SL HW 8

Joshua Ingram

11/20/2019

Problem 1

```r
total.error <- 0
for (i in 1:nrow(Auto)){
  test.data <- Auto[i,]
  train.data <- Auto[-i,]
  lm.train <- lm(mpg~horsepower, data = train.data)
  prediction <- predict(lm.train, newdata = test.data)
  squared.error <- (test.data$mpg - prediction)^2
  total.error <- total.error + squared.error
}
test.error.estimate <- sum(total.error) / nrow(Auto)
print("Coded LOOCV Output")
```

```
## [1] "Coded LOOCV Output"
```

```r
test.error.estimate
```

```
## [1] 24.23151
```

```r
glm.obj <- glm(mpg~horsepower, data = Auto)
print("cv.glm Output")
```

```
## [1] "cv.glm Output"
```

```r
cv.glm(Auto, glm.obj)$delta[1]
```

```
## [1] 24.23151
```

Problem 2

a.

See Figure 1

b.

See Figure 2

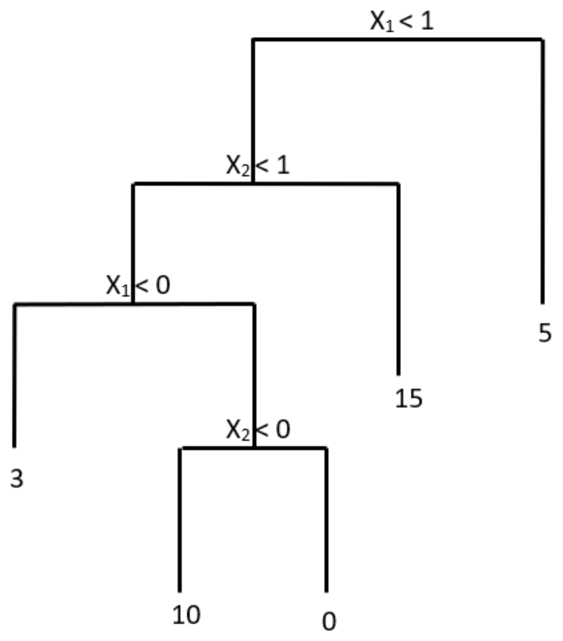


Figure 1: Corresponding Tree

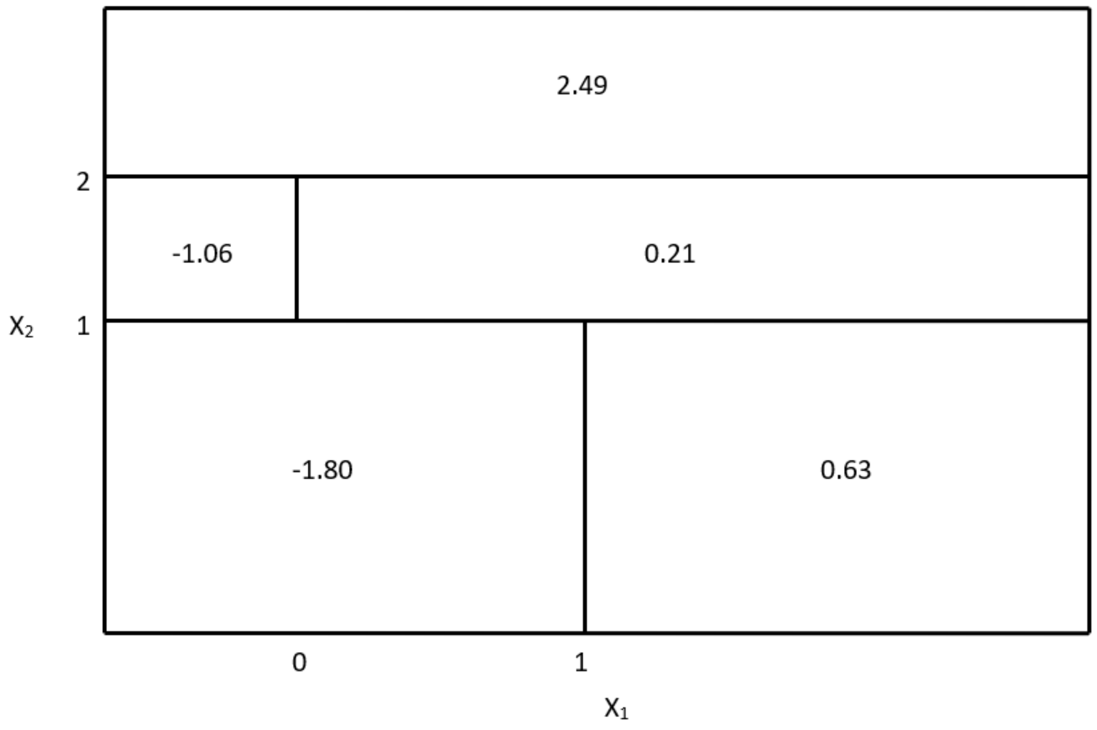


Figure 2: Corresponding Diagram

Problem 3

a.

```
fit.lm1 <- lm(Sales~., data = Carseats)
correlations <- round(cor(model.matrix(fit.lm1)[,2:12]), 2)
ifelse(abs(correlations) > 0.75, abs(correlations),0)
```

```
##                CompPrice Income Advertising Population Price
## CompPrice              1      0           0          0     0
## Income                 0      1           0          0     0
## Advertising            0      0           1          0     0
## Population             0      0           0          1     0
## Price                  0      0           0          0     1
## ShelveLocGood          0      0           0          0     0
## ShelveLocMedium        0      0           0          0     0
## Age                    0      0           0          0     0
## Education              0      0           0          0     0
## UrbanYes               0      0           0          0     