STAE03: Business Analytics

Assignment 4

Diana (990820T222)

1. Introduction

This assignment is divided into two parts. Both parts analysed the Carseats data from ISLR library. The data provides information related to sales of child car seats at 400 different stores. It contains 400 observations and 11 variables: Sales, CompPrice, Income, Advertising, Population, Price, ShelveLoc, Age, Education, Urban, Store, and US Store. The response variable is Sales and one utilized all other variables as the predictor variables. One utilized set.seed(0820) and divided the dataset into training and test data with a ratio of 3:1. The first part is aimed at predicting whether the Sales is high or not by using Classification trees. The second part is aimed at predicting Sales by using Regression trees.

2. Part 1: Classification Trees

One created a new variable which is called as High. It assigns Sales data which are higher than 8 to "Yes", otherwise "No". One attempted to predict High variable with all predictor variables using Classification Trees. One obtained the results as follows:

```
Classification tree:
tree(formula = High ~ . - Sales, data = Carseats, subset = train)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"      "Age"         "CompPrice"   "US"          "Advertising"
[8] "Education"
Number of terminal nodes:  23
Residual mean deviance:  0.3459 = 95.83 / 277
Misclassification error rate: 0.07333 = 22 / 300
```

Figure 2.1: Classification Tree result

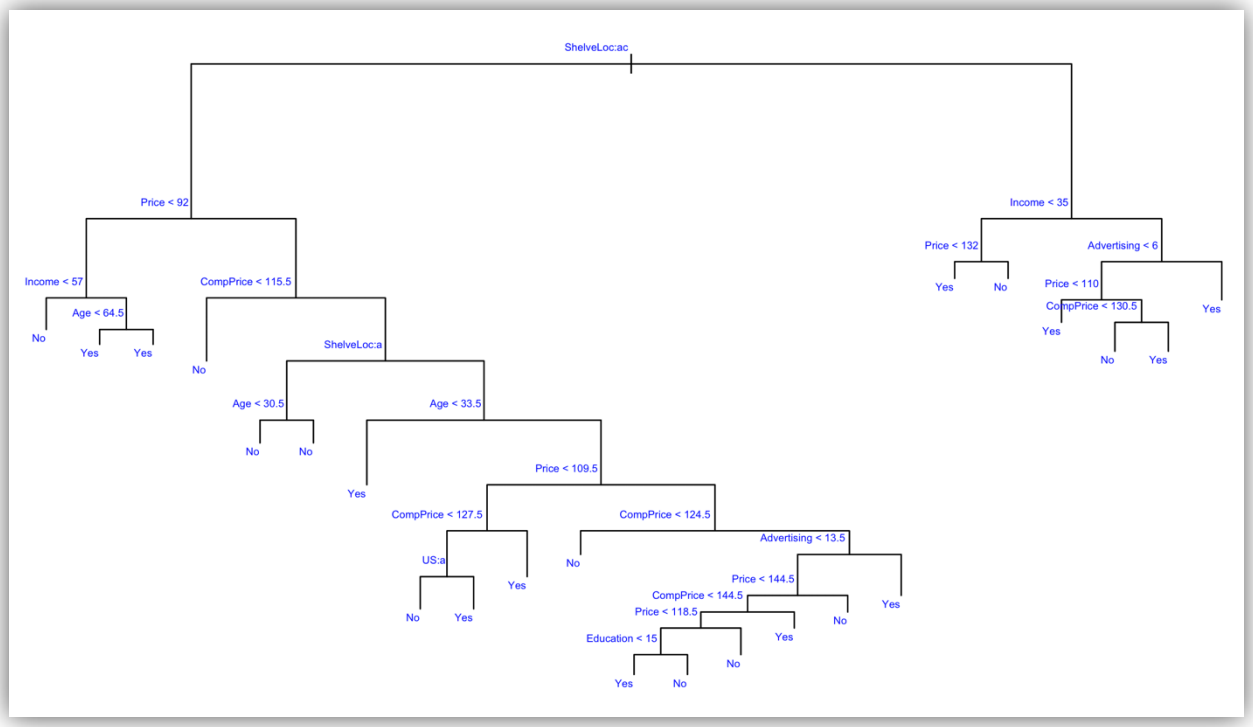

Figure 2.2: The construction of the tree

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

   1) root 300 406.800 No ( 0.58667 0.41333 )
     2) ShelveLoc: Bad,Medium 230 281.000 No ( 0.70000 0.30000 )
       4) Price < 92 37  46.630 Yes ( 0.32432 0.67568 )
         8) Income < 57 9   9.535 No ( 0.77778 0.22222 ) *
         9) Income > 57 28  26.280 Yes ( 0.17857 0.82143 )
          18) Age < 64.5 18   7.724 Yes ( 0.05556 0.94444 ) *
          19) Age > 64.5 10  13.460 Yes ( 0.40000 0.60000 ) *
       5) Price > 92 193 207.200 No ( 0.77202 0.22798 )
        10) CompPrice < 115.5 37   0.000 No ( 1.00000 0.00000 ) *
        11) CompPrice > 115.5 156 185.600 No ( 0.71795 0.28205 )
          22) ShelveLoc: Bad 43  16.180 No ( 0.95349 0.04651 )
            44) Age < 30.5 7   8.376 No ( 0.71429 0.28571 ) *
            45) Age > 30.5 36   0.000 No ( 1.00000 0.00000 ) *
          23) ShelveLoc: Medium 113 149.100 No ( 0.62832 0.37168 )
            46) Age < 33.5 14   7.205 Yes ( 0.07143 0.92857 ) *
            47) Age > 33.5 99 119.700 No ( 0.70707 0.29293 )
              94) Price < 109.5 24  31.760 Yes ( 0.37500 0.62500 )
               188) CompPrice < 127.5 13  16.050 No ( 0.69231 0.30769 )
                 376) US: No 8   0.000 No ( 1.00000 0.00000 ) *
                 377) US: Yes 5   5.004 Yes ( 0.20000 0.80000 ) *
               189) CompPrice > 127.5 11   0.000 Yes ( 0.00000 1.00000 ) *
              95) Price > 109.5 75  72.200 No ( 0.81333 0.18667 )
               190) CompPrice < 124.5 17   0.000 No ( 1.00000 0.00000 ) *
               191) CompPrice > 124.5 58  64.110 No ( 0.75862 0.24138 )
                 382) Advertising < 13.5 51  44.310 No ( 0.84314 0.15686 )
                   764) Price < 144.5 37  38.630 No ( 0.78378 0.21622 )
                    1528) CompPrice < 144.5 30  23.560 No ( 0.86667 0.13333 )
                      3056) Price < 118.5 11  14.420 No ( 0.63636 0.36364 )
                        6112) Education < 15 6   7.638 Yes ( 0.33333 0.66667 ) *
                        6113) Education > 15 5   0.000 No ( 1.00000 0.00000 ) *
                      3057) Price > 118.5 19   0.000 No ( 1.00000 0.00000 ) *
                    1529) CompPrice > 144.5 7   9.561 Yes ( 0.42857 0.57143 ) *
                   765) Price > 144.5 14   0.000 No ( 1.00000 0.00000 ) *
                 383) Advertising > 13.5 7   5.742 Yes ( 0.14286 0.85714 ) *
     3) ShelveLoc: Good 70  72.740 Yes ( 0.21429 0.78571 )
       6) Income < 35 12  15.280 No ( 0.66667 0.33333 )
        12) Price < 132 7   9.561 Yes ( 0.42857 0.57143 ) *
        13) Price > 132 5   0.000 No ( 1.00000 0.00000 ) *
       7) Income > 35 58  42.720 Yes ( 0.12069 0.87931 )
        14) Advertising < 6 25  29.650 Yes ( 0.28000 0.72000 )
          28) Price < 110 9   0.000 Yes ( 0.00000 1.00000 ) *
          29) Price > 110 16  21.930 Yes ( 0.43750 0.56250 )
            58) CompPrice < 130.5 7   5.742 No ( 0.85714 0.14286 ) *
            59) CompPrice > 130.5 9   6.279 Yes ( 0.11111 0.88889 ) *
        15) Advertising > 6 33   0.000 Yes ( 0.00000 1.00000 ) *
```

Figure 2.3: Detailed construction of the tree

```
> mean(cstree.pred == Hightest)
[1] 0.77
```

Figure 2.4: Percentage of correct prediction for test data

```
            Hightest
cstree.pred No Yes
        No  49  12
        Yes 11  28
```

```
> sens
[1] 0.7
> spec
[1] 0.8166667
```

Figure 2.5: Sensitivity and specificity

There are 8 predictor variables that are included in the construction of the tree. The other 2 predictor variables, Population and Urban, are omitted in the tree. The number of terminal nodes is 23, which is considered as a pretty high number. One observed that there are several branches that are not necessary because they yield the same leaves. For example, when ShelveLoc:ac, Price < 92, and Income > 57, regardless of the average age of the local population, Sales are always considered as high. This suggests that there is an opportunity to cut down several leaves through pruning.

The Misclassification error rate is 0.0733 which means that the percentage of correct prediction for training data is 92.67%. However, when one attempted to fit this tree to the test data, one only obtained a percentage of correct prediction of 77%. The percentage of correct prediction drops from 92.67% to 77% and this is definitely considered as a significant drop. But one must understand that there is a difference in the number of training and test data. 92.67% of correct prediction in training data implies that there are 22 out of 300 observations that are misclassified. 77% of correct prediction in test data implies that there are 23 out of 100 observations that are misclassified. Furthermore, sensitivity and specificity are both quite high. This shows that despite the significant drop in percentage of correct prediction, the fitted classification tree still performed well in this dataset.

In order to prune the tree, one performed a 10-fold cross validation and obtained the following result:

```
$size
 [1] 23 21 17 16 13 11 10  9  4  3  2  1

$dev
 [1]  74  74  72  73  72  77  75  79  80  78  86 124

$k
 [1]      -Inf  0.000000  0.750000  1.000000  1.666667  2.500000  3.000000  4.000000  4.600000  5.000000
[11] 13.000000 40.000000

