Week6\_Assignment

Mohit Mehndiratta

02/09/2021

### Attach the “Carseats” dataset from ISLR library and explore.

library(ISLR)  
library(tree)

## Warning: package 'tree' was built under R version 4.1.1

attach(Carseats)  
head(Carseats)

## Sales CompPrice Income Advertising Population Price ShelveLoc Age Education  
## 1 9.50 138 73 11 276 120 Bad 42 17  
## 2 11.22 111 48 16 260 83 Good 65 10  
## 3 10.06 113 35 10 269 80 Medium 59 12  
## 4 7.40 117 100 4 466 97 Medium 55 14  
## 5 4.15 141 64 3 340 128 Bad 38 13  
## 6 10.81 124 113 13 501 72 Bad 78 16  
## Urban US  
## 1 Yes Yes  
## 2 Yes Yes  
## 3 Yes Yes  
## 4 Yes Yes  
## 5 Yes No  
## 6 No Yes

dim(Carseats)

## [1] 400 11

str(Carseats)

## 'data.frame': 400 obs. of 11 variables:  
## $ Sales : num 9.5 11.22 10.06 7.4 4.15 ...  
## $ CompPrice : num 138 111 113 117 141 124 115 136 132 132 ...  
## $ Income : num 73 48 35 100 64 113 105 81 110 113 ...  
## $ Advertising: num 11 16 10 4 3 13 0 15 0 0 ...  
## $ Population : num 276 260 269 466 340 501 45 425 108 131 ...  
## $ Price : num 120 83 80 97 128 72 108 120 124 124 ...  
## $ ShelveLoc : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...  
## $ Age : num 42 65 59 55 38 78 71 67 76 76 ...  
## $ Education : num 17 10 12 14 13 16 15 10 10 17 ...  
## $ Urban : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...  
## $ US : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...

There are 400 observations with 11 variables. There are 3 factor variables “ShelveLoc”, “Urban”, “US” and rest all are numeric variables.

## Question 01: Regression Trees: Use “Carseats” dataset with Sales as the Target Variable

### a) Divide the dataset into Training and Testing sets.

set.seed(5)  
trainingIndexes <- sample(1:nrow(Carseats), nrow(Carseats)\* 0.5)  
  
training\_DS <- Carseats[trainingIndexes, ]  
dim(training\_DS)

## [1] 200 11

testing\_DS <- Carseats[-trainingIndexes,]  
dim(testing\_DS)

## [1] 200 11

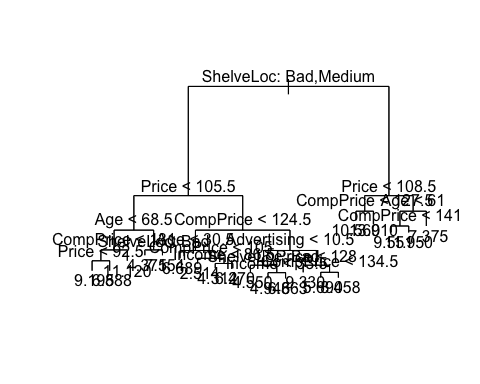
We divided training and testing data set into equal halves of the original data set. Now both of the data set contains 200 Observations of 11 variables each.

### b) Fit a Regression Tree for training data set.

reg\_tree <- tree(Sales~., data = Carseats, subset = trainingIndexes)  
summary(reg\_tree)

##   
## Regression tree:  
## tree(formula = Sales ~ ., data = Carseats, subset = trainingIndexes)  
## Variables actually used in tree construction:  
## [1] "ShelveLoc" "Price" "Age" "CompPrice" "Income"   
## [6] "Advertising"  
## Number of terminal nodes: 20   
## Residual mean deviance: 2.184 = 393.1 / 180   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -3.79500 -1.06000 0.08106 0.00000 0.91510 4.47500

plot(reg\_tree)  
text(reg\_tree, pretty = 0)



In this regression tree, we can see that we used 6 variables to form this tree with 20 terminal nodes.

