Decision-Trees.R

patriciamaya

2020-09-16

```r
library(MASS)
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tibble)
library(ggplot2)
library(knitr)
library(gdata)
```

```
## Warning: package 'gdata' was built under R version 4.0.2

## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.

##

## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.

##
## Attaching package: 'gdata'

## The following objects are masked from 'package:dplyr':
##
##     combine, first, last

## The following object is masked from 'package:stats':
##
##     nobs
```

```
## The following object is masked from 'package:utils':
##
##     object.size
```

```
## The following object is masked from 'package:base':
##
##     startsWith
```

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.0.2
```

```
data("Carseats")
attach(Carseats)
#create new categorical variables "High"
High <- ifelse(Sales >= 8, "YES", "NO" )
High <- as.factor(High)

#Attach new variable to df & remove 1st column (Sales) of df
Carseats <- data.frame(Carseats, High)
Carseats <- Carseats[-1]

#Divide data into train and test
set.seed(3)
indx <- sample(2, nrow(Carseats), replace=T, prob= c(0.7, 0.3))
train <- Carseats[indx == 1, ]
test <- Carseats[indx ==2, ]

#most common package for decision trees
#this function uses gini/information gain for classfication prob
#install.packages("rpart")
library(rpart)
```

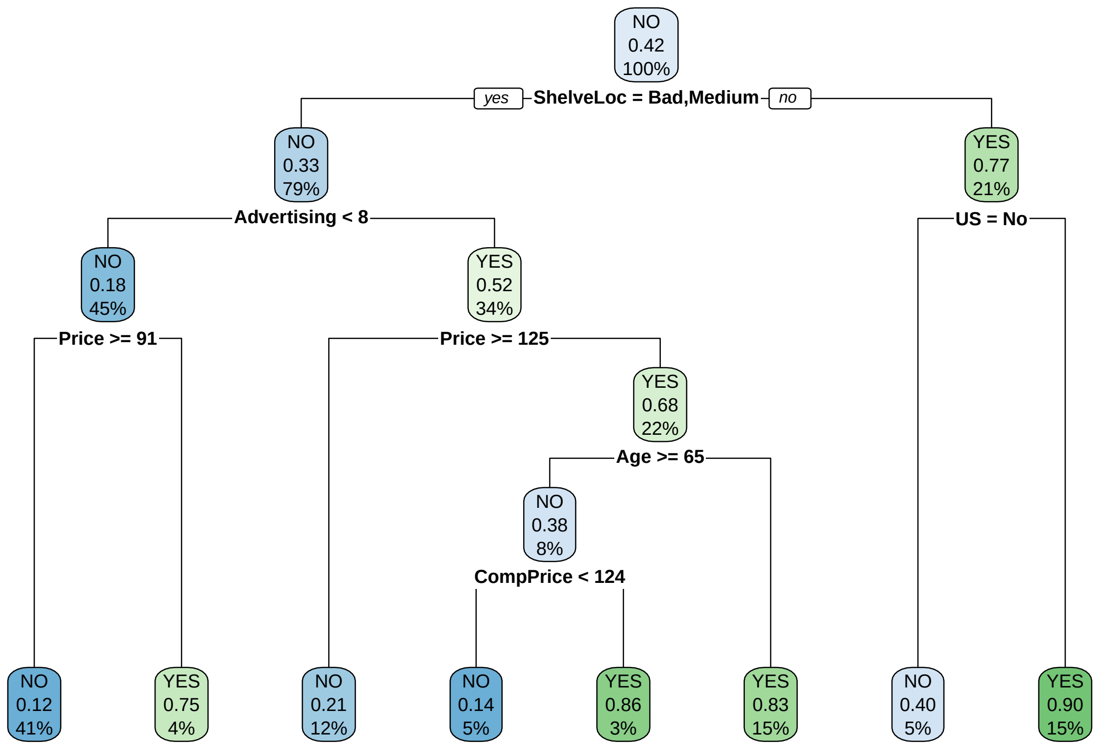
```
## Warning: package 'rpart' was built under R version 4.0.2
```