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```

Figure 2.6: 10-fold cross validation result

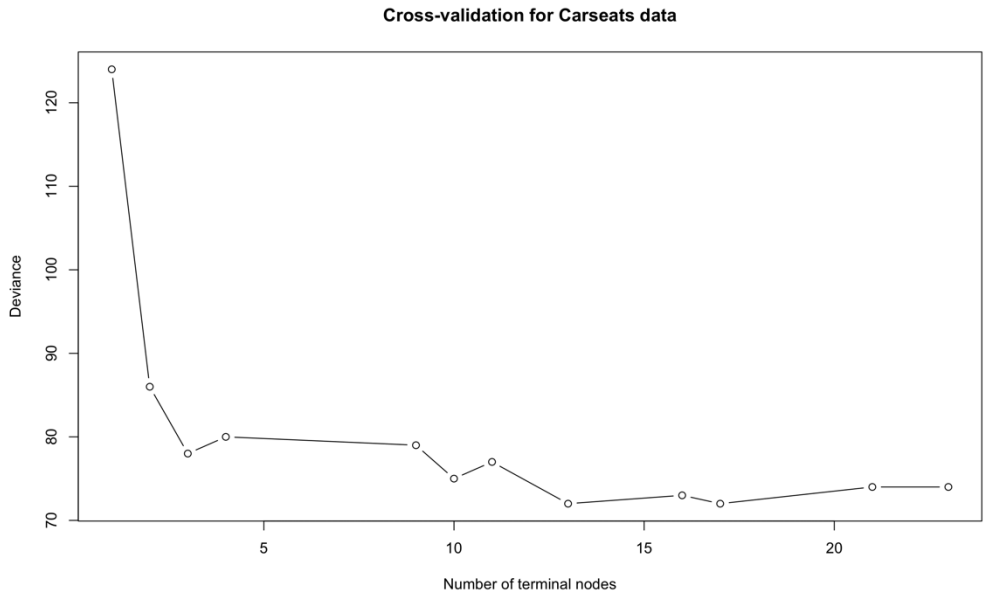

Figure 2.7: Deviance vs Number of terminal nodes plot

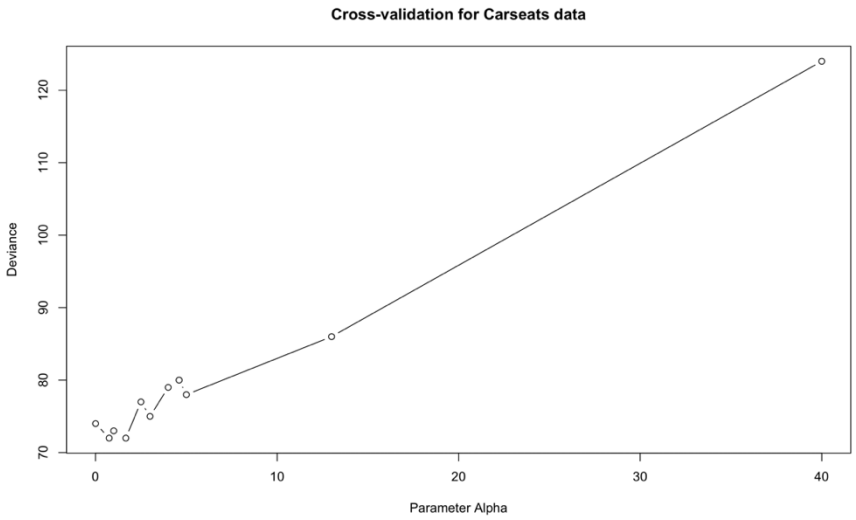

Figure 2.8: Deviance vs Alpha plot

From the results above, one could observe that the deviance is lowest when terminal nodes = 17 or 13 and alpha = 0.75 or 1.67. Thus, one proceed with pruning the tree to 17 terminal nodes and 13 terminal nodes in order to decide which would be the best fit to Carseats dataset.

2.1. Terminal nodes = 17.

```
Classification tree:
snip.tree(tree = cstree, nodes = c(9L, 22L, 382L))
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Age"         "US"          "Advertising"
Number of terminal nodes:  17
Residual mean deviance:  0.48 = 135.8 / 283
Misclassification error rate: 0.08333 = 25 / 300
```

Figure 2.1.1: Pruning with 17 terminal nodes result

```
> mean(prunecstree.pred == Hightest)
[1] 0.77
```

Figure 2.1.2: Percentage of correct prediction

```
> prune.table
                 Hightest
prunecstree.pred No Yes
              No  51  14
              Yes  9  26
```

```
> prune.sens
[1] 0.65
> prune.spec
[1] 0.85
```

