 10.03 MM (0.01786 0.98214)

this means, the node is being split based on LoyalCH (Customer brand loyalty for CH) with a threshold of 0.0356415 56 observations follow the left-side of the branch 10.03 deviance is associated with this branch (0.01786 0.98214) the proportion of values in this branch that are absent and present

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

```
plot(tree.fit)
text(tree.fit, pretty = 0, cex = 0.5)
```



(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?

```
tree.pred <- predict(tree.fit, OJ[test,], type = 'class') #use our model fit on test set
cat("Confusion Matrix:")
```

```
## Confusion Matrix:
```

```
table(tree.pred, OJ[test,]$Purchase) #create a confusion matrix
```

```
##
## tree.pred  CH  MM
##        CH 150  34
##        MM  16  70
```

```
#classification error
error <- (34 + 16) / 270
cat("\n")
```

```
cat("Test error rate: \n", error)
```

```
## Test error rate:
##  0.1851852
```

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

```
set.seed(123)
cv.OJ <- cv.tree(tree.fit, FUN=prune.misclass)
cv.OJ
```

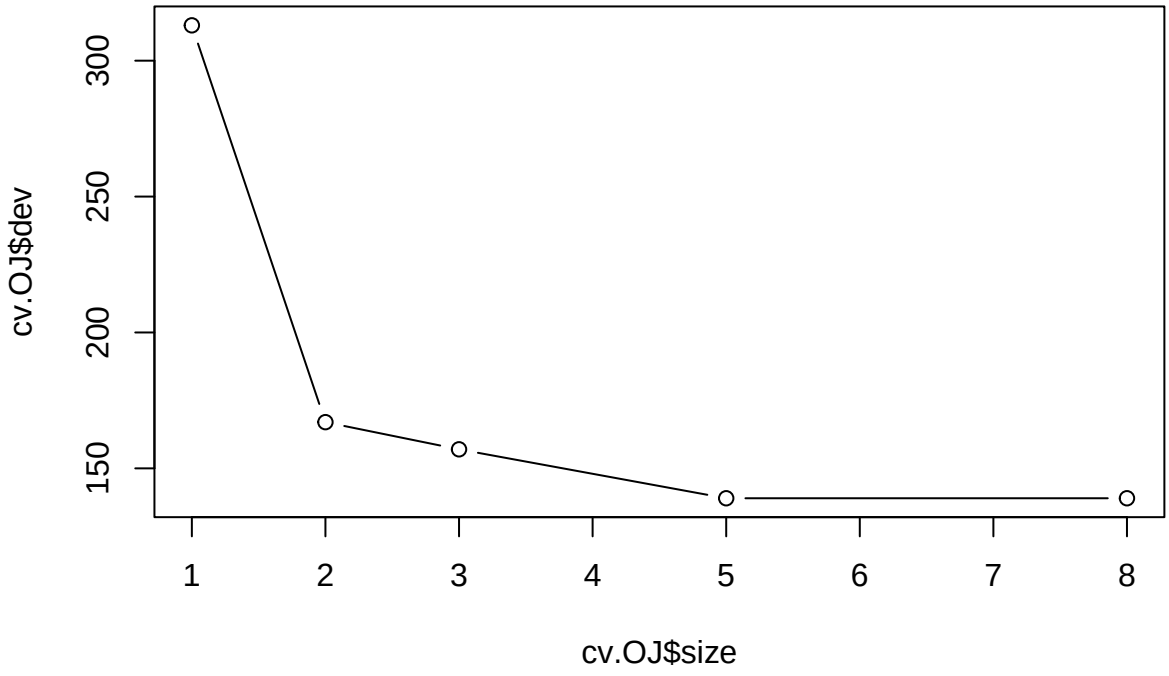
```
## $size
## [1] 8 5 3 2 1
##
## $dev
## [1] 139 139 157 167 313
##
## $k
## [1] -Inf    0    8   11  154
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
cat("Minimum classification error:", sort(cv.OJ$dev)[1])
```

```
## Minimum classification error: 139
```

(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')
```

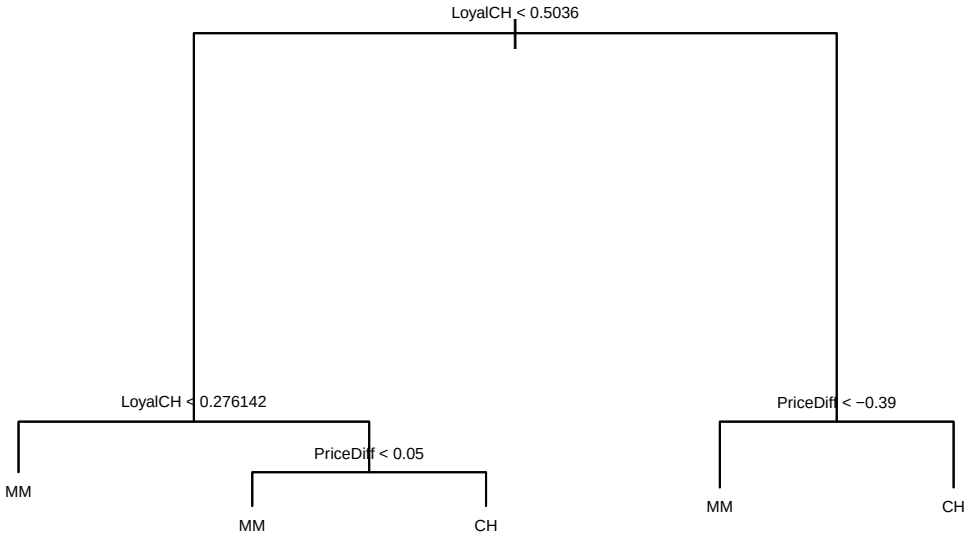


(h) Which tree size corresponds to the lowest cross-validated classification error rate?

tree size 5 or 8 correspond with the lowest cross-validation classification error rate. 8 terminal nodes was the original tree model, lets try 5!

(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_5 <- prune.misclass(tree.fit, best=5)
plot(prune.OJ_5)
text(prune.OJ_5, pretty=0, cex=.5)
```



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

```
summary(prune.OJ_5)
```

```
##
## Classification tree:
## snip.tree(tree = tree.fit, nodes = c(4L, 7L))
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  5
## Residual mean deviance:  0.826 = 656.6 / 795
## Misclassification error rate: 0.165 = 132 / 800
```

Training error rate of 0.165 - same as tree with 8 terminal nodes!

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

```
pruned_5.pred <- predict(prune.OJ_5, OJ[test,], type = 'class') #use our pruned model fit on test set
table(pruned_5.pred, OJ[test,]$Purchase)
```

```
##
## pruned_5.pred  CH  MM
##            CH 150  34
##            MM  16  70
```

```
#test classification error
error <- (34 + 16) / 270
cat("\n")
```

```
cat("Test error rate: \n", error)
```

```
## Test error rate:
##  0.1851852
```

Same test error rate as original tree!