For the accuracy of this model, we need to check mean squared errors and Root mean squared errors.

yhat1 <- predict(reg\_tree, newdata = testing\_DS)  
  
MSE <- mean((yhat1 - testing\_DS[, "Sales"])^2)  
MSE

## [1] 5.041825

## now the root mean squared error is :   
  
RMSE1 <- sqrt(MSE)  
RMSE1

## [1] 2.245401

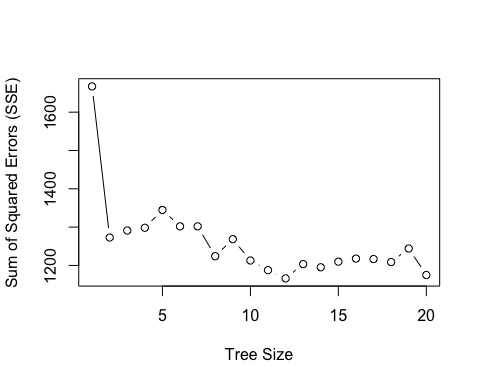
From this we can see that mean squared error and root mean squared error are **5.041825** and **2.245401** respectively.

### c) Construct the Cross validation plot

set.seed(6)  
cv\_reg\_tree <- cv.tree(reg\_tree)  
cv\_reg\_tree

## $size  
## [1] 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  
##   
## $dev  
## [1] 1175.004 1244.205 1208.848 1216.725 1217.858 1209.815 1195.177 1203.473  
## [9] 1166.139 1187.676 1212.913 1268.573 1224.098 1302.015 1302.015 1344.822  
## [17] 1298.239 1291.152 1272.818 1667.239  
##   
## $k  
## [1] -Inf 18.96802 20.37889 20.89588 21.06229 22.66376 25.56731  
## [8] 28.08164 29.41080 30.91787 35.93367 39.20154 42.90497 58.22055  
## [15] 58.43934 61.39928 78.15803 80.37642 136.17907 431.62093  
##   
## $method  
## [1] "deviance"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

plot(cv\_reg\_tree$size, cv\_reg\_tree$dev, type = "b", xlab = "Tree Size",   
 ylab = "Sum of Squared Errors (SSE)")



From the plot, we can observe that how sum of squared errors are affecting according to the tree size. Clearly, we can say that the higher the tree size the lower the SSE for this data set.

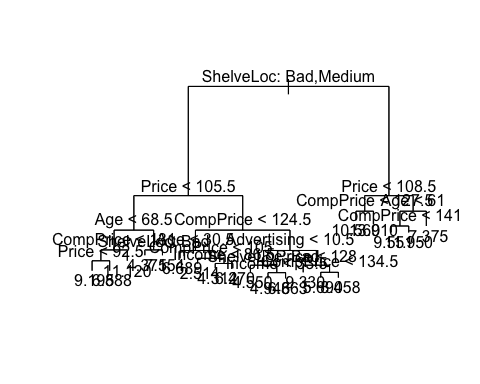
### d) Select the best size of the tree

From the graph of cross validation plot, we can see that there is a continuous drop till the largest size. So we have to select fully grown tree as our best size of the tree.

**Best size = 20**

### e) Obtain the best regression tree by pruning

pruned\_regtree <- prune.tree(reg\_tree, best = 20)  
plot(pruned\_regtree)  
text(pruned\_regtree, pretty = 0)



Here we can see that it is similar to original because we used best size as fully grown tree itself.

### f) Test the model accuracy.

yhat\_pruned <- predict(pruned\_regtree, newdata = testing\_DS)  
  
MSE\_pruned <- mean((yhat\_pruned - testing\_DS[, "Sales"])^2)  
MSE\_pruned

## [1] 5.041825

## now the root mean squared error is :   
RMSE\_pruned <- sqrt(MSE\_pruned)  
RMSE\_pruned

## [1] 2.245401

From this we can see that mean squared error and root mean squared error are **5.041825** and **2.245401** respectively, which is exactly same as our original regression tree.