Figure 2.1.3: Sensitivity and Specificity

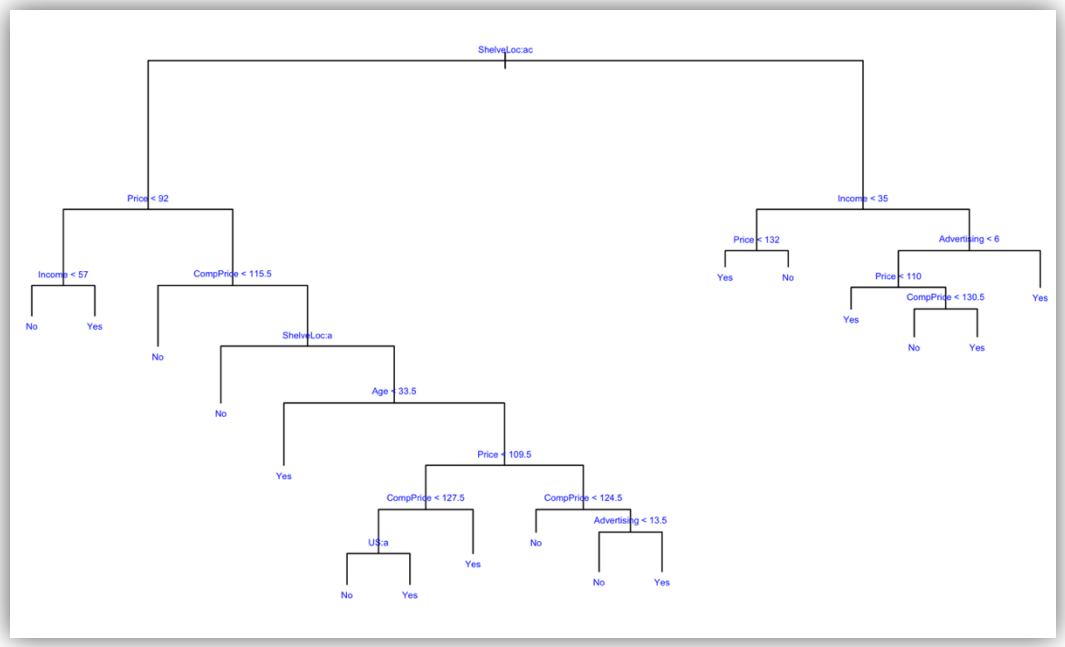

Figure 2.1.4: Construction of tree with 17 terminal nodes

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 300 406.800 No ( 0.58667 0.41333 )
   2) ShelveLoc: Bad,Medium 230 281.000 No ( 0.70000 0.30000 )
     4) Price < 92 37  46.630 Yes ( 0.32432 0.67568 )
       8) Income < 57 9   9.535 No ( 0.77778 0.22222 ) *
       9) Income > 57 28  26.280 Yes ( 0.17857 0.82143 ) *
     5) Price > 92 193 207.200 No ( 0.77202 0.22798 )
      10) CompPrice < 115.5 37   0.000 No ( 1.00000 0.00000 ) *
      11) CompPrice > 115.5 156 185.600 No ( 0.71795 0.28205 )
        22) ShelveLoc: Bad 43  16.180 No ( 0.95349 0.04651 ) *
        23) ShelveLoc: Medium 113 149.100 No ( 0.62832 0.37168 )
          46) Age < 33.5 14   7.205 Yes ( 0.07143 0.92857 ) *
          47) Age > 33.5 99 119.700 No ( 0.70707 0.29293 )
            94) Price < 109.5 24  31.760 Yes ( 0.37500 0.62500 )
             188) CompPrice < 127.5 13  16.050 No ( 0.69231 0.30769 )
               376) US: No 8   0.000 No ( 1.00000 0.00000 ) *
               377) US: Yes 5   5.004 Yes ( 0.20000 0.80000 ) *
             189) CompPrice > 127.5 11   0.000 Yes ( 0.00000 1.00000 ) *
            95) Price > 109.5 75  72.200 No ( 0.81333 0.18667 )
             190) CompPrice < 124.5 17   0.000 No ( 1.00000 0.00000 ) *
             191) CompPrice > 124.5 58  64.110 No ( 0.75862 0.24138 )
               382) Advertising < 13.5 51  44.310 No ( 0.84314 0.15686 ) *
               383) Advertising > 13.5 7   5.742 Yes ( 0.14286 0.85714 ) *
   3) ShelveLoc: Good 70  72.740 Yes ( 0.21429 0.78571 )
     6) Income < 35 12  15.280 No ( 0.66667 0.33333 )
      12) Price < 132 7   9.561 Yes ( 0.42857 0.57143 ) *
      13) Price > 132 5   0.000 No ( 1.00000 0.00000 ) *
     7) Income > 35 58  42.720 Yes ( 0.12069 0.87931 )
      14) Advertising < 6 25  29.650 Yes ( 0.28000 0.72000 )
        28) Price < 110 9   0.000 Yes ( 0.00000 1.00000 ) *
        29) Price > 110 16  21.930 Yes ( 0.43750 0.56250 )
          58) CompPrice < 130.5 7   5.742 No ( 0.85714 0.14286 ) *
          59) CompPrice > 130.5 9   6.279 Yes ( 0.11111 0.88889 ) *
      15) Advertising > 6 33   0.000 Yes ( 0.00000 1.00000 ) *
```

Figure 2.1.5: Detailed construction of tree with 17 terminal nodes

2.2. Terminal nodes = 13.

```
Classification tree:
snip.tree(tree = cstree, nodes = c(9L, 22L, 382L, 6L, 7L))
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Age"         "US"          "Advertising"
Number of terminal nodes:  13
Residual mean deviance:  0.6002 = 172.3 / 287
Misclassification error rate: 0.1033 = 31 / 300
```

Figure 2.2.1: Pruning with 13 terminal nodes result

```
> mean(prunecstree.pred == Hightest)
[1] 0.73
```

Figure 2.2.2: Percentage of correct prediction

```
> prune.table
                 Hightest
prunecstree.pred No Yes
              No  50  17
              Yes 10  23
```

```
> prune.sens
[1] 0.575
> prune.spec
[1] 0.8333333
```

Figure 2.2.3: Sensitivity and specificity

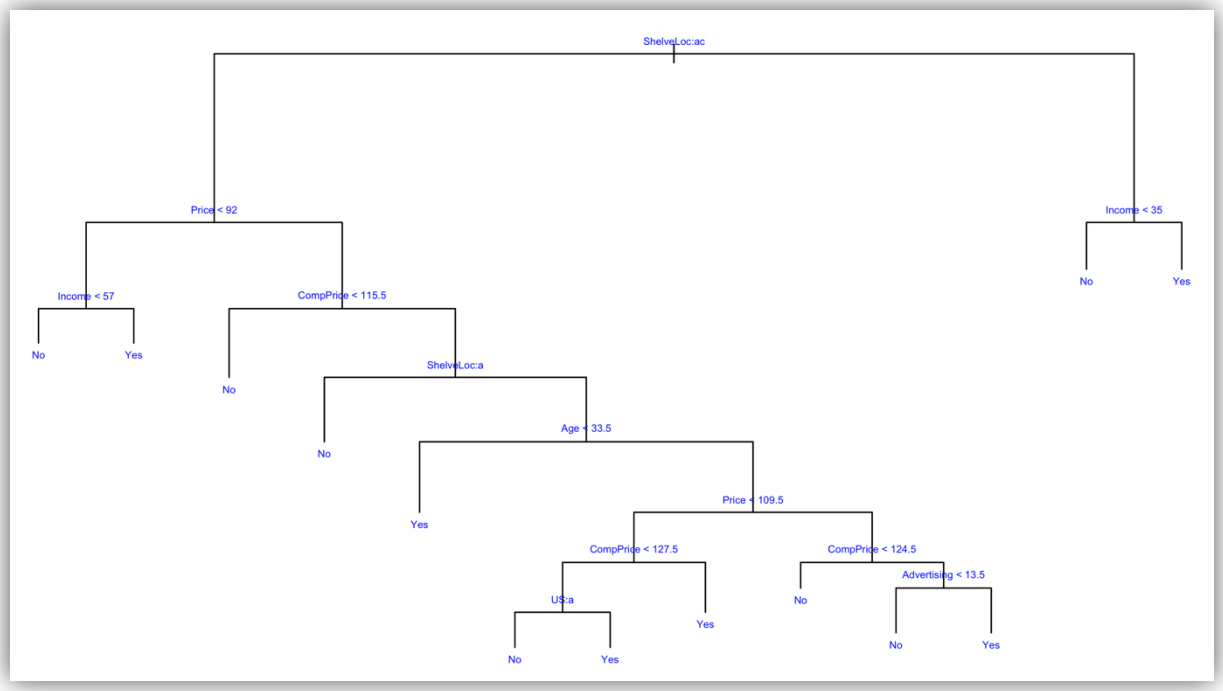

Figure 2.2.4: Construction of tree for 13 terminal nodes

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

  1) root 300 406.800 No ( 0.58667 0.41333 )
    2) ShelveLoc: Bad,Medium 230 281.000 No ( 0.70000 0.30000 )
      4) Price < 92 37  46.630 Yes ( 0.32432 0.67568 )
        8) Income < 57 9   9.535 No ( 0.77778 0.22222 ) *
        9) Income > 57 28  26.280 Yes ( 0.17857 0.82143 ) *
      5) Price > 92 193 207.200 No ( 0.77202 0.22798 )
       10) CompPrice < 115.5 37   0.000 No ( 1.00000 0.00000 ) *
       11) CompPrice > 115.5 156 185.600 No ( 0.71795 0.28205 )
         22) ShelveLoc: Bad 43  16.180 No ( 0.95349 0.04651 ) *
         23) ShelveLoc: Medium 113 149.100 No ( 0.62832 0.37168 )
           46) Age < 33.5 14   7.205 Yes ( 0.07143 0.92857 ) *
           47) Age > 33.5 99 119.700 No ( 0.70707 0.29293 )
             94) Price < 109.5 24  31.760 Yes ( 0.37500 0.62500 )
              188) CompPrice < 127.5 13  16.050 No ( 0.69231 0.30769 )
                376) US: No 8   0.000 No ( 1.00000 0.00000 ) *
                377) US: Yes 5   5.004 Yes ( 0.20000 0.80000 ) *
              189) CompPrice > 127.5 11   0.000 Yes ( 0.00000 1.00000 ) *
             95) Price > 109.5 75  72.200 No ( 0.81333 0.18667 )
              190) CompPrice < 124.5 17   0.000 No ( 1.00000 0.00000 ) *
              191) CompPrice > 124.5 58  64.110 No ( 0.75862 0.24138 )
                382) Advertising < 13.5 51  44.310 No ( 0.84314 0.15686 ) *
                383) Advertising > 13.5 7   5.742 Yes ( 0.14286 0.85714 ) *
    3) ShelveLoc: Good 70  72.740 Yes ( 0.21429 0.78571 )
      6) Income < 35 12  15.280 No ( 0.66667 0.33333 ) *
      7) Income > 35 58  42.720 Yes ( 0.12069 0.87931 ) *
```

