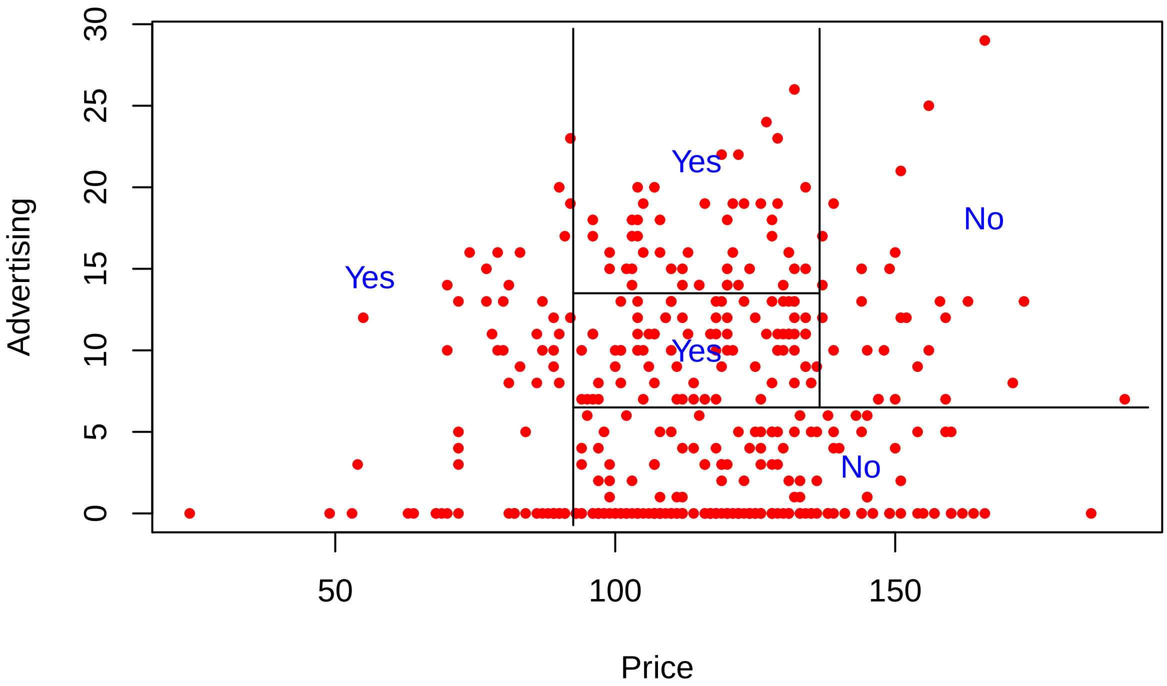
```r
# cattree.r
RNGkind(sample.kind = 'Rounding')
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

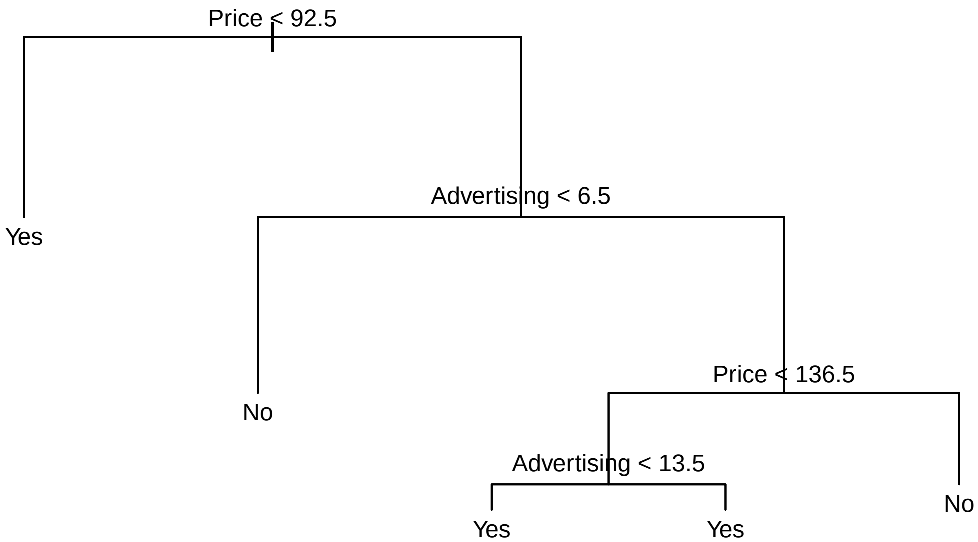
```r
library(tree)      # tree() cv.tree()
library(ISLR)      # data set

d0=Carseats
str(d0)
```

```
## '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 ...
```

```r
? Carseats
# Education is the average education (years) in the local population
#
# there are 10 predictors, some categorical
n = nrow(d0)

# create categorical response
high=ifelse(d0$Sales<=8,"No","Yes")
d1=data.frame(d0,high)

# tree - 2 predictors, full dataset
#=================================================================
tree0=tree(high~Price+Advertising,d1)

# scatterplot on predictors space (Price~Advertising does not work?)
plot(Advertising~Price,d1,pch=19,cex=0.6,col="red")
# regions and predicted category
partition.tree(tree0,add = T,col="blue")
```




```
#
# tree plot
plot(tree0)
text(tree0,cex=0.75)
```




```
#
# Combining plot and tree diagram, we conclude
#
# If price < $92.5 store is High Sales
# If price > $136.5 store is Low Sales
# If $92.5 < price < $136.5 store is High Sales if Advertising > $6.5
#
#
#
#
```

```r
# full model -Sales
#=================================================================
tree1=tree(high~.-Sales,d1)
summary(tree1)
```

```
##
## Classification tree:
## tree(formula = high ~ . - Sales, data = d1)
## 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
```

```r
#
# This is a large tree with 27 regions
#
# Total deviance 170.7 is sum of deviances of terminal nodes
#   with 400-27 = 373 dof

# misclassifications
ypred = predict(tree1,d1,type="class")
table(ypred,d1$high)
```

```
##
## ypred  No Yes
##   No  213  13
##   Yes  23 151
```

```r
# there are 36 misclassified obs
# training error rate 36/400 = 0.09

names(tree1)
```

```
## [1] "frame"   "where"   "terms"   "call"    "y"       "weights"
```

```r
dim(tree1$frame)   # [1] 53  6
```

```
## [1] 53  6
```

```r
head(tree1$frame)
```

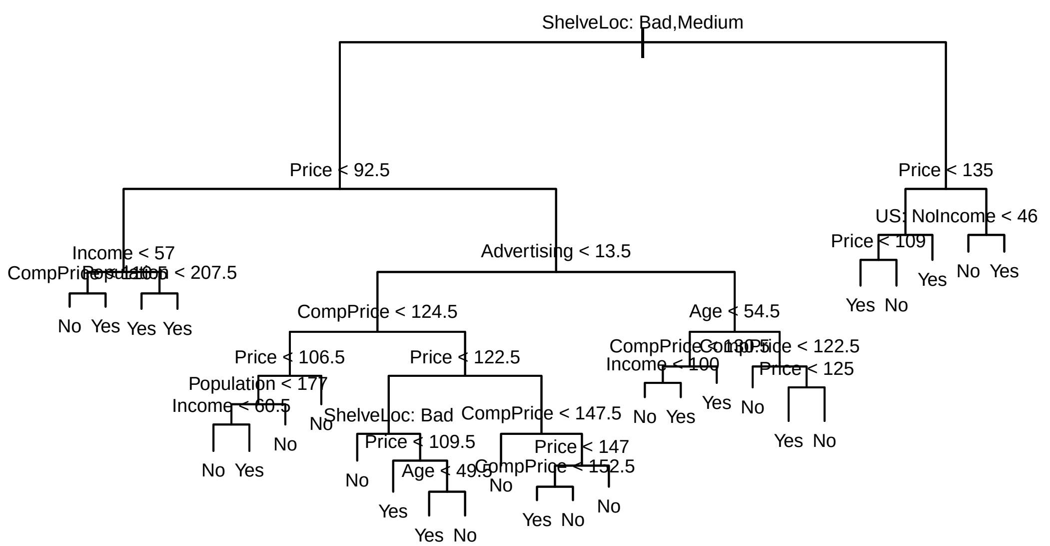
```
##          var   n        dev yval splits.cutleft splits.cutright  yprob.No
## 1  ShelveLoc 400 541.486837   No            :ac              :b 0.5900000
## 2      Price 315 390.591685   No          <92.5           >92.5 0.6888889
## 4     Income  46  56.534305  Yes            <57             >57 0.3043478
## 8  CompPrice  10  12.217286   No         <110.5          >110.5 0.7000000
## 16    <leaf>   5   0.000000   No                                1.0000000
## 17    <leaf>   5   6.730117  Yes                                0.4000000
##    yprob.Yes
## 1  0.4100000
## 2  0.3111111
## 4  0.6956522
## 8  0.3000000
## 16 0.0000000
## 17 0.6000000
```

```r
tail(tree1$frame)
```

```
##        var  n      dev yval splits.cutleft splits.cutright   yprob.No  yprob.Yes
## 24 <leaf>  8  0.00000  Yes                                0.00000000 1.00000000
## 25 <leaf>  9 11.45726   No                                0.66666667 0.33333333
## 13 <leaf> 51 16.87524  Yes                                0.03921569 0.96078431
## 7  Income 17 22.07444   No            <46             >46 0.64705882 0.35294118
## 14 <leaf>  6  0.00000   No                                1.00000000 0.00000000
## 15 <leaf> 11 15.15820  Yes                                0.45454545 0.54545455
```

```r
# Factor ShelveLoc is most important classifier
# <leaf> rows are terminal nodes

plot(tree1)
text(tree1,cex=0.6,pretty=0)    # pretty shows class names on tree
title("Tree from the full dataset")
```

Tree from the full dataset

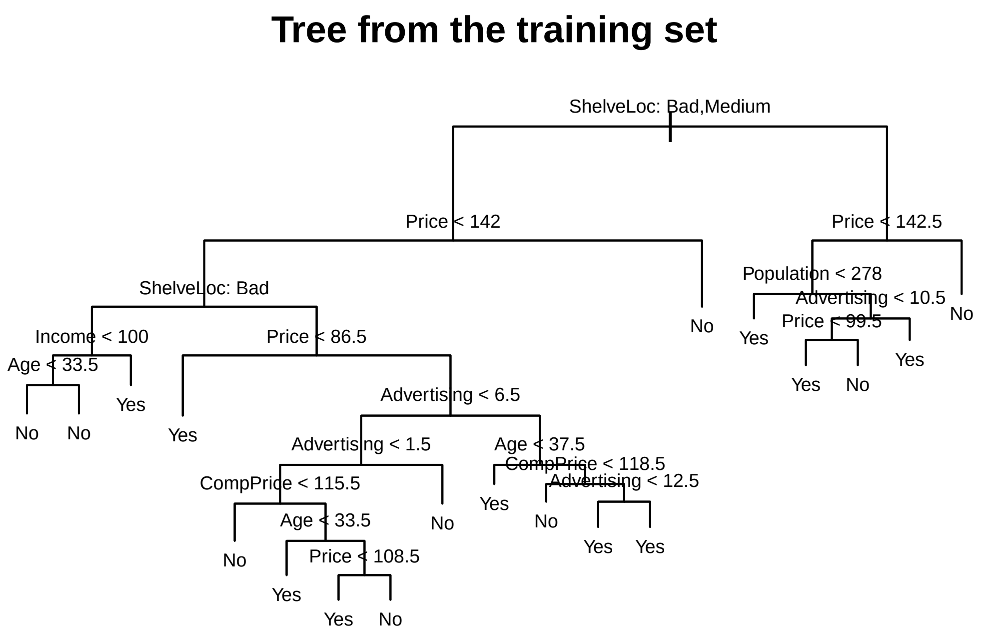



```r
# Bad, medium is the left-hand branch
# 1st branch differentiates good locations from bad & medium

# training and test sets
#=====================================================================
set.seed(2)
n = 1:nrow(d1)
train=sample(n, 200)
d1.test=d1[-train,]        # test set with response
y.test=high[-train]    # response in test set

# training model
tree2=tree(high~.-Sales,d1,subset=train)

summary(tree2)
```

```
##
## Classification tree:
```

```
## tree(formula = high ~ . - Sales, data = d1, subset = train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "Age"         "Advertising"
## [6] "CompPrice"   "Population"
## Number of terminal nodes:  19
## Residual mean deviance:  0.4282 = 77.51 / 181
## Misclassification error rate: 0.105 = 21 / 200
```

```
#
# Training accuracy rate is 90%
#
plot(tree2)
text(tree2,cex=0.6,pretty=0)
title("Tree from the training set")
```




```
# test error rate
pred2=predict(tree2,d1.test,type="class")
#
# use type="class" to get Yes,No predictions

table(pred2,y.test)
```

```
##       y.test
## pred2 No Yes
##   No  86  27
##   Yes 30  57
```

```
#
# out of 200 obs

aux=prop.table(table(pred2,y.test))
sum(diag(aux))    #    test accuracy rate
```

```
## [1] 0.715
```

```r
# CV on (mis)classification error rate
#=========================================================================
set.seed(3)
tree3=cv.tree(tree2,FUN=prune.misclass)  # compare misclassifications

names(tree3)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```r
tree3$size
```

```
## [1] 19 17 14 13  9  7  3  2  1
```

```r
tree3$dev
```

```
## [1] 55 55 53 52 50 56 69 65 80
```

```r
#
# complexity parameter
#
round(tree3$k,2)
```

```
## [1]  -Inf  0.00  0.67  1.00  1.75  2.00  4.25  5.00 23.00
```

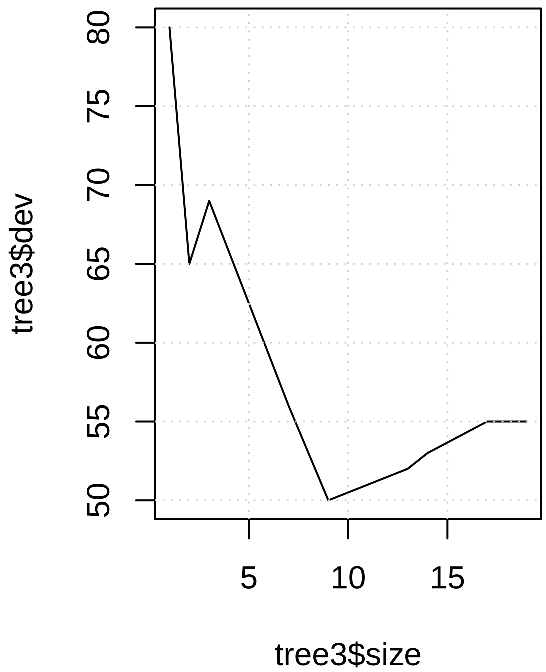
```r
# size values are n. terminal nodes
# dev is n. obs. misclassified  (the CV error rate)
# k is alpha = complexity parameter
# tree with 9 terminal nodes has lowest dev (CV error rate)


# cv on deviance - to compare
#
tree4=cv.tree(tree2)
tree4$size
```

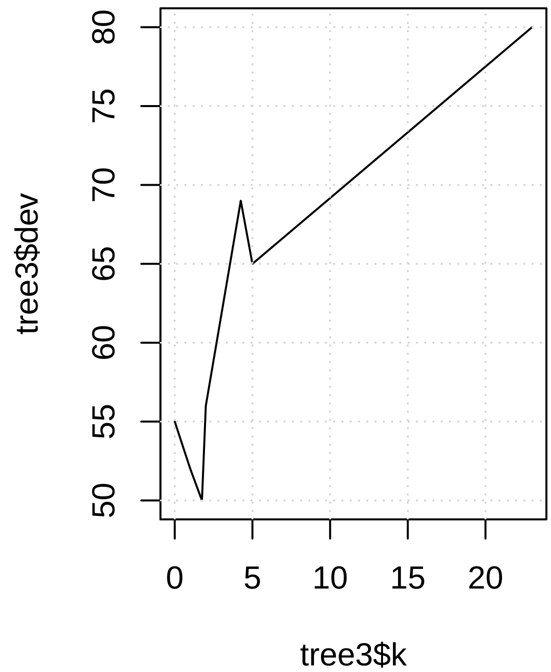
```
##  [1] 19 16 14 13 12 11 10  9  8  7  6  5  3  2  1
```