```
#TRAIN
#simplest model- using all (.) other variables as input variables
tree_model <- rpart(High ~ . , data=train)

#To visualize tree model
#install.packages("rpart.plot")
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.0.2
```

```
rpart.plot(tree_model)
```




```
print(tree_model)
```

```
## n= 277
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 277 116 NO (0.5812274 0.4187726)
##    2) ShelveLoc=Bad,Medium 220  72 NO (0.6727273 0.3272727)
##      4) Advertising< 7.5 125  23 NO (0.8160000 0.1840000)
##        8) Price>=90.5 113  14 NO (0.8761062 0.1238938) *
##        9) Price< 90.5 12   3 YES (0.2500000 0.7500000) *
##      5) Advertising>=7.5 95  46 YES (0.4842105 0.5157895)
##       10) Price>=124.5 33   7 NO (0.7878788 0.2121212) *
##       11) Price< 124.5 62  20 YES (0.3225806 0.6774194)
##         22) Age>=65 21   8 NO (0.6190476 0.3809524)
##           44) CompPrice< 123.5 14   2 NO (0.8571429 0.1428571) *
##           45) CompPrice>=123.5 7   1 YES (0.1428571 0.8571429) *
##         23) Age< 65 41   7 YES (0.1707317 0.8292683) *
##    3) ShelveLoc=Good 57  13 YES (0.2280702 0.7719298)
##      6) US=No 15   6 NO (0.6000000 0.4000000) *
##      7) US=Yes 42   4 YES (0.0952381 0.9047619) *
```

```
#READS AS: node), split, n, loss, yval, (yprob)
#split - variabel name & condition , n - number of instances on the node, loss - instances predicted in
#yval - predicted class, (yprob) - probablitity in being in each class->
#(yprob) (x,y) first num x corresponds to the class predicted in first node
# * are terminal nodes

#PREDICTION ON TEST DATA
tree_pred_probability <- predict(tree_model, test)
```

```r
print(tree_pred_probability) #shows prob(confidence) of being in class 1 or 2
```

```
##            NO       YES
## 2   0.0952381 0.9047619
## 15  0.0952381 0.9047619
## 16  0.8761062 0.1238938
## 18  0.0952381 0.9047619
## 19  0.0952381 0.9047619
## 26  0.6000000 0.4000000
## 28  0.8761062 0.1238938
## 30  0.1707317 0.8292683
## 37  0.6000000 0.4000000
## 42  0.8761062 0.1238938
## 50  0.6000000 0.4000000
## 53  0.8761062 0.1238938
## 54  0.1707317 0.8292683
## 55  0.7878788 0.2121212
## 56  0.8761062 0.1238938
## 61  0.1707317 0.8292683
## 63  0.8761062 0.1238938
## 65  0.1707317 0.8292683
## 67  0.8761062 0.1238938
## 71  0.0952381 0.9047619
## 72  0.7878788 0.2121212
## 73  0.8761062 0.1238938
## 74  0.0952381 0.9047619
## 79  0.8761062 0.1238938
## 80  0.2500000 0.7500000
## 81  0.8571429 0.1428571
## 85  0.8761062 0.1238938
## 86  0.8761062 0.1238938
## 90  0.8761062 0.1238938
## 94  0.8761062 0.1238938
## 95  0.2500000 0.7500000
## 101 0.8571429 0.1428571
## 104 0.8761062 0.1238938
## 115 0.8571429 0.1428571
## 116 0.8761062 0.1238938
## 117 0.8761062 0.1238938
## 121 0.7878788 0.2121212
## 122 0.1707317 0.8292683
## 123 0.8761062 0.1238938
## 129 0.8761062 0.1238938
## 135 0.8761062 0.1238938
## 144 0.8761062 0.1238938
## 145 0.6000000 0.4000000
## 148 0.0952381 0.9047619
## 151 0.0952381 0.9047619
## 153 0.6000000 0.4000000
## 161 0.8761062 0.1238938
## 164 0.8761062 0.1238938
## 165 0.8761062 0.1238938
## 166 0.8761062 0.1238938
## 167 0.7878788 0.2121212
```