### g) Describe the terminal nodes of the resulting decision tree

pruned\_regtree

## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 200 1633.000 7.382   
## 2) ShelveLoc: Bad,Medium 152 913.500 6.556   
## 4) Price < 105.5 47 278.300 7.971   
## 8) Age < 68.5 36 158.400 8.684   
## 16) CompPrice < 131 30 107.600 8.196   
## 32) Price < 92.5 17 48.270 9.195 \*  
## 33) Price > 92.5 13 20.160 6.888 \*  
## 17) CompPrice > 131 6 7.888 11.120 \*  
## 9) Age > 68.5 11 41.740 5.638   
## 18) ShelveLoc: Bad 6 16.180 4.375 \*  
## 19) ShelveLoc: Medium 5 4.500 7.154 \*  
## 5) Price > 105.5 105 499.000 5.923   
## 10) CompPrice < 124.5 39 153.900 4.785   
## 20) Age < 30.5 7 11.410 6.689 \*  
## 21) Age > 30.5 32 111.600 4.368   
## 42) CompPrice < 105 5 22.180 2.514 \*  
## 43) CompPrice > 105 27 69.060 4.712   
## 86) Income < 86.5 22 32.580 4.312 \*  
## 87) Income > 86.5 5 17.510 6.470 \*  
## 11) CompPrice > 124.5 66 264.600 6.596   
## 22) Advertising < 10.5 46 141.200 5.976   
## 44) ShelveLoc: Bad 11 25.950 4.550 \*  
## 45) ShelveLoc: Medium 35 85.840 6.425   
## 90) Income < 33.5 8 10.190 4.946 \*  
## 91) Income > 33.5 27 52.990 6.863 \*  
## 23) Advertising > 10.5 20 65.220 8.020   
## 46) Price < 128 9 3.001 9.330 \*  
## 47) Price > 128 11 34.130 6.948   
## 94) CompPrice < 134.5 6 4.375 5.690 \*  
## 95) CompPrice > 134.5 5 8.863 8.458 \*  
## 3) ShelveLoc: Good 48 288.400 9.996   
## 6) Price < 108.5 14 53.120 11.760   
## 12) CompPrice < 127.5 9 6.683 10.560 \*  
## 13) CompPrice > 127.5 5 10.500 13.910 \*  
## 7) Price > 108.5 34 173.800 9.270   
## 14) Age < 61 23 74.570 10.180   
## 28) CompPrice < 141 17 44.110 9.551 \*  
## 29) CompPrice > 141 6 4.892 11.950 \*  
## 15) Age > 61 11 40.830 7.375 \*

From this we can see that we have 20 terminal nodes(the nodes which are marked with \* sign) and how we obtained those 20 terminal nodes and predicted the value of the Sales.

So our tree representation is like,

Terminal node 1:

If ShelveLoc is either Bad or medium and if price is less than 105.5 and Age is less than 68.5 and CompPrice is less than 131 and Price is less than 92.5 then average sales predicted is 9.195.

Terminal node 2:

If ShelveLoc is either Bad or medium and if price is less than 105.5 and Age is less than 68.5 and CompPrice is less than 131 and Price is greater than 92.5 then average sales predicted is 6.888.

Terminal node 3:

If ShelveLoc is either Bad or medium and if price is less than 105.5 and Age is less than 68.5 and CompPrice is greater than 131 then average sales predicted is 11.120.

Terminal node 4:

If ShelveLoc is either Bad or medium and if price is less than 105.5 and Age is greater than 68.5 and if ShelveLoc is bad then average sales predicted is 4.375.

Terminal node 5:

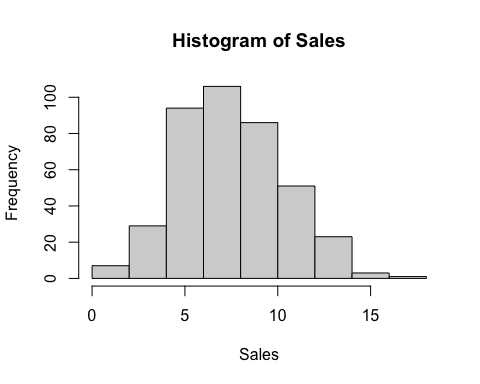
If ShelveLoc is either Bad or medium and if price is less than 105.5 and Age is greater than 68.5 and if ShelveLoc is medium then average sales predicted is 7.154.

and so on upto Terminal 20.

## Question 02: Classification Trees: Use “Carseats” dataset with Sales as the Target Variable

### a) Transfer Sales Variable from a Continuous variable to a Categorical Variable. Assign Assign Sales as “Yes” if Sales value is greater than 8 and “No” otherwise.

hist(Sales)



HighSales <- ifelse(Sales <= 8, "NO", "YES")  
head(HighSales)

## [1] "YES" "YES" "YES" "NO" "NO" "YES"

table(HighSales)

## HighSales  
## NO YES   
## 236 164

# attaching the HighSales variable to the original data set.  
CarSeatsNew <- data.frame(Carseats, HighSales)  
# removing the Sales variable as we don't need it in classification now.  
CarSeatsNew <- CarSeatsNew[, -1]  
head(CarSeatsNew)

## CompPrice Income Advertising Population Price ShelveLoc Age Education Urban  
## 1 138 73 11 276 120 Bad 42 17 Yes  
## 2 111 48 16 260 83 Good 65 10 Yes  
## 3 113 35 10 269 80 Medium 59 12 Yes  
## 4 117 100 4 466 97 Medium 55 14 Yes  
## 5 141 64 3 340 128 Bad 38 13 Yes  
## 6 124 113 13 501 72 Bad 78 16 No  
## US HighSales  
## 1 Yes YES  
## 2 Yes YES  
## 3 Yes YES  
## 4 Yes NO  
## 5 No NO  
## 6 Yes YES

str(CarSeatsNew)

## 'data.frame': 400 obs. of 11 variables:  
## $ CompPrice : num 138 111 113 117 141 124 115 136 132 132 ...  
## $ Income : num 73 48 35 100 64 113 105 81 110 113 ...  
## $ Advertising: num 11 16 10 4 3 13 0 15 0 0 ...  
## $ Population : num 276 260 269 466 340 501 45 425 108 131 ...  
## $ Price : num 120 83 80 97 128 72 108 120 124 124 ...  
## $ ShelveLoc : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...  
## $ Age : num 42 65 59 55 38 78 71 67 76 76 ...  
## $ Education : num 17 10 12 14 13 16 15 10 10 17 ...  
## $ Urban : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...  
## $ US : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...  
## $ HighSales : chr "YES" "YES" "YES" "NO" ...

# Converting HighSales to form a factor variable for classification.  
CarSeatsNew$HighSales = as.factor(CarSeatsNew$HighSales)  
str(CarSeatsNew)

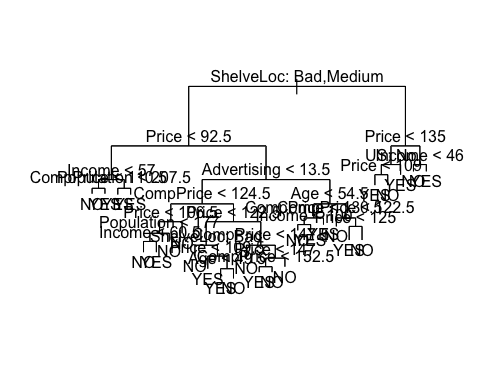
## 'data.frame': 400 obs. of 11 variables:  
## $ CompPrice : num 138 111 113 117 141 124 115 136 132 132 ...  
## $ Income : num 73 48 35 100 64 113 105 81 110 113 ...  
## $ Advertising: num 11 16 10 4 3 13 0 15 0 0 ...  
## $ Population : num 276 260 269 466 340 501 45 425 108 131 ...  
## $ Price : num 120 83 80 97 128 72 108 120 124 124 ...  
## $ ShelveLoc : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...  
## $ Age : num 42 65 59 55 38 78 71 67 76 76 ...  
## $ Education : num 17 10 12 14 13 16 15 10 10 17 ...  
## $ Urban : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...  
## $ US : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...  
## $ HighSales : Factor w/ 2 levels "NO","YES": 2 2 2 1 1 2 1 2 1 1 ...

### b) Fit a Classification Tree for the full data set.

class\_tree\_full <- tree(HighSales~., data = CarSeatsNew)  
summary(class\_tree\_full)

##   
## Classification tree:  
## tree(formula = HighSales ~ ., data = CarSeatsNew)  
## Variables actually used in tree construction:  
## [1] "ShelveLoc" "Price" "Income" "CompPrice" "Population"   
## [6] "Advertising" "Age" "US"   
## Number of terminal nodes: 27   
## Residual mean deviance: 0.4575 = 170.7 / 373   
## Misclassification error rate: 0.09 = 36 / 400

plot(class\_tree\_full)  
text(class\_tree\_full,pretty = 0)



Here, we can see that there are 27 terminal nodes and misclassification rate for fully grown tree is 9% without split.