0
## USYes                  0      0           0          0     0
##                ShelveLocGood ShelveLocMedium Age Education UrbanYes USYes
## CompPrice                  0               0   0         0        0     0
## Income                     0               0   0         0        0     0
## Advertising                0               0   0         0        0     0
## Population                 0               0   0         0        0     0
## Price                      0               0   0         0        0     0
## ShelveLocGood              1               0   0         0        0     0
## ShelveLocMedium            0               1   0         0        0     0
## Age                        0               0   1         0        0     0
## Education                  0               0   0         1        0     0
## UrbanYes                   0               0   0         0        1     0
## USYes                      0               0   0         0        0     1
```

There does not seem to be any collinearity by looking through a correlation matrix. (no correlations are greater than 0.75)

```
vif(fit.lm1)
```

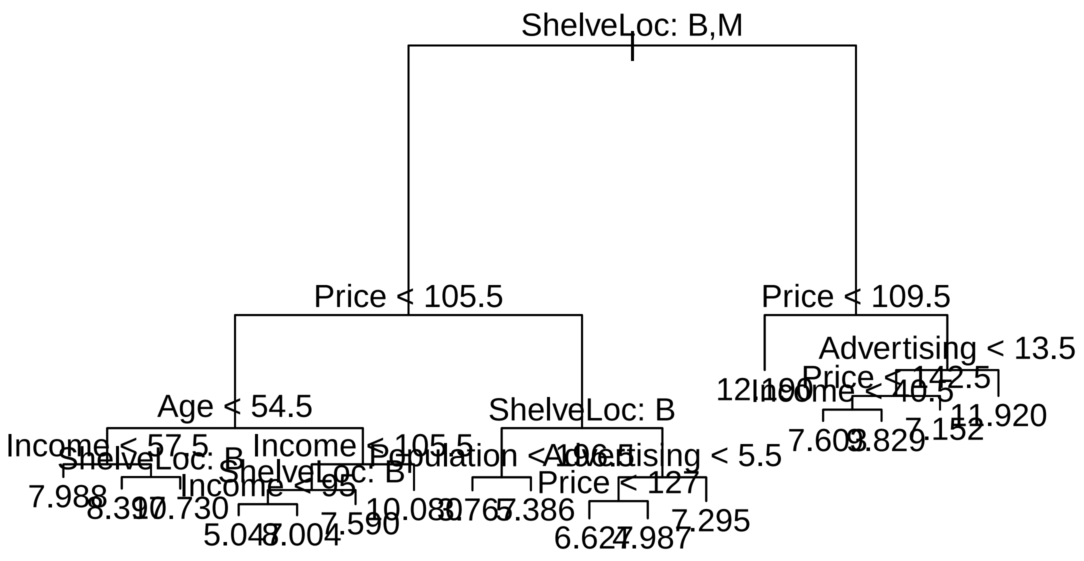
```
##                 GVIF Df GVIF^(1/(2*Df))
## CompPrice   1.554618  1        1.246843
## Income      1.024731  1        1.012290
## Advertising 2.103136  1        1.450219
## Population  1.145534  1        1.070296
## Price       1.537068  1        1.239785
## ShelveLoc   1.033891  2        1.008367
## Age         1.021051  1        1.010471
## Education   1.026342  1        1.013086
## Urban       1.022705  1        1.011289
## US          1.980720  1        1.407380
```

Checked for multi-collinearity using the Variance Inflation Factor. No variables have a VIF greater than 5.

b.

