Tree based methods II

Classification Tree examples (using rpart)

STAT 32950-24620

Spring 2025 (wk9)

Classification Tree

Building a **classification tree** is similar to building a regression tree.

Regression Tree: The response variable is continuous.

Classification Tree: The response variable is categorical.

```
library(rpart)       # better plots using rpart.plot
library(rpart.plot)  # better plots
#library(tree)  # alternative to rpart, concise summary
library(ISLR)
```

R library:

rpart — Recursive Partitioning And Regression Trees

cart — Classification and Regression Trees

tree — (duh)

Example data

```
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) #'d.f': 400 obs 12 vars
colnames(Carseats)
```

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

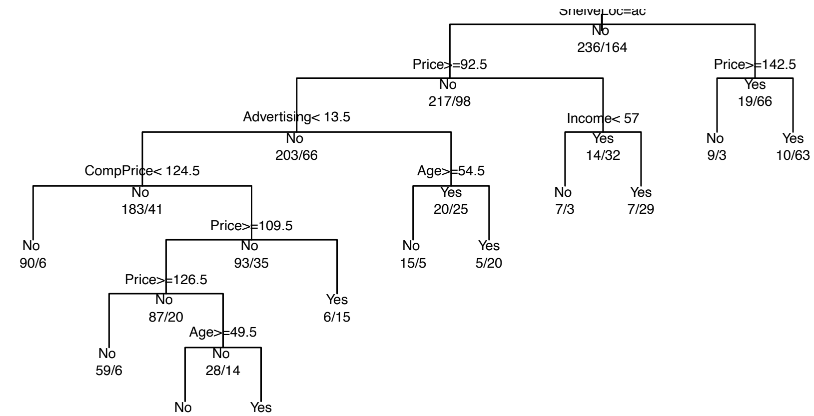
Fit a classification tree using library(rpart)

```r
Rtree.carseats=rpart(High~.-Sales,Carseats)
attributes(Rtree.carseats)
```

```
## $names
##  [1] "frame"               "where"               "call"
##  [4] "terms"               "cptable"             "methoc
##  [7] "parms"               "control"             "funct:
## [10] "numresp"             "splits"              "csplit
## [13] "variable.importance" "y"                   "ordere
##
## $xlevels
## $xlevels$ShelveLoc
## [1] "Bad"    "Good"   "Medium"
##
## $xlevels$Urban
## [1] "No"  "Yes"
##
## $xlevels$US
```

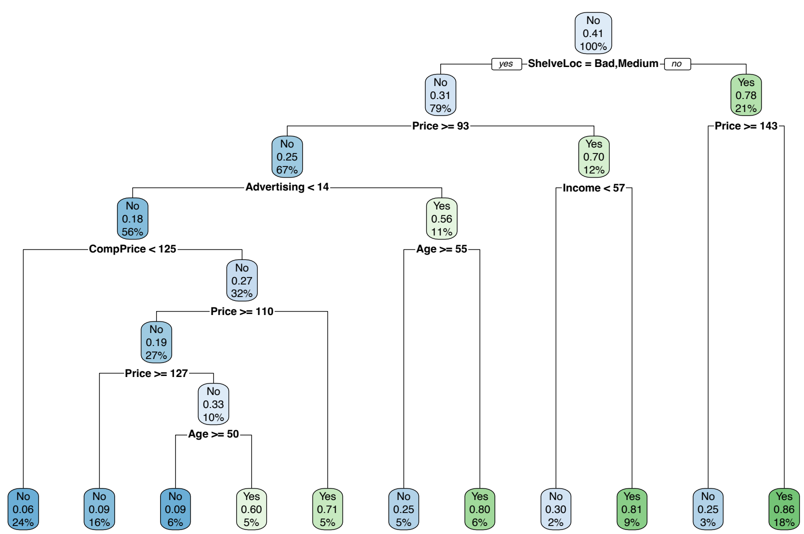

Plot a classification tree

```r
# Plot classification tree using (rpart)
plot(Rtree.carseats,uniform=T)
text(Rtree.carseats,use.n=T,all=T,cex=.6,digit=3)
```



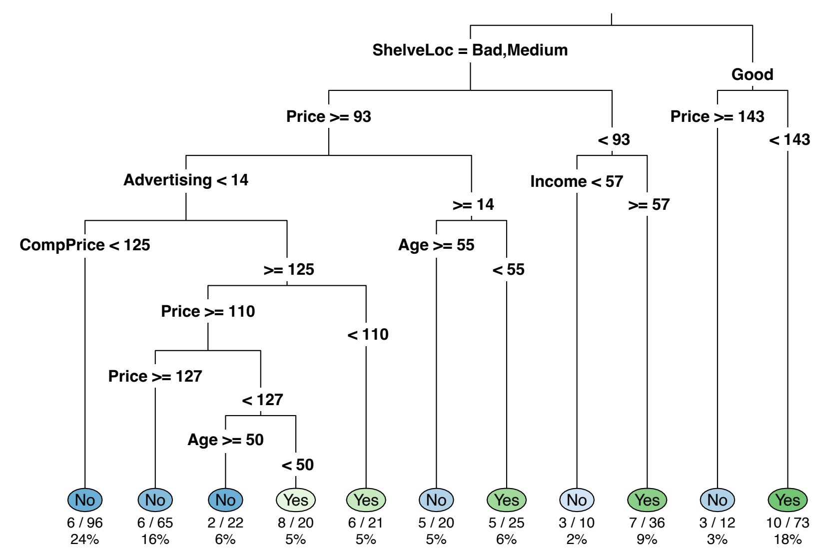
Plot a nicer tree using library(rpart.plot)

```r
rpart.plot(Rtree.carseats)
```



Plot the tree with alternative style

```r
rpart.plot(Rtree.carseats,extra=103, under=T,type=3)
```



Notations at the nodes

At each each node

- Condition: splitting rule of a variable.
  E. g. `Price >= 93`

- Decision: Response variable classification rule.
  E. g. `No` when `Price >= 143`

- Ratio of the response variable (if given)
  E.g. 10/73: 10 `No`, 63 `yes` at the leave; ruled as `Yes` = `High`
  E.g. 3/12: 3 `Yes`, 9 `No` at the leave; ruled as `No` = `Low`
  (alternative display: 3/9 or 9/3: 3 `Yes`, 9 `No`)

Splitting nodes and ending nodes

Splitting nodes

- Left branch: Condition in the label satisfied.
  E. g. Label: `Price >= 93`
  The left branch below have `Price >= 93`.

- Right branch: Condition in the label not satisfied.
  E. g. Label: `Price >= 93`
  The right branch below have `Price < 93`.

Ending nodes (leaves)

- Decision rule (`Yes` or `No`)

Variable predictability

```
table(High,ShelveLoc);table(High,Urban);table(High,US)
```

```
##      ShelveLoc
## High  Bad Good Medium
##   No   82   19    135
##   Yes  14   66     84

##      Urban
## High   No Yes
##   No   64 172
##   Yes  54 110

##      US
## High   No Yes
##   No  101 135
##   Yes  41 123
```


Examine the fit using summary(rpart(. . .),digits=3)

```
Call:
rpart(formula = High ~ . - Sales, data = Carseats)
  n= 400
      CP nsplit rel error xerror   xstd
1 0.2866      0     1.000  1.000 0.0600
2 0.1098      1     0.713  0.713 0.0555
3 0.0457      2     0.604  0.683 0.0548
4 0.0366      4     0.512  0.720 0.0556
5 0.0274      5     0.476  0.713 0.0555
6 0.0244      7     0.421  0.695 0.0551
7 0.0122      8     0.396  0.652 0.0540
8 0.0100     10     0.372  0.622 0.0532
Variable importance
      Price   ShelveLoc    Age  Advertising    CompPrice
         34          25     11           11            9
  Income  Population  Education
       5           3          1
```

Node details

Node index: 1-11; 16-19; 34, 35; 68, 69; 138, 139.

```
Node number 1: 400 observations,  complexity param=0.287
  predicted class=No   expected loss=0.41  P(node) =1
    class counts:   236   164
   probabilities: 0.590 0.410
  left son=2 (315 obs) right son=3 (85 obs)
  Primary splits:
  ShelveLoc   splits as  LRL,      improve=29.00,(0 missing
  Price       < 92.5 to the right,improve=19.50,(0 missing
  Advertising < 6.5  to the left, improve=17.30,(0 missing
  Age         < 61.5 to the right,improve= 9.26,(0 missing
  Income      < 60.5 to the left, improve= 7.25,(0 missing
... ...
Node number 139: 20 observations
  predicted class=Yes  expected loss=0.4  P(node) =0.05
    class counts:     8    12
   probabilities: 0.400 0.600
```


Training, testing, prediction error

Using all the data to fit one treee often results in overfit.

To avoid overfit, split data to training and testing sets.

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

Next: fit a tree on training data, check the fit on testing data

Evaluate the fitted tree on training data (summary output)

```
## Call:
## rpart(formula= High~.-Sales,data=Carseats,subset=train)
##   n= 200
##         CP nsplit rel error xerror    xstd
## 1 0.28000      0    1.0000 1.0000 0.09129
## 2 0.08000      1    0.7200 0.8267 0.08721
## 3 0.05333      3    0.5600 0.8400 0.08759
## 4 0.04000      4    0.5067 0.8267 0.08721
## 5 0.02667      5    0.4667 0.8267 0.08721
## 6 0.01333      6    0.4400 0.8400 0.08759
## 7 0.01000      7    0.4267 0.8667 0.08832
##
## Variable importance
##   ShelveLoc       Price Advertising   CompPrice
##          25          20          17          10
##   Age  Population   US   Education       Income
##    10           7    6           4           3
```


Check the fit on testing data

```
Rtree.pred=predict(Rtree.carseats,Carseats.test,
                   type="class")
table(Rtree.pred,High.test)  # check the error rate
```

```
##           High.test
## Rtree.pred No Yes
##        No  85  32
##        Yes 26  57
```

Prune: use Cost Complexity (or Weakest Link)

```
Rtree.carseats$cptable # Using `rpart` to prune
```

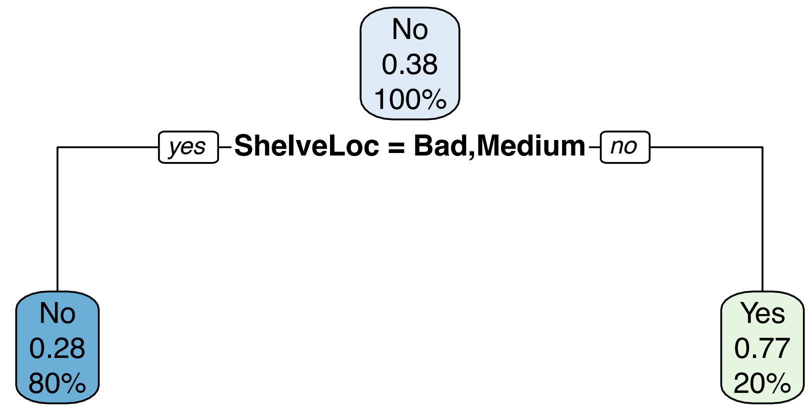
```
##         CP nsplit rel error xerror    xstd
## 1 0.28000      0    1.0000 1.0000 0.09129
## 2 0.08000      1    0.7200 0.7200 0.08371
## 3 0.05333      3    0.5600 0.7600 0.08512
## 4 0.04000      4    0.5067 0.7733 0.08556
## 5 0.02667      5    0.4667 0.7467 0.08466
## 6 0.01333      6    0.4400 0.8133 0.08682
## 7 0.01000      7    0.4267 0.8000 0.08641
```

```
# which.min(Rtree.carseats$cptable[,"xerror"]);  #2
Rtree.carseats$cptable[
  which.min(Rtree.carseats$cptable[,"xerror"]),"CP"]
```

```
## [1] 0.08
```

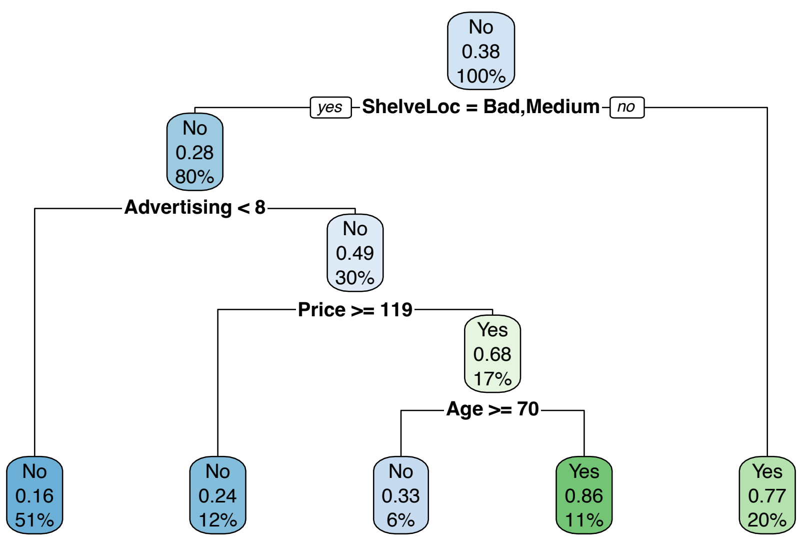

Plot the min CP tree

```
pfit.min<- prune(Rtree.carseats,cp=Rtree.carseats$cptable[
  which.min(Rtree.carseats$cptable[,"xerror"]),"CP"])
rpart.plot(pfit.min)
```



Pick a pruned tree

```
pfit2<- prune(Rtree.carseats, cp=0.05) #.1:2 leaf min tr
rpart.plot(pfit2)
```



Pick another pruned tree

```
pfit3<- prune(Rtree.carseats, cp=0.01)
rpart.plot(pfit3)
```