Introduction to Classification & Regression Trees

ISLR Chapter 8

March 27, 2017

Classification and Regression Trees

Carseat data from ISLR package

- Binary Outcome `High` 1 if Sales > 8, otherwise 0
- Fit a Classification tree model to `Price` and `Income`
- Pick a predictor and a cutpoint to split data

$$X_j \leq s \text{ and } X_k > s$$

to minimize deviance (or SSE for regression) - leads to a root node in a tree

- continue splitting/partitioning data until stopping criterion is reached (number of observations in a node > 10 and within node deviance > 0.01 deviance of the root node)
- Prediction is mean or proportion of successes of data in terminal nodes
- Output is a decision tree
- regression or classification function is nonlinear in predictors
- Captures interactions

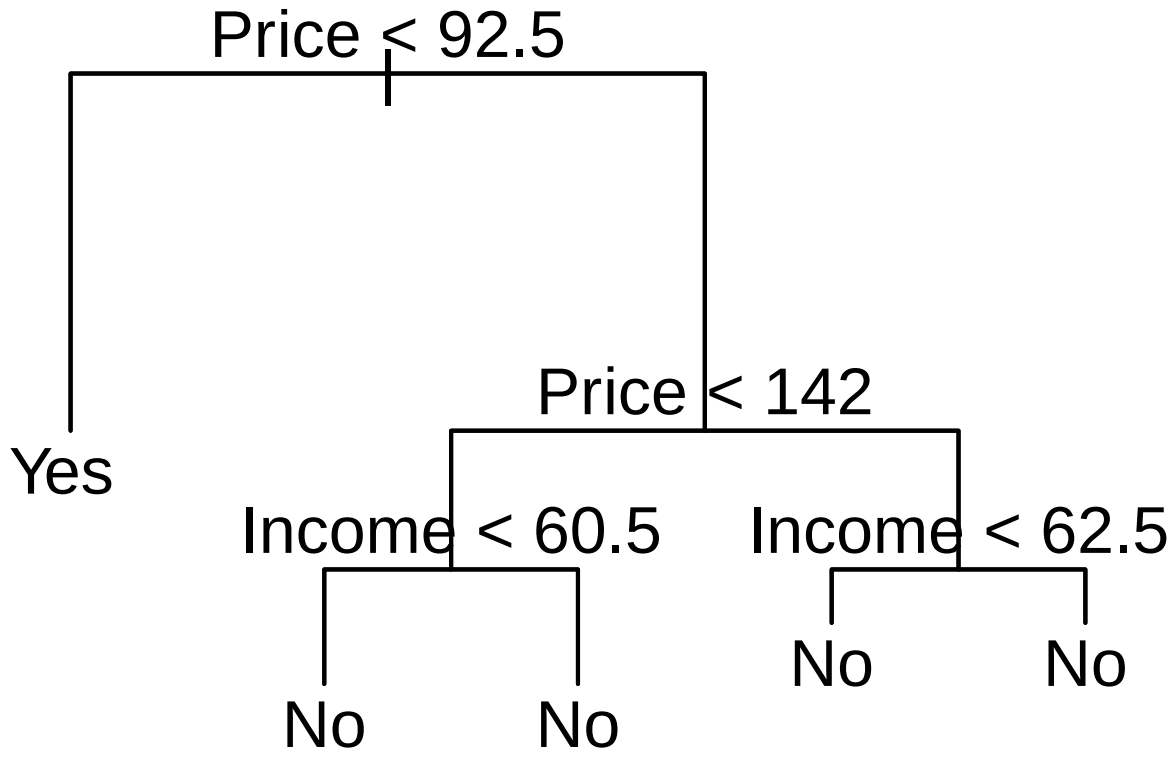
Carseat Example

```
library(tree)
data(Carseats)
  Carseats = mutate(Carseats, High=factor(ifelse (
  Carseats$Sales <=8,
    "No",
    "Yes "))
  )

tree.carseats = tree(High ~ Price + Income,
                     data=Carseats)
```

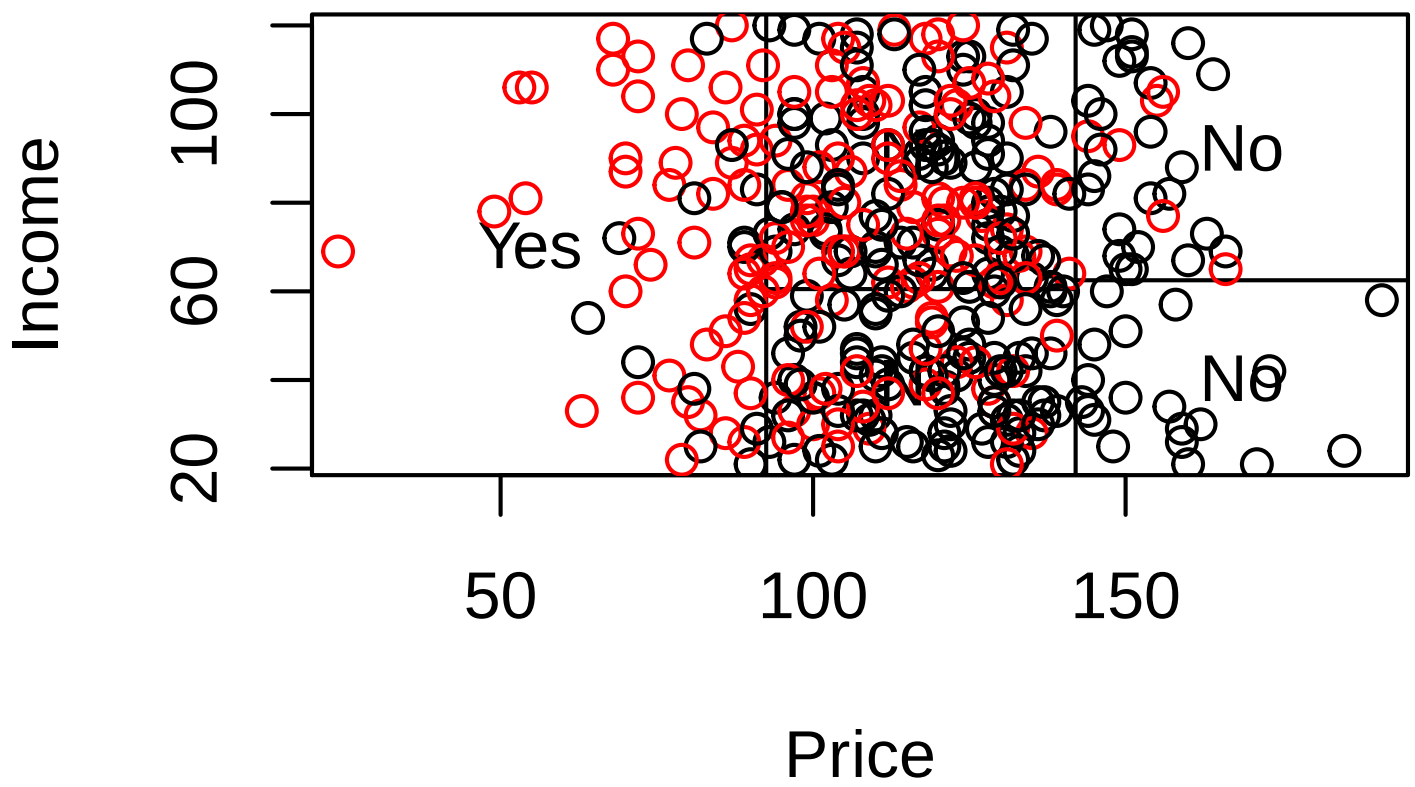
Carseat Example

```
plot(tree.carseats)
text(tree.carseats)
```



Partition

```
partition.tree(tree.carseats)
points(Carseats$Price,Carseats$Income, col=Carseats$High)
```



Splits

```
tree.carseats

## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 400 541.50 No ( 0.5900 0.4100 )
##    2) Price < 92.5 62  66.24 Yes  ( 0.2258 0.7742 ) *
##    3) Price > 92.5 338 434.80 No ( 0.6568 0.3432 )
##      6) Price < 142 287 382.10 No ( 0.6167 0.3833 )
##       12) Income < 60.5 113 128.70 No ( 0.7434 0.2566 )
##       13) Income > 60.5 174 240.40 No ( 0.5345 0.4655 )
##      7) Price > 142 51  36.95 No ( 0.8824 0.1176 )
##       14) Income < 62.5 19   0.00 No ( 1.0000 0.0000 ) *
##       15) Income > 62.5 32  30.88 No ( 0.8125 0.1875 ) *
```

Summary

```
 summary(tree.carseats)

##
## Classification tree:
## tree(formula = High ~ Price + Income, data = Carseats)
## