Tree based methods II-b

Classification Tree examples (using tree)

STAT 32950-24620

Spring 2023 (5/16)

Classification Tree Example - Using tree library

```
#library(rpart)       # better plots using rpart.plot
#library(rpart.plot)  # better plots
library(tree)   # alternative to rpart, better summary
library(ISLR); attach(Carseats); 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 13
##  $ 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
##  $ Price      : num  120 83 80 97 128 72 108 120 124 12
##  $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium'
##  $ 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 
##  $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 
```

Create a binary response

For fitting a tree to predict High using all variables but Sales.

```
High=as.factor(ifelse(Sales<=8,"No","Yes"))
table(High)   # No 236; Yes 164
```

```
## High
##  No Yes
## 236 164
```

```
#table(ShelveLoc) # Bad 96    Good 85  Medium  219
Carseats=data.frame(Carseats,High)      # 'data.frame': 40
colnames(Carseats)
```

```
##  [1] "Sales"       "CompPrice"   "Income"      "Advertis
##  [6] "Price"       "ShelveLoc"   "Age"         "Educatio
## [11] "US"          "High"
```

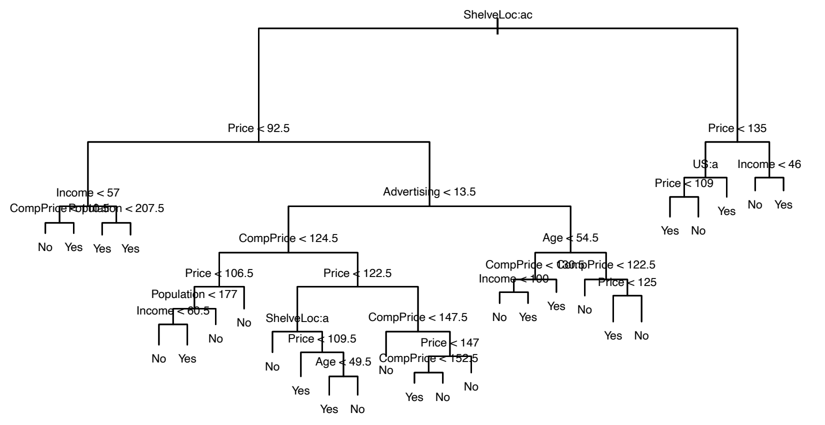
Fit a tree with tree library

```
tree.carseats=tree(High~.-Sales,Carseats)
summary(tree.carseats)
```

```
##
## Classification tree:
## tree(formula = High ~ . - Sales, data = Carseats)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "CompPrice
## [6] "Advertising" "Age"         "US"
## Number of terminal nodes:  27
## Residual mean deviance:  0.458 = 171 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

Use tree plot

```
plot(tree.carseats);   # some terminal nodes are combined
text(tree.carseats,cex=0.5,digits=2)
```



Training, testing, prediction error using tree

To avoid overfitting, using training and testing proceses.

```
set.seed(3)
train=sample(1:nrow(Carseats), 200)
Carseats.test=Carseats[-train,]
High.test=High[-train]
Ttree.carseats=tree(High~.-Sales,Carseats,subset=train)
```

Check the fit on training data

```
Ttree.carseats=tree(High~.-Sales,Carseats,subset=train)
summary(tree(High~.-Sales,Carseats,subset=train))
```

```
##
## Classification tree:
## tree(formula = High ~ . - Sales, data = Carseats, subset
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Advertising" "CompPrice"   "Price"
## [6] "Age"         "Income"      "Education"
## Number of terminal nodes:  20
## Residual mean deviance:  0.4 = 72 / 180
## Misclassification error rate: 0.1 = 20 / 200
```

Check the fit on testing data

```
#Ttree.carseats=tree(High~.-Sales,Carseats,subset=train)
tree.pred=predict(Ttree.carseats,Carseats.test,type="class'
```

Check classification error rate

```
table(tree.pred,High.test)  # check the error rate
```

```
##          High.test
## tree.pred No Yes
##       No  88  41
##       Yes 23  48
```

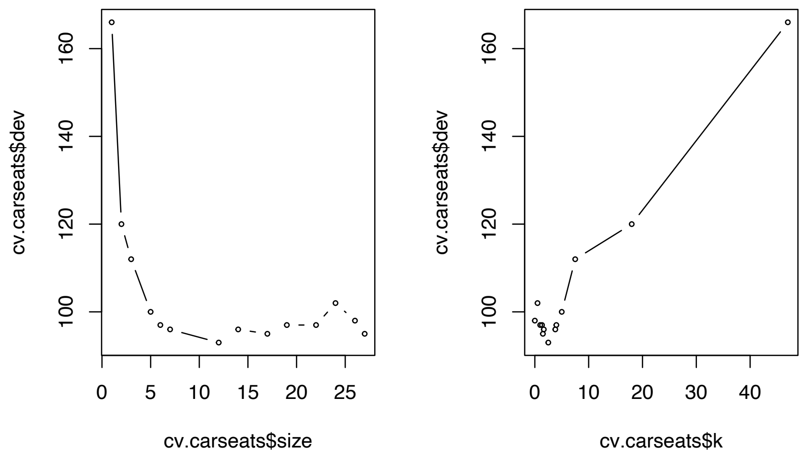
Cross-validation

```
set.seed(3)
cv.carseats=cv.tree(tree.carseats,FUN=prune.misclass)
# names(cv.carseats) # "size"   "dev"   "k"   "method"
cv.carseats
```

```
## $size
##  [1] 27 26 24 22 19 17 14 12  7  6  5  3  2  1
##
## $dev
##  [1]  95  98 102  97  97  95  96  93  96  97 100 112 12
##
## $k
##  [1]   -Inf  0.000  0.500  1.000  1.333  1.500  1.667
## [11]  5.000  7.500 18.000 47.000
##
## $method
## [1] "misclass"
##
```

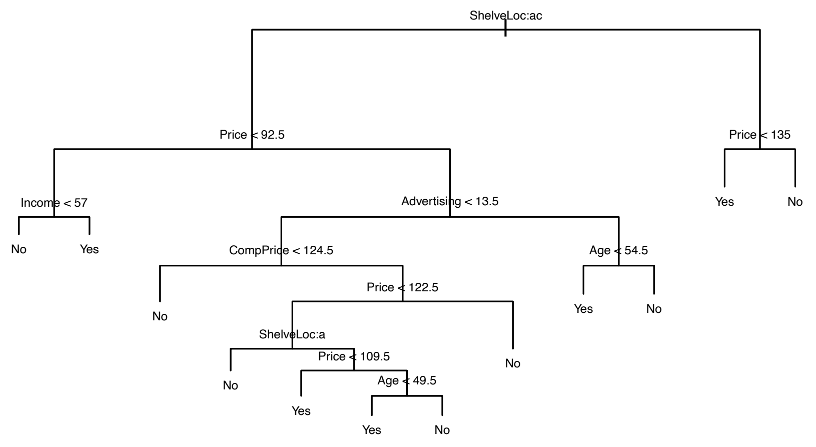
Compare error across size

```
par(mfrow=c(1,2))
plot(cv.carseats$size,cv.carseats$dev,type="b",cex=.5)
plot(cv.carseats$k,cv.carseats$dev,type="b",cex=.5)
```



Alternative criteria: Using mis-classification rate

```
prune.carseats=prune.misclass(tree.carseats,best=9)
plot(prune.carseats)
text(prune.carseats,cex=0.5,digits=2)
```



Check the cv-pruned tree

```
summary(prune.carseats)
```

```
##
## Classification tree:
## snip.tree(tree = tree.carseats, nodes = c(9L, 22L, 7L,
## 6L, 43L, 23L))
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "Advertis
## [6] "Age"
## Number of terminal nodes:  12
## Residual mean deviance:  0.783 = 304 / 388
## Misclassification error rate: 0.14 = 56 / 400
```

Testing error of the cv-pruned tree

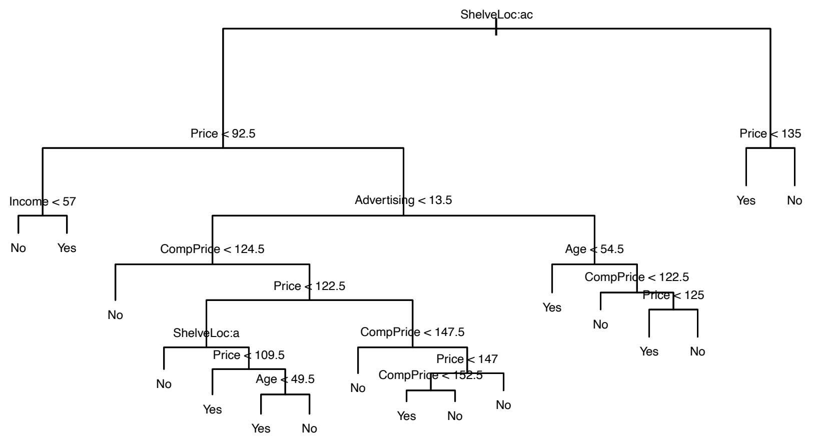
```
tree.pred=predict(prune.carseats,
                  newdata=Carseats.test,type="class")
table(tree.pred,High.test)
```

```
##          High.test
## tree.pred  No Yes
##       No  100  12
##       Yes  11  77
```

```
# (100+77)/200=.885, (97+58)/200 =.775 for another seed(9)
```

Use best to get large tree with lower err

```
prune.carseats=prune.misclass(tree.carseats,best=15)
plot(prune.carseats)
text(prune.carseats,cex=0.5,digits=2)
```



Testing error of the larger tree

```
tree.pred=predict(prune.carseats,Carseats.test,
                        type="class")
table(tree.pred,High.test)
```

```
##          High.test
## tree.pred  No Yes
##       No  100  11
##       Yes  11  78
```

```
# (100+78)/200=.89, (102+53)/200 =.775 for another seed
```

More details - nodes information, controlling parameters

```
#tree.carseats #show node,split,n,deviance,yval,(yprob)
```

```
#control = rpart.control(...)

#minsplit - the minimum number of observations
            that must exist in a node
#minbucket  - the minimum number of observations
              in any terminal node.
#cp - complexity parameter.
#Any split that does not decrease the overall lack of fit
#by a factor of cp is not attempted.

#xval- number of cross-validations.

#maxdepth - Set the maximum depth of any node
of the final tree, root node = 0
```