```
## 169 0.8761062 0.1238938
## 174 0.8761062 0.1238938
## 178 0.8761062 0.1238938
## 187 0.2500000 0.7500000
## 192 0.0952381 0.9047619
## 195 0.7878788 0.2121212
## 197 0.8761062 0.1238938
## 198 0.8761062 0.1238938
## 201 0.8761062 0.1238938
## 202 0.8761062 0.1238938
## 205 0.8761062 0.1238938
## 207 0.8761062 0.1238938
## 209 0.2500000 0.7500000
## 211 0.8761062 0.1238938
## 216 0.7878788 0.2121212
## 220 0.0952381 0.9047619
## 222 0.8761062 0.1238938
## 226 0.2500000 0.7500000
## 227 0.6000000 0.4000000
## 236 0.7878788 0.2121212
## 242 0.8761062 0.1238938
## 248 0.8761062 0.1238938
## 249 0.8761062 0.1238938
## 250 0.8761062 0.1238938
## 259 0.2500000 0.7500000
## 260 0.8571429 0.1428571
## 261 0.1707317 0.8292683
## 262 0.8761062 0.1238938
## 265 0.0952381 0.9047619
## 272 0.8761062 0.1238938
## 273 0.6000000 0.4000000
## 274 0.1707317 0.8292683
## 288 0.2500000 0.7500000
## 289 0.8761062 0.1238938
## 297 0.0952381 0.9047619
## 298 0.8571429 0.1428571
## 303 0.8571429 0.1428571
## 304 0.1707317 0.8292683
## 307 0.8761062 0.1238938
## 309 0.7878788 0.2121212
## 310 0.1428571 0.8571429
## 316 0.0952381 0.9047619
## 318 0.6000000 0.4000000
## 319 0.0952381 0.9047619
## 323 0.0952381 0.9047619
## 325 0.8761062 0.1238938
## 327 0.8761062 0.1238938
## 328 0.8571429 0.1428571
## 329 0.8761062 0.1238938
## 330 0.0952381 0.9047619
## 331 0.8761062 0.1238938
## 333 0.1707317 0.8292683
## 336 0.8571429 0.1428571
## 337 0.8761062 0.1238938
```

```
## 339 0.8761062 0.1238938
## 350 0.1707317 0.8292683
## 351 0.1707317 0.8292683
## 353 0.0952381 0.9047619
## 354 0.1707317 0.8292683
## 358 0.2500000 0.7500000
## 360 0.7878788 0.2121212
## 364 0.6000000 0.4000000
## 375 0.8761062 0.1238938
## 376 0.8761062 0.1238938
## 377 0.0952381 0.9047619
## 385 0.0952381 0.9047619
## 386 0.7878788 0.2121212
## 387 0.8761062 0.1238938
## 389 0.1428571 0.8571429
## 392 0.8761062 0.1238938
## 394 0.1707317 0.8292683
## 398 0.7878788 0.2121212
```

```
tree_pred_class <- predict(tree_model, test, type = "class")
print(tree_pred_class) #shows predicted class
```

```
##   2  15  16  18  19  26  28  30  37  42  50  53  54  55  56  61  63  65  67  71
## YES YES  NO YES YES  NO  NO YES  NO  NO  NO  NO YES  NO  NO YES  NO YES  NO YES
##  72  73  74  79  80  81  85  86  90  94  95 101 104 115 116 117 121 122 123 129
##  NO  NO YES  NO YES  NO  NO  NO  NO  NO YES  NO  NO  NO  NO  NO  NO YES  NO  NO
## 135 144 145 148 151 153 161 164 165 166 167 169 174 178 187 192 195 197 198 201
##  NO  NO  NO YES YES  NO  NO  NO  NO  NO  NO  NO  NO  NO YES YES  NO  NO  NO  NO
## 202 205 207 209 211 216 220 222 226 227 236 242 248 249 250 259 260 261 262 265
##  NO  NO  NO YES  NO  NO YES  NO YES  NO  NO  NO  NO  NO  NO YES  NO YES  NO YES
## 272 273 274 288 289 297 298 303 304 307 309 310 316 318 319 323 325 327 328 329
##  NO  NO YES YES  NO YES  NO  NO YES  NO  NO YES YES  NO YES YES  NO  NO  NO  NO
## 330 331 333 336 337 339 350 351 353 354 358 360 364 375 376 377 385 386 387 389
## YES  NO YES  NO  NO  NO YES YES YES YES YES  NO  NO  NO  NO YES YES  NO  NO YES
## 392 394 398
##  NO YES  NO
## Levels: NO YES
```

```
#ACCURACY OF TEST DATA
#compares actual values == predicted
mean(test$High == tree_pred_class)
```

```
## [1] 0.7398374
```

```
#ACCURACY ON TRAIN DATA (==)
tree_pred_class_train <- predict(tree_model, train, type = "class")
mean(train$High == tree_pred_class_train)
```

```
## [1] 0.8411552
```

```
#ERROR RATE ON TRAINING (!=)
mean(train$High != tree_pred_class_train)
```

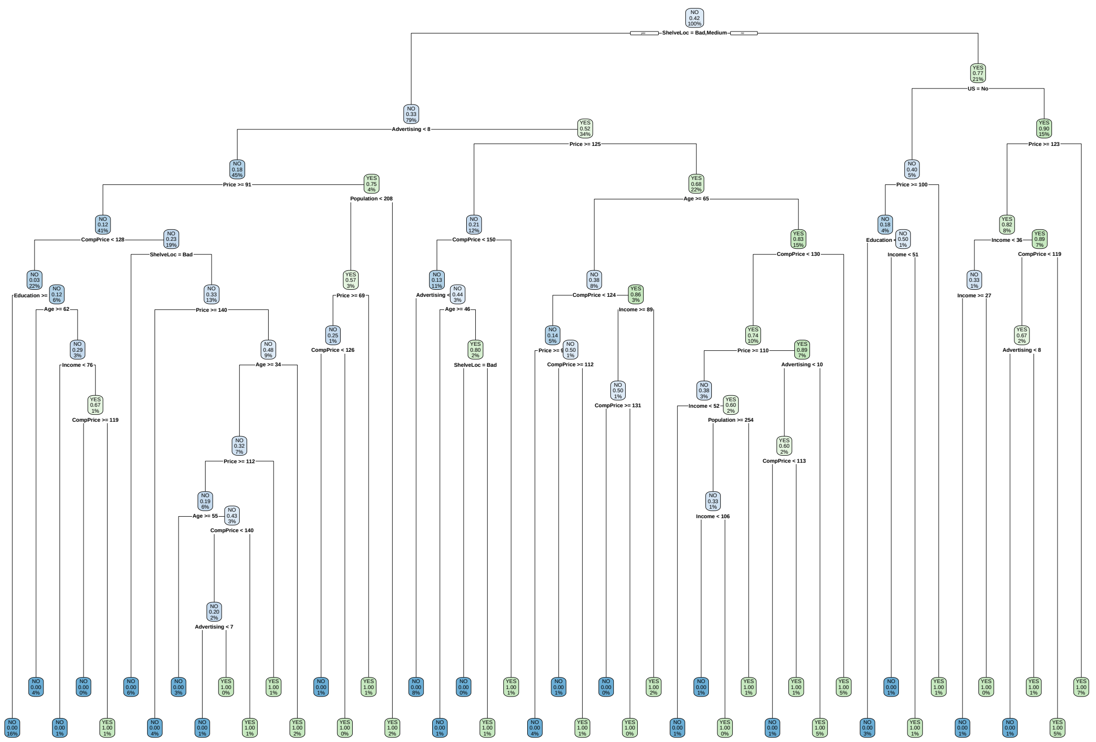
```
## [1] 0.1588448
```

```
#rpart(formula, data=train, parms= , control= )
#control-> controls how to split. control = rpart.control(minsplit=10)
```

```r
#minsplit=10 -> at least 10 instances must be in each node
#minbucket=10 -> min num of instances expected in terminal nodes
#cp -> complexity parameter -> want the one with min error & also size of tree
#when cp is large - size of tree is small and error is larger
#when cp is small - size of tree is large and error is smaller

#_______________________________________________________________________________________________________
#TRAIN NEW MODEL - *FULL TREE*
tree_model_full <- rpart(High ~ . , data=train, parms = list(split="information"), control = rpart.cont
rpart.plot(tree_model_full)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```r
#PREDICTION ON TEST DATA *FULL TREE*
tree_pred_probability_full <- predict(tree_model_full, test)
print(tree_pred_probability_full) #shows prob(confidence) of being in class 1 or 2
```

```
##     NO YES
## 2    0   1
## 15   0   1
## 16   1   0
## 18   0   1
## 19   0   1
## 26   0   1
## 28   1   0
## 30   0   1
## 37   0   1
## 42   1   0
## 50   0   1
## 53   1   0
## 54   0   1
```

```
## 55    1    0
## 56    1    0
## 61    1    0
## 63    1    0
## 65    0    1
## 67    1    0
## 71    0    1
## 72    1    0
## 73    1    0
## 74    0    1
## 79    1    0
## 80    0    1
## 81    1    0
## 85    1    0
## 86    1    0
## 90    1    0
## 94    0    1
## 95    1    0
## 101   1    0
## 104   1    0
## 115   1    0
## 116   1    0
## 117   1    0
## 121   1    0
## 122   0    1
## 123   1    0
## 129   1    0
## 135   1    0
## 144   1    0
## 145   1    0
## 148   0    1
## 151   0    1
## 153   1    0
## 161   1    0
## 164   1    0
## 165   1    0
## 166   1    0
## 167   1    0
## 169   1    0
## 174   1    0
## 178   0    1
## 187   1    0
## 192   0    1
## 195   0    1
## 197   1    0
## 198   1    0
## 201   1    0
## 202   1    0
## 205   0    1
## 207   1    0
## 209   0    1
## 211   1    0
## 216   1    0
## 220   0    1
```

```
## 222   1     0
## 226   0     1
## 227   1     0
## 236   1     0
## 242   0     1
## 248   1     0
## 249   1     0
## 250   1     0
## 259   0     1
## 260   1     0
## 261   0     1
## 262   1     0
## 265   1     0
## 272   1     0
## 273   0     1
## 274   0     1
## 288   0     1
## 289   1     0
## 297   0     1
## 298   1     0
## 303   1     0
## 304   0     1
## 307   1     0
## 309   1     0
## 310   1     0
## 316   0     1
## 318   1     0
## 319   0     1
## 323   0     1
## 325   1     0
## 327   1     0
## 328   1     0
## 329   1     0
## 330   0     1
## 331   1     0
## 333   0     1
## 336   1     0
## 337   1     0
## 339   1     0
## 350   0     1
## 351   0     1
## 353   0     1
## 354   0     1
## 358   0     1
## 360   1     0
## 364   1     0
## 375   0     1
## 376   1     0
## 377   0     1
## 385   0     1
## 386   1     0
## 387   1     0
## 389   1     0
## 392   1     0
```

```
## 394  1   0
## 398  0   1
```

```r
tree_pred_class_full <- predict(tree_model_full, test, type = "class")
print(tree_pred_class_full) #shows predicted class
```

```
##   2  15  16  18  19  26  28  30  37  42  50  53  54  55  56  61  63  65  67  71
## YES YES  NO YES YES YES  NO YES YES  NO YES  NO YES  NO  NO  NO  NO YES  NO YES
##  72  73  74  79  80  81  85  86  90  94  95 101 104 115 116 117 121 122 123 129
##  NO  NO YES  NO YES  NO  NO  NO  NO YES  NO  NO  NO  NO  NO  NO  NO YES  NO  NO
## 135 144 145 148 151 153 161 164 165 166 167 169 174 178 187 192 195 197 198 201
##  NO  NO  NO YES YES  NO  NO  NO  NO  NO  NO  NO  NO YES  NO YES YES  NO  NO  NO
## 202 205 207 209 211 216 220 222 226 227 236 242 248 249 250 259 260 261 262 265
##  NO YES  NO YES  NO  NO YES  NO YES  NO  NO YES  NO  NO  NO YES  NO YES  NO  NO
## 272 273 274 288 289 297 298 303 304 307 309 310 316 318 319 323 325 327 328 329
##  NO YES YES YES  NO YES  NO  NO YES  NO  NO  NO YES  NO YES YES  NO  NO  NO  NO
## 330 331 333 336 337 339 350 351 353 354 358 360 364 375 376 377 385 386 387 389
## YES  NO YES  NO  NO  NO YES YES YES YES YES  NO  NO YES  NO YES YES  NO  NO  NO
## 392 394 398
##  NO  NO YES
## Levels: NO YES
```

```r
#ACCURACY ON TRAIN DATA (==) *FULL TREE*
tree_pred_class_train_full <- predict(tree_model_full, train, type = "class")
mean(train$High == tree_pred_class_train_full)
```

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

```r
#ERROR RATE ON TRAINING (!=) *FULL TREE*
mean(train$High != tree_pred_class_train_full)
```

```
## [1] 0
```

```r
#ACCURACY OF TEST DATA *FULL TREE*
#compares actual values == predicted
mean(test$High == tree_pred_class_full)
```

```
## [1] 0.7723577
```

```r
#____________________________________________________________________
summary(tree_model)
```

```
## Call:
## rpart(formula = High ~ ., data = train)
##   n= 277
##
##           CP nsplit rel error    xerror       xstd
## 1 0.26724138      0 1.0000000 1.0000000 0.07078546
## 2 0.09482759      1 0.7327586 0.7327586 0.06617016
## 3 0.05172414      3 0.5431034 0.6293103 0.06320896
## 4 0.04310345      4 0.4913793 0.6551724 0.06401871
## 5 0.02586207      6 0.4051724 0.6465517 0.06375415
## 6 0.01000000      7 0.3793103 0.5775862 0.06143971
##
## Variable importance
##       Price Advertising   ShelveLoc          US    CompPrice         Age
##          22          20          20          13           11           7
##   Education  Population      Income
```

```
##           3           2           1
##
## Node number 1: 277 observations,    complexity param=0.2672414
##   predicted class=NO   expected loss=0.4187726  P(node) =1
##     class counts:   161   116
##    probabilities: 0.581 0.419
##   left son=2 (220 obs) right son=3 (57 obs)
##   Primary splits:
##       ShelveLoc   splits as  LRL,       improve=17.901860, (0 missing)
##       Advertising < 6.5   to the left,  improve=14.389040, (0 missing)
##       Price       < 90.5  to the right, improve=11.379220, (0 missing)
##       US          splits as  LR,        improve= 6.430861, (0 missing)
##       Age         < 61.5  to the right, improve= 6.174862, (0 missing)
##
## Node number 2: 220 observations,    complexity param=0.09482759
##   predicted class=NO   expected loss=0.3272727  P(node) =0.7942238
##     class counts:   148    72
##    probabilities: 0.673 0.327
##   left son=4 (125 obs) right son=5 (95 obs)
##   Primary splits:
##       Advertising < 7.5   to the left,  improve=11.884100, (0 missing)
##       Price       < 80.5  to the right, improve= 9.809182, (0 missing)
##       ShelveLoc   splits as  L-R,       improve= 7.318561, (0 missing)
##       Age         < 50.5  to the right, improve= 4.958077, (0 missing)
##       Income      < 57.5  to the left,  improve= 4.011298, (0 missing)
##   Surrogate splits:
##       US         splits as  LR,        agree=0.782, adj=0.495, (0 split)
##       Population < 233.5 to the left,  agree=0.586, adj=0.042, (0 split)
##       Income     < 110.5 to the left,  agree=0.582, adj=0.032, (0 split)
##       Price      < 90.5  to the right, agree=0.582, adj=0.032, (0 split)
##       CompPrice  < 97.5  to the right, agree=0.577, adj=0.021, (0 split)
##
## Node number 3: 57 observations,    complexity param=0.02586207
##   predicted class=YES  expected loss=0.2280702  P(node) =0.2057762
##     class counts:    13    44
##    probabilities: 0.228 0.772
##   left son=6 (15 obs) right son=7 (42 obs)
##   Primary splits:
##       US          splits as  LR,        improve=5.632080, (0 missing)
##       Price       < 136.5 to the right, improve=5.402090, (0 missing)
##       Advertising < 2.5   to the left,  improve=3.675370, (0 missing)
##       Population  < 338   to the left,  improve=2.520175, (0 missing)
##       Education   < 14.5  to the left,  improve=1.952675, (0 missing)
##   Surrogate splits:
##       Advertising < 0.5   to the left,  agree=0.930, adj=0.733, (0 split)
##       CompPrice   < 100   to the left,  agree=0.772, adj=0.133, (0 split)
##       Price       < 142.5 to the right, agree=0.772, adj=0.133, (0 split)
##       Age         < 27.5  to the left,  agree=0.772, adj=0.133, (0 split)
##
## Node number 4: 125 observations,    complexity param=0.05172414
##   predicted class=NO   expected loss=0.184  P(node) =0.4512635
##     class counts:   102    23
##    probabilities: 0.816 0.184
##   left son=8 (113 obs) right son=9 (12 obs)
```

```
##   Primary splits:
##       Price     < 90.5  to the right, improve=8.505027, (0 missing)
##       Age       < 33.5  to the right, improve=2.518125, (0 missing)
##       CompPrice < 98.5  to the right, improve=1.706940, (0 missing)
##       ShelveLoc splits as  L-R,       improve=1.668335, (0 missing)
##       US        splits as  RL,        improve=1.370893, (0 missing)
##   Surrogate splits:
##       CompPrice < 99.5  to the right, agree=0.928, adj=0.25, (0 split)
##
## Node number 5: 95 observations,    complexity param=0.09482759
##   predicted class=YES  expected loss=0.4842105  P(node) =0.3429603
##     class counts:    46    49
##    probabilities: 0.484 0.516
##   left son=10 (33 obs) right son=11 (62 obs)
##   Primary splits:
##       Price     < 124.5 to the right, improve=9.325554, (0 missing)
##       ShelveLoc splits as  L-R,       improve=6.071176, (0 missing)
##       Income    < 57.5  to the left,  improve=4.083401, (0 missing)
##       Education < 17.5  to the right, improve=3.256249, (0 missing)
##       Age       < 49    to the right, improve=2.830409, (0 missing)
##   Surrogate splits:
##       CompPrice   < 131.5 to the right, agree=0.716, adj=0.182, (0 split)
##       Advertising < 24    to the right, agree=0.684, adj=0.091, (0 split)
##       Income      < 30.5  to the left,  agree=0.663, adj=0.030, (0 split)
##       Education   < 17.5  to the right, agree=0.663, adj=0.030, (0 split)
##
## Node number 6: 15 observations
##   predicted class=NO   expected loss=0.4  P(node) =0.05415162
##     class counts:     9     6
##    probabilities: 0.600 0.400
##
## Node number 7: 42 observations
##   predicted class=YES  expected loss=0.0952381  P(node) =0.1516245
##     class counts:     4    38
##    probabilities: 0.095 0.905
##
## Node number 8: 113 observations
##   predicted class=NO   expected loss=0.1238938  P(node) =0.4079422
##     class counts:    99    14
##    probabilities: 0.876 0.124
##
## Node number 9: 12 observations
##   predicted class=YES  expected loss=0.25  P(node) =0.0433213
##     class counts:     3     9
##    probabilities: 0.250 0.750
##
## Node number 10: 33 observations
##   predicted class=NO   expected loss=0.2121212  P(node) =0.1191336
##     class counts:    26     7
##    probabilities: 0.788 0.212
##
## Node number 11: 62 observations,    complexity param=0.04310345
##   predicted class=YES  expected loss=0.3225806  P(node) =0.2238267
##     class counts:    20    42
```