```r
round(tree4$dev,1)
```

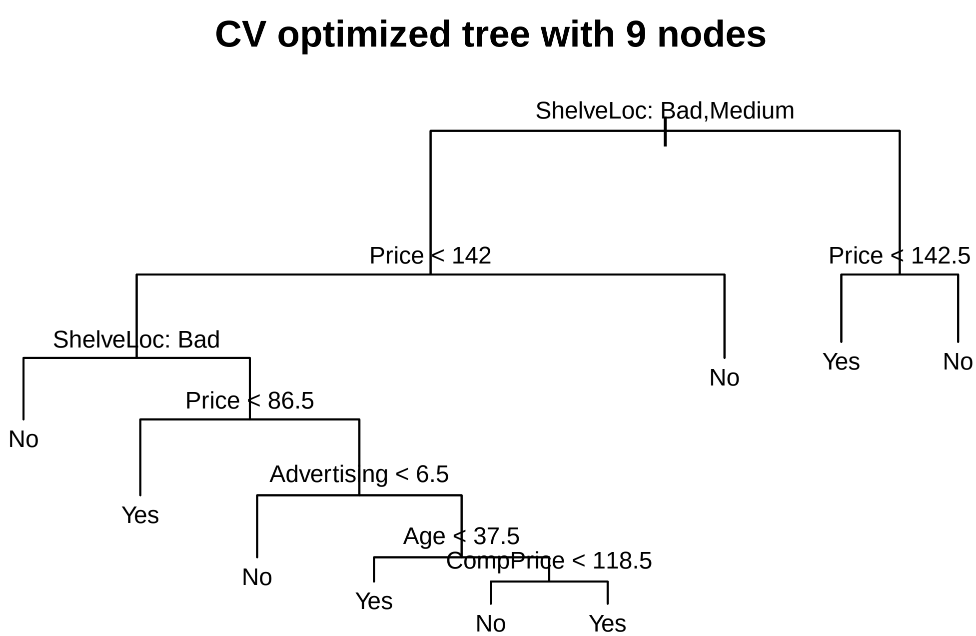
```
##  [1] 555.2 513.0 503.2 485.6 485.6 409.9 395.3 366.6 342.5 343.9 335.3 311.8
## [13] 310.1 290.4 278.6
```

```r
#
# did not work, since smallest deviance is for tree with one terminal node

# plot n. of misclassifications vs size, k
#
par(mfrow=c(1,2))
plot(tree3$dev~tree3$size,type="l");grid()
plot(tree3$dev~tree3$k,type="l");grid()
```






```r
#
# smallest n. of misclassifications
# with 9 terminal nodes
# with k = 1.75

# use FUN=prune.misclass to
# prune training tree2 to 9 terminal nodes
prune9=prune.misclass(tree2,best=9)
par(mfrow=c(1,1))
plot(prune9)
text(prune9,cex=0.75,pretty=0)
title("CV optimized tree with 9 nodes")
```



```r
# test error of the pruned tree
yhat9=predict(prune9,d1.test,type="class")
table(yhat9,y.test)
```

```
##      y.test
## yhat9 No Yes
##   No  94  24
##   Yes 22  60
```

```r
#

aux=prop.table(table(yhat9,y.test))
sum(diag(aux))    #[1] 0.77  test accuracy rate
```

```
## [1] 0.77
```

```r
summary(prune9)
```

```
##
## Classification tree:
## snip.tree(tree = tree2, nodes = c(159L, 6L, 8L, 38L))
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Advertising" "Age"         "CompPrice"
## Number of terminal nodes:  9
## Residual mean deviance:  0.8103 = 154.8 / 191
## Misclassification error rate: 0.155 = 31 / 200
```