We should use regression trees for our approach since sales is a quantitative response

```r
tree.obj1 <- tree(Sales~., data = Carseats)
plot(tree.obj1)
text(tree.obj1, pretty = T)
```



```r
summary(tree.obj1)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Income"      "Population"
## [6] "Advertising"
## Number of terminal nodes:  17
## Residual mean deviance:  2.878 = 1102 / 383
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.98700 -1.23000 -0.06125  0.00000  1.22500  4.75400
```

“ShelveLoc,” “Price,” “Age,” “Income,” “Population,” and “Advertising” are the only variables used in the tree out of the 12 in the dataset. The most used appear to be “Income” and “ShelveLoc.”

c.

```r
set.seed(1)
cv.obj <- cv.tree(tree.obj1, K=nrow(Carseats))
cv.obj
```

```
## $size
##  [1] 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
##
## $dev
##  [1] 2030.390 2061.509 2047.356 2029.884 2099.475 2092.817 2079.069
##  [8] 2065.729 2059.700 2187.701 2006.717 1991.989 1989.914 2143.442
## [15] 2184.032 2733.888 3635.893
##
## $k
##  [1]      -Inf  32.78204  33.43341  34.30000  37.83019  38.65535  40.44960
##  [8]  41.83218  51.05171  70.52963  76.20847  76.57441 106.90014 145.33849
## [15] 162.67977 334.36974 797.19286
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

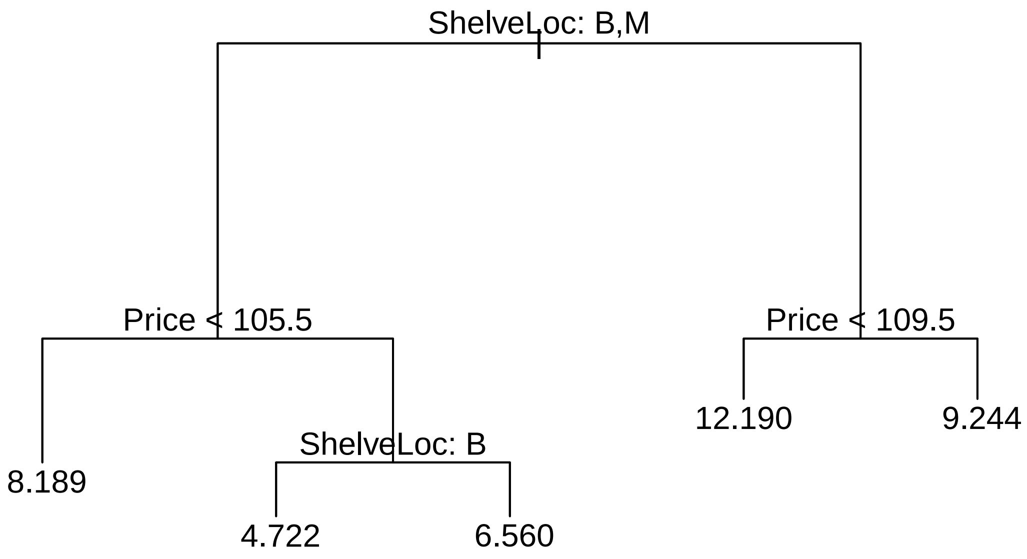
```r
cv.obj$size[which.min(cv.obj$dev)]
```

```
## [1] 5
```

```r
prune.sizes <- prune.tree(tree.obj1)$size
subtree.obj <- prune.tree(tree.obj1, best=prune.sizes[13])
summary(subtree.obj)
```

```
##
## Regression tree:
## snip.tree(tree = tree.obj1, nodes = c(10L, 11L, 7L, 4L))
## Variables actually used in tree construction:
## [1] "ShelveLoc" "Price"
## Number of terminal nodes:  5
## Residual mean deviance:  4.412 = 1743 / 395
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -6.56000 -1.45300 -0.08439  0.00000  1.36300  5.41800
```

```r
plot(subtree.obj)
text(subtree.obj, pretty = T)
```



```
MSE = 1989.914/nrow(Carseats)
MSE
```

```
## [1] 4.974785
```

optimal subtree: 10 nodes

test error: SSE = 1989.914, MSE = 4.974785

most important predictors “ShelveLoc” and “Price”

d.

```
subtree.obj
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 400 3182.00  7.496
##    2) ShelveLoc: Bad,Medium 315 1860.00  6.763
##      4) Price < 105.5 108  568.60  8.189 *
##      5) Price > 105.5 207  956.60  6.019
##       10) ShelveLoc: Bad 61  240.80  4.722 *
##       11) ShelveLoc: Medium 146  570.40  6.560 *
##    3) ShelveLoc: Good 85  525.50 10.210
##      6) Price < 109.5 28   85.58 12.190 *
##      7) Price > 109.5 57  277.30  9.244 *
```

"15) Advertising > 13.5 9 15.2700 11.920" The "9" is the number of observations between the two outcomes. 15.27 is the sum of square errors for an advertising value above 13.5 and 11.92 is the average sale value for an advertising value above 13.5

Problem 4

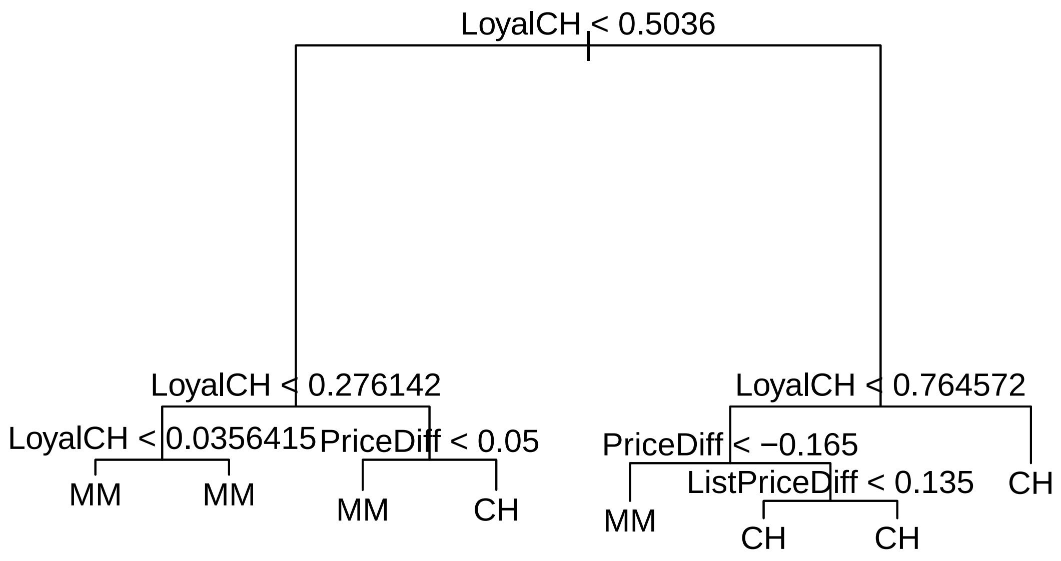
a.

Classification tree seems to be most appropriate here since the response is categorical.

```
tree.obj2 <- tree(Purchase~., data = OJ)
summary(tree.obj2)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "ListPriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7571 = 804 / 1062
## Misclassification error rate: 0.1636 = 175 / 1070
```

```
plot(tree.obj2)
text(tree.obj2, pretty = T)
```



b.

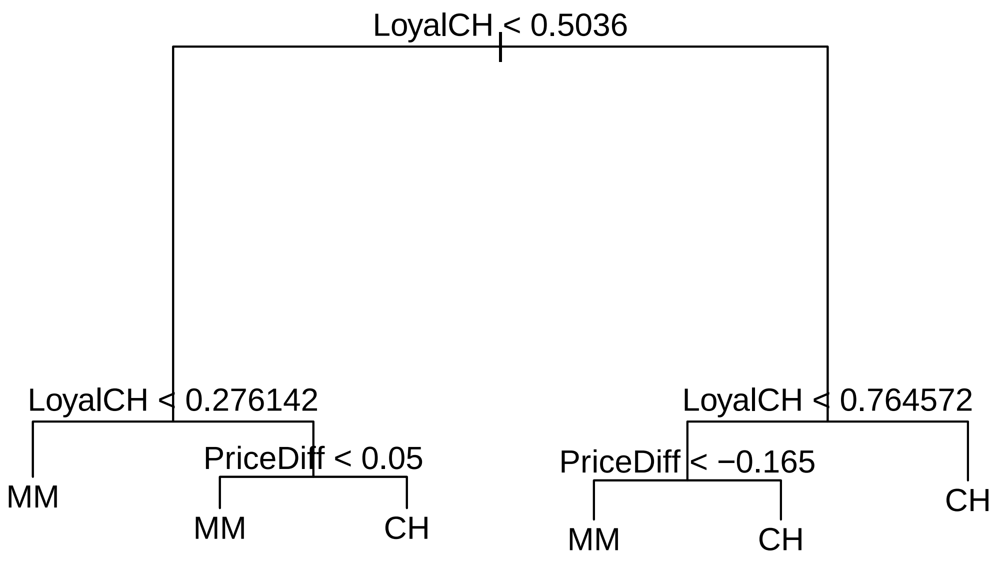
```
set.seed(1)
cv.obj2 <- cv.tree(tree.obj2, FUN = prune.misclass, K = nrow(OJ))
cv.obj2
```

```
## $size
## [1] 8 6 4 2 1
##
## $dev
## [1] 208 208 257 285 499
##
## $k
## [1]  -Inf   0.0   8.0  11.5 203.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
subtree.obj2 <- prune.misclass(tree.obj2, best = 6)
summary(subtree.obj2)
```

```
##
## Classification tree:
## snip.tree(tree = tree.obj2, nodes = c(4L, 13L))
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  6
## Residual mean deviance:  0.789 = 839.5 / 1064
## Misclassification error rate: 0.1636 = 175 / 1070
```

```
plot(subtree.obj2)
text(subtree.obj2, pretty = T)
```



```
miss.class.rate <- 208 / nrow(OJ)
miss.class.rate
```

```
## [1] 0.1943925
```

Trees with 8 nodes and 6 nodes has the same missclassification rate, so decided to select the 6 node subtree for the optimal size to avoid over-fitting.

test missclasification error rate: .1943925

“LoyalCH” and “PriceDiff” are the most important predictors

c.

```
subtree.obj2
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1070 1431.00 CH ( 0.61028 0.38972 )
##    2) LoyalCH < 0.5036 469  559.30 MM ( 0.28358 0.71642 )
##      4) LoyalCH < 0.276142 223  164.60 MM ( 0.12108 0.87892 ) *
##      5) LoyalCH > 0.276142 246  336.30 MM ( 0.43089 0.56911 )
##       10) PriceDiff < 0.05 101  105.90 MM ( 0.21782 0.78218 ) *
##       11) PriceDiff > 0.05 145  197.30 CH ( 0.57931 0.42069 ) *
##    3) LoyalCH > 0.5036 601  475.20 CH ( 0.86522 0.13478 )
##      6) LoyalCH < 0.764572 251  289.20 CH ( 0.73705 0.26295 )
##       12) PriceDiff < -0.165 40   48.87 MM ( 0.30000 0.70000 ) *
##       13) PriceDiff > -0.165 211  199.00 CH ( 0.81991 0.18009 ) *
##      7) LoyalCH > 0.764572 350  123.80 CH ( 0.95714 0.04286 ) *
```

Node 7): there are 350 observations within this region and there is an entropy value of 123.8 (95.7% CH response) and the response will be predicted to be "CH"

d.

Within in the 6 node tree there are no splits that lead to the same predicted class, but the original tree, tree.obj2, has two splits that have the same predicted class. These splits are made because in the creation of a tree, the algorithm attempts to minimize the overall entropy (and lead towards node purity). When splits with the same predictions are made, there is a decrease in entropy by creating new regions as there is an improved node purity for at least one of the new nodes which means there is more certainty in the prediction.