Figure 2.2.5: Detailed construction of tree for 13 terminal nodes

2.3. Comparison of results

Terminal nodes	Percentage of correct prediction (train)	Percentage of correct prediction (test)	Sensitivity	Specificity
23	92.67%	77%	0.70	0.82
17	91.67%	77%	0.65	0.85
13	89.67%	73%	0.58	0.83

By comparing all terminal nodes above, one observed that they yield approximately similar results. One strongly believes that in order to achieve the best fit for the Carseats data, one should consider the most efficient tree that eliminates all unnecessary leaves and has the highest possible percentage of correct prediction, sensitivity and specificity. When terminal nodes = 23, there are still some non-essential terminal nodes. After the tree was pruned, one observed that terminal nodes = 13 or 17 gave the lowest deviance. However, terminal nodes = 17 attained a better result in terms of percentage of correct prediction for both test and train data, sensitivity and specificity. This indicates that one should utilize Classification tree with 17 terminal nodes to predict whether the Sales is high or not.

Initially, there are 8 variables that are utilized to predict High. After the pruning to 17 terminal nodes, one would observe that Education is no longer included in the set of predictor variables. Thus, the pruning has also reduced the number of predictor variables from 8 variables to 7 variables. This suggests that Education is not a significant predictor variable to predict whether sales of child car seats are high or not.

3. Part 2: Regression Tree

```
Regression tree:
tree(formula = Sales ~ . - High, data = Carseats, subset = train)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Age"         "Income"      "CompPrice"   "Advertising" "US"
Number of terminal nodes:  17
Residual mean deviance:  2.498 = 706.9 / 283
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-5.16500 -1.02400 -0.04479  0.00000  0.97860  4.10200
```