### c) Test the model accuracy.

misclassification rate that is 9%. To be more confident we divide the data into training and testing data set. If misclassification rate is as low as this in testing data set then we can say this is good model

### d) Divide the dataset into Training and Testing sets and fit a Classification Tree for training dataset.

set.seed(5)  
tr <- sample(1:nrow(CarSeatsNew), nrow(CarSeatsNew) \* 0.5)  
  
CarSeats\_Training <- CarSeatsNew[tr, ]  
dim(CarSeats\_Training)

## [1] 200 11

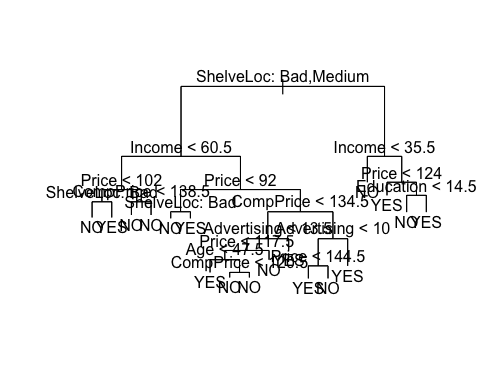
CarSeats\_Testing <- CarSeatsNew[-tr, ]  
dim(CarSeats\_Testing)

## [1] 200 11

# Classigication tree for training DS  
class\_tree\_training <- tree(HighSales~., data = CarSeats\_Training)  
summary(class\_tree\_training)

##   
## Classification tree:  
## tree(formula = HighSales ~ ., data = CarSeats\_Training)  
## Variables actually used in tree construction:  
## [1] "ShelveLoc" "Income" "Price" "CompPrice" "Advertising"  
## [6] "Age" "Education"   
## Number of terminal nodes: 18   
## Residual mean deviance: 0.4899 = 89.16 / 182   
## Misclassification error rate: 0.125 = 25 / 200

plot(class\_tree\_training)  
text(class\_tree\_training, pretty = 0)



Here, we can see that with the 50% of the data, terminal nodes came down to 18 and misclassification rate is 12.5%.

### e) Test the model accuracy.

pred\_class\_test1 <- predict(class\_tree\_training, newdata = CarSeats\_Testing, type = "class")  
head(pred\_class\_test1)

## [1] NO NO NO YES YES NO   
## Levels: NO YES

observed\_test <- CarSeats\_Testing[, "HighSales"]  
  
misclassification\_matrix <- table(pred\_class\_test1, observed\_test)  
misclassification\_matrix

## observed\_test  
## pred\_class\_test1 NO YES  
## NO 83 36  
## YES 32 49

misclassrate <- ( misclassification\_matrix[1,2] + misclassification\_matrix[2,1] )/ sum(misclassification\_matrix)  
  
misclassrate

## [1] 0.34

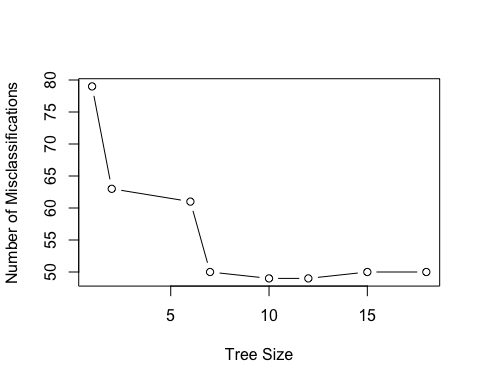
Here, we can see that misclassification rate increased to 34%, for our training data set tree with 18 terminal nodes.

### f) Construct the Cross validation plot.

set.seed(1)  
cv\_classtree <- cv.tree(class\_tree\_training, FUN = prune.misclass) ## we give this function for classification problem,  
# else we set K value  
cv\_classtree

## $size  
## [1] 18 15 12 10 7 6 2 1  
##   
## $dev  
## [1] 50 50 49 49 50 61 63 79  
##   
## $k  
## [1] -Inf 0.0000000 0.3333333 0.5000000 1.0000000 4.0000000 4.7500000  
## [8] 26.0000000  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

plot(cv\_classtree$size, cv\_classtree$dev, type = "b", xlab = "Tree Size",   
 ylab = "Number of Misclassifications")



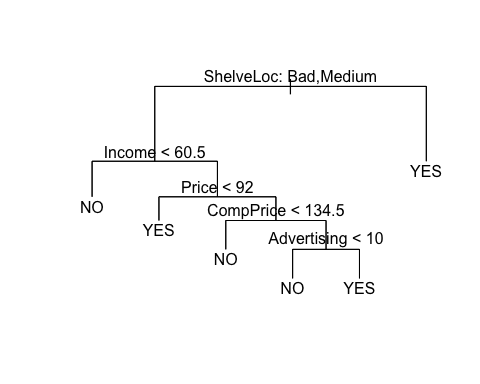
From here, we can see that the deviance(number of misclassification) are lower for tree size 6.

### g) Select the best size of the tree model.

From 66 we can see 53 is less. so when tree size is 6 prune tree produces better results in cross validation so we will go with pruned tree => best = 6

### h) Obtain the best classification tree by pruning.

pruned\_classtree <- prune.misclass(class\_tree\_training, best = 6)  
plot(pruned\_classtree)  
text(pruned\_classtree, pretty = 0)



### i) Test the model accuracy.

pred\_class\_test2 <- predict(pruned\_classtree, newdata = CarSeats\_Testing, type = "class")  
head(pred\_class\_test2)

## [1] YES NO NO NO YES YES  
## Levels: NO YES

observed\_test <- CarSeats\_Testing[, "HighSales"]  
  
misclassification\_matrix\_pruned <- table(pred\_class\_test2, observed\_test)  
misclassification\_matrix\_pruned

## observed\_test  
## pred\_class\_test2 NO YES  
## NO 98 42  
## YES 17 43

misclassrate <- ( misclassification\_matrix\_pruned[1,2] + misclassification\_matrix\_pruned[2,1] )/ sum(misclassification\_matrix\_pruned)  
  
misclassrate

## [1] 0.295

Here, we can observe that with pruning misclassification rate goes down to 29.5% with 6 terminal nodes.

### j) Describe the terminal nodes of the resulting decision tree.

pruned\_classtree

## node), split, n, deviance, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 200 268.40 NO ( 0.60500 0.39500 )   
## 2) ShelveLoc: Bad,Medium 152 179.20 NO ( 0.72368 0.27632 )   
## 4) Income < 60.5 61 39.22 NO ( 0.90164 0.09836 ) \*  
## 5) Income > 60.5 91 122.20 NO ( 0.60440 0.39560 )   
## 10) Price < 92 17 18.55 YES ( 0.23529 0.76471 ) \*  
## 11) Price > 92 74 91.72 NO ( 0.68919 0.31081 )   
## 22) CompPrice < 134.5 49 43.61 NO ( 0.83673 0.16327 ) \*  
## 23) CompPrice > 134.5 25 33.65 YES ( 0.40000 0.60000 )   
## 46) Advertising < 10 15 19.10 NO ( 0.66667 0.33333 ) \*  
## 47) Advertising > 10 10 0.00 YES ( 0.00000 1.00000 ) \*  
## 3) ShelveLoc: Good 48 51.67 YES ( 0.22917 0.77083 ) \*

From this we can see that we have 6 terminal nodes(the nodes which are marked with \* sign) and how we obtained those 6 terminal nodes and predicted the value of the Sales variable.

So our tree representation is like,

Terminal node 1:

If ShelveLoc is Bad or Medium and if income is less than 60.5 then there are no high sales.

Terminal node 2:

If ShelveLoc is Bad or Medium and if income is greater than 60.5 and price is less than 92 then there are high sales.

Terminal node 3:

If ShelveLoc is Bad or Medium and if income is greater than 60.5 and price is greater than 92 and CompPrice is less than 134.5 then there are no high sales.

Terminal node 4:

If ShelveLoc is Bad or Medium and if income is greater than 60.5 and price is greater than 92 and CompPrice is greater than 134.5 and Advertisings are less than 10 then there are no high sales.

Terminal node 5:

If ShelveLoc is Bad or Medium and if income is greater than 60.5 and price is greater than 92 and CompPrice is greater than 134.5 and Advertisings are greater than 10 then there are high sales.

Terminal node 6:

If ShelveLoc is Good, then there are high sales.