```
##    probabilities: 0.323 0.677
##   left son=22 (21 obs) right son=23 (41 obs)
##   Primary splits:
##       Age        < 65    to the right, improve=5.582256, (0 missing)
##       ShelveLoc  splits as  L-R,       improve=4.685427, (0 missing)
##       CompPrice  < 124.5 to the left,  improve=3.844637, (0 missing)
##       Income     < 57    to the left,  improve=2.482644, (0 missing)
##       Population < 437   to the left,  improve=1.703441, (0 missing)
##   Surrogate splits:
##       Income     < 34.5  to the left,  agree=0.694, adj=0.095, (0 split)
##       Education  < 17.5  to the right, agree=0.694, adj=0.095, (0 split)
##       Population < 64.5  to the left,  agree=0.677, adj=0.048, (0 split)
##       ShelveLoc  splits as  L-R,       agree=0.677, adj=0.048, (0 split)
##       US         splits as  LR,        agree=0.677, adj=0.048, (0 split)
##
## Node number 22: 21 observations,    complexity param=0.04310345
##   predicted class=NO   expected loss=0.3809524  P(node) =0.07581227
##     class counts:    13     8
##    probabilities: 0.619 0.381
##   left son=44 (14 obs) right son=45 (7 obs)
##   Primary splits:
##       CompPrice   < 123.5 to the left,  improve=4.7619050, (0 missing)
##       Advertising < 11.5  to the left,  improve=1.6932230, (0 missing)
##       ShelveLoc   splits as  L-R,       improve=1.6932230, (0 missing)
##       Population  < 348.5 to the left,  improve=1.5393770, (0 missing)
##       Education   < 14.5  to the right, improve=0.5411255, (0 missing)
##   Surrogate splits:
##       Education   < 10.5  to the right, agree=0.810, adj=0.429, (0 split)
##       Advertising < 12.5  to the left,  agree=0.762, adj=0.286, (0 split)
##       Population  < 348.5 to the left,  agree=0.762, adj=0.286, (0 split)
##       Price       < 111.5 to the left,  agree=0.762, adj=0.286, (0 split)
##
## Node number 23: 41 observations
##   predicted class=YES  expected loss=0.1707317  P(node) =0.1480144
##     class counts:     7    34
##    probabilities: 0.171 0.829
##
## Node number 44: 14 observations
##   predicted class=NO   expected loss=0.1428571  P(node) =0.05054152
##     class counts:    12     2
##    probabilities: 0.857 0.143
##
## Node number 45: 7 observations
##   predicted class=YES  expected loss=0.1428571  P(node) =0.02527076
##     class counts:     1     6
##    probabilities: 0.143 0.857
```

```
#important to look at CP- xerror, the best CP value to use is the one with smallest xerror
```