Figure 3.1: Regression Tree result

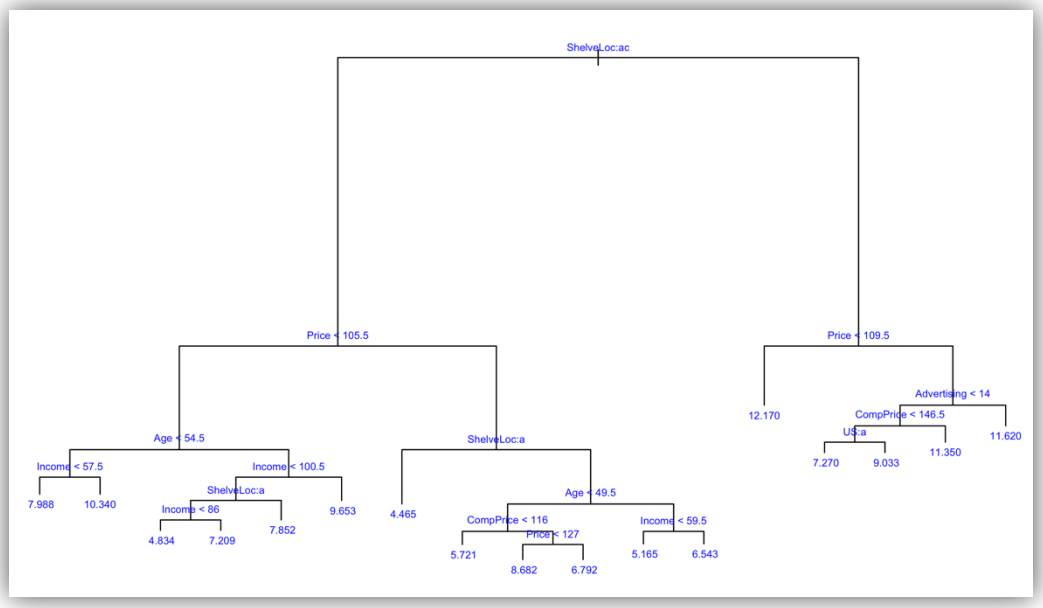

Figure 3.2: Construction of tree

```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 300 2371.000  7.526
   2) ShelveLoc: Bad,Medium 230 1290.000  6.704
     4) Price < 105.5 80  386.900  8.102
       8) Age < 54.5 29  100.900  9.284
        16) Income < 57.5 13   19.240  7.988 *
        17) Income > 57.5 16   42.100 10.340 *
       9) Age > 54.5 51  222.400  7.430
        18) Income < 100.5 42  149.900  6.954
          36) ShelveLoc: Bad 18   67.650  5.757
            72) Income < 86 11   32.310  4.834 *
            73) Income > 86 7   11.210  7.209 *
          37) ShelveLoc: Medium 24   37.160  7.852 *
        19) Income > 100.5 9   18.440  9.653 *
     5) Price > 105.5 150  663.600  5.958
      10) ShelveLoc: Bad 41  122.500  4.465 *
      11) ShelveLoc: Medium 109  415.400  6.519
        22) Age < 49.5 48  158.600  7.379
          44) CompPrice < 116 9   20.550  5.721 *
          45) CompPrice > 116 39  107.600  7.761
            90) Price < 127 20   23.240  8.682 *
            91) Price > 127 19   49.540  6.792 *
        23) Age > 49.5 61  193.500  5.842
          46) Income < 59.5 31   86.760  5.165 *
          47) Income > 59.5 30   77.720  6.543 *
   3) ShelveLoc: Good 70  412.800 10.230
     6) Price < 109.5 24   80.860 12.170 *
     7) Price > 109.5 46  194.800  9.219
      14) Advertising < 14 39  135.100  8.788
        28) CompPrice < 146.5 34   95.720  8.411
          56) US: No 12   21.080  7.270 *
          57) US: Yes 22   50.520  9.033 *
        29) CompPrice > 146.5 5    1.647 11.350 *
      15) Advertising > 14 7   12.020 11.620 *
```

Figure 3.3: Detailed Construction of tree

```
> mean((pred - cars.test)^2)
[1] 4.592989
> mean((pred - cars.train)^2)
[1] 12.64595
```

Figure 3.4: Mean squared errors for both training and test data

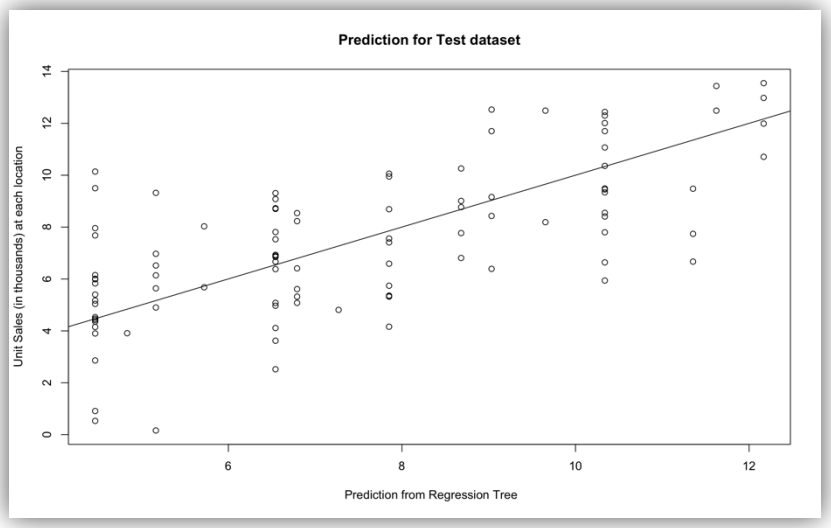

Figure 3.5: Prediction for test dataset

Out of 10 predictor variables, only 7 variables are actually used in constructing the tree. One noticed that the 7 predictor variables here are the same with the 7 predictor variables used in the Classification tree after pruning. This further proves that Education, Urban, and Population are non-significant variables and other variables are significant in predicting sales of child car seats. The residual mean deviance is 2.498, which is considered as quite low. Furthermore, one observed that the mean squared errors for test data is lower than for training data. In order to obtain the best fit to the Carseats dataset, one explore the possibilities to prune the tree because one believe that there are still opportunities to minimize the mean squared errors for both test and training data.

3.1. Cross-validation and Pruning

In order to decide the best number of terminal nodes, one performed a 10-fold cross validation and following are the results of the cross-validation:

```
$size
 [1] 17 15 14 12 11 10  9  8  7  6  5  4  3  2  1

$dev
 [1] 1521.856 1466.398 1382.764 1395.101 1391.933 1432.767 1403.769 1405.285 1373.258 1364.170 1364.170
[12] 1537.076 1540.470 1744.470 2383.005

$k
 [1]      -Inf  24.12655  28.97053  32.62487  37.76473  39.60317  45.12032  47.69610  54.00318  63.38314
[11]  63.52995 125.66830 137.11772 240.01804 667.49298

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

Figure 3.1.1: 10-fold cross validation result

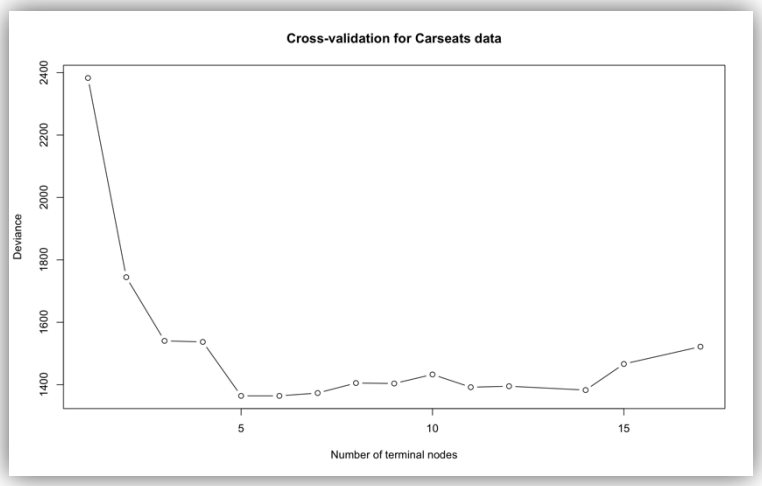

Figure 3.1.2: Deviance vs Number of terminal nodes

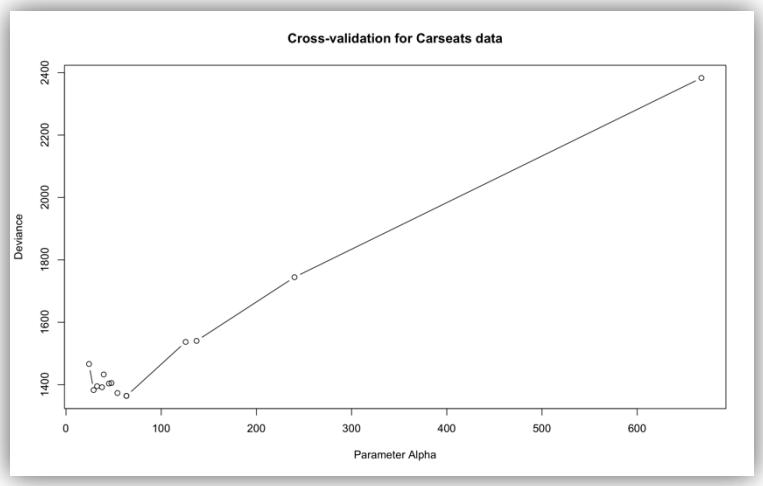

Figure 3.1.3: Deviance vs Parameter Alpha

After performing 10-fold cross validation, one noticed that deviance is lowest when number of terminal nodes is 5 and alpha is 54. This implies that one should prune the tree to 5 terminal nodes.

```
Regression tree:
snip.tree(tree = tree.cars, nodes = c(7L, 11L, 4L))
Variables actually used in tree construction:
[1] "ShelveLoc" "Price"
Number of terminal nodes:  5
Residual mean deviance:  4.069 = 1200 / 295
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-6.51900 -1.32000 -0.09564  0.00000  1.29800  5.25800
```

Figure 3.1.4: Pruning result

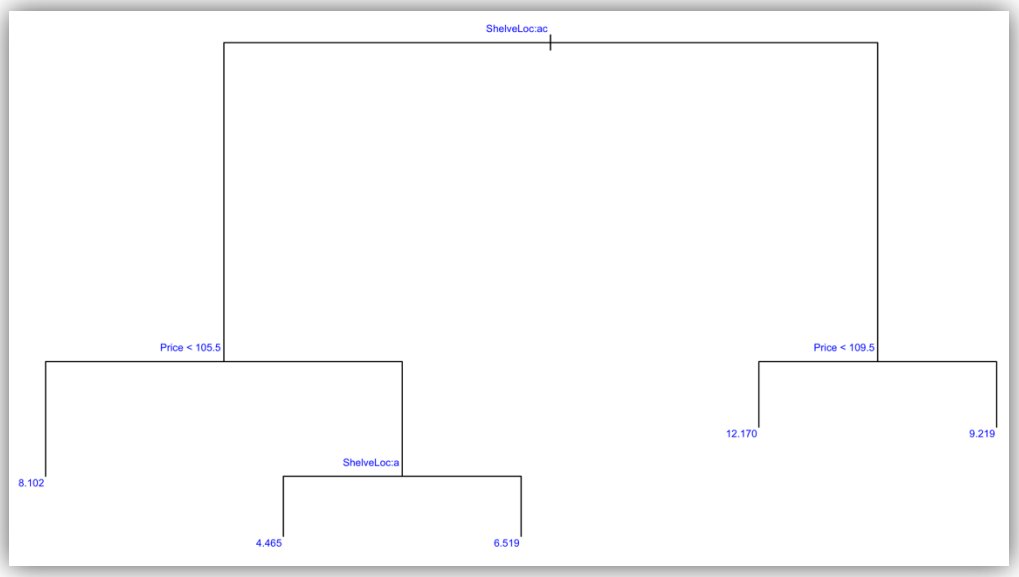

Figure 3.1.5: Construction of tree with 5 terminal nodes

```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 300 2371.00  7.526
   2) ShelveLoc: Bad,Medium 230 1290.00  6.704
     4) Price < 105.5 80  386.90  8.102 *
     5) Price > 105.5 150  663.60  5.958
      10) ShelveLoc: Bad 41  122.50  4.465 *
      11) ShelveLoc: Medium 109  415.40  6.519 *
   3) ShelveLoc: Good 70  412.80 10.230
     6) Price < 109.5 24   80.86 12.170 *
     7) Price > 109.5 46  194.80  9.219 *
```

Figure 3.1.6: Detailed construction of tree with 5 terminal nodes

```
> mean((ppred - cars.test)^2)
[1] 5.473715
> mean((ppred - cars.train)^2)
[1] 10.54429
```

Figure 3.1.7: Mean squared errors for training and test data after pruning

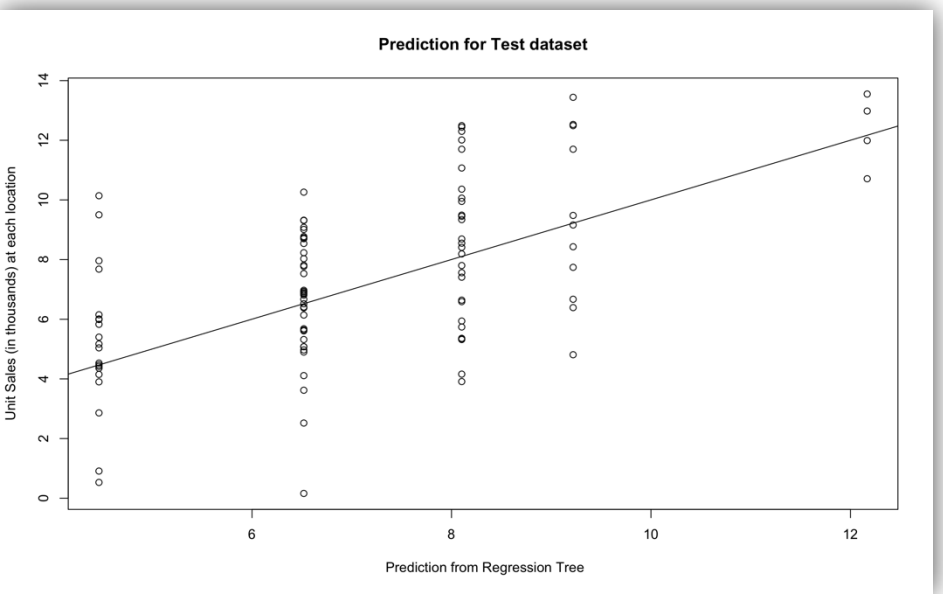

Figure 3.1.8: Prediction for test dataset after pruning

After pruning to 5 terminal nodes, there are only 2 predictor variables used to predict sales of child car seats. They are ShelveLoc and Price. The pruning simplifies the original tree a lot and increase the mean squared error for test data while decreasing the mean squared error for training data. In overall, one understood that there should be some trade-offs for pruning the original tree to only 5 terminal nodes. Simplifying the original regression tree came at a cost in increasing the residual mean deviance and mean squared error for test data. However the pruned tree provides an overall lower deviance and mean squared error for training data. Since one concerned about the overall impact of pruning more than the increase or decrease in test or training data, one concluded that pruning could be a great choice to simplify the original regression tree while obtaining a pretty similar result as the initial tree.

4. Appendix

```
getwd()

setwd("/Users/angeladianas/Desktop/Lund/r_files")

library(ISLR)

library(tree)

help(Carseats)

attach(Carseats)

dim(Carseats)


## Part 1

High <- ifelse(Sales <= 8, "No", "Yes")
```

```
Carseats <- data.frame(Carseats, High)
summary(Carseats)

random.seed(0820)
n = 300
train <- sample(1:nrow(Carseats), n)
test <- Carseats[-train,]
Hightest <- High[-train]
summary(test)

cstree <- tree(High~. - Sales, Carseats, subset = train)
summary(cstree)
cstree
plot(cstree)
text(cstree, col = "blue", adj = c(1.05,0), cex = 0.5)

cstree.pred <- predict(cstree, test, type = "class")
mean(cstree.pred == Hightest)

conftable <- table(cstree.pred, Hightest)
conftable
sens <- conftable[2,2]/(conftable[2,2] + conftable[1,2])
spec <- conftable[1,1]/(conftable[1,1] + conftable[2,1])
sens
spec
set.seed(0820)
cv.carseats <- cv.tree(cstree, FUN = prune.misclass, K = 10)
cv.carseats
```

```
plot(cv.carseats$size, cv.carseats$dev, type = "b", xlab = "Number of terminal nodes", ylab =
"Deviance", main = "Cross-validation for Carseats data")

dev.new()

plot(cv.carseats$k, cv.carseats$dev, type = "b", xlab = "Parameter Alpha", ylab = "Deviance", main =
"Cross-validation for Carseats data")


## best = 17

prune.carseats <- prune.misclass(cstree, best = 17)

summary(prune.carseats)

prune.carseats

plot(prune.carseats)

text(prune.carseats, col = "blue", cex = 0.5)


prunecstree.pred <- predict(prune.carseats, test, type = "class")

mean(prunecstree.pred == Hightest)

prunecstree.pred

prune.table <- table(prunecstree.pred, Hightest)

prune.table


prune.sens <- prune.table[2,2] / (prune.table[2,2] + prune.table[1,2])

prune.spec <- prune.table[1,1] / (prune.table[1,1] + prune.table[2,1])

prune.sens

prune.spec


## best = 13

prune.carseats <- prune.misclass(cstree, best = 13)

summary(prune.carseats)

prune.carseats

plot(prune.carseats)

text(prune.carseats, col = "blue", cex = 0.5)
```

```
prunecstree.pred <- predict(prune.carseats, test, type = "class")

mean(prunecstree.pred == Hightest)

prunecstree.pred

prune.table <- table(prunecstree.pred, Hightest)

prune.table


prune.sens <- prune.table[2,2] / (prune.table[2,2] + prune.table[1,2])

prune.spec <- prune.table[1,1] / (prune.table[1,1] + prune.table[2,1])

prune.sens

prune.spec


## Part 2

tree.cars <- tree(Sales~. - High, Carseats, subset = train)

tree.cars

summary(tree.cars)

plot(tree.cars)

text(tree.cars, col = "blue", cex = 0.6)


cv.cars = cv.tree(tree.cars, K = 10)

cv.cars

plot(cv.cars$size, cv.cars$dev, type = "b", xlab = "Number of terminal nodes", ylab = "Deviance", main
= "Cross-validation for Carseats data")

plot(cv.cars$k, cv.cars$dev, type = "b", xlab = "Parameter Alpha", ylab = "Deviance", main = "Cross-
validation for Carseats data")


prune.cars <- prune.tree(tree.cars, best = 5)

summary(prune.cars)

prune.cars

plot(prune.cars)
```

```
text(prune.cars, col = "blue", cex = 0.6, adj = c(1.05, 0))


pred <- predict(tree.cars, newdata = Carseats[-train,])

cars.test <- Carseats[-train, "Sales"]

plot(pred, cars.test, xlab = " Prediction from Regression Tree", ylab = "Unit Sales (in thousands) at
each location", main = "Prediction for Test dataset")

abline(0,1)


cars.train <- Carseats[train, "Sales"]


## MSE for data without pruning

mean((pred - cars.test)^2)

mean((pred - cars.train)^2)


ppred <- predict(prune.cars, newdata = Carseats[-train,])

plot(ppred, cars.test, xlab = " Prediction from Regression Tree", ylab = "Unit Sales (in thousands) at
each location", main = "Prediction for Test dataset")

abline(0,1)


## MSE for data with pruning

mean((ppred - cars.test)^2)

mean((ppred - cars.train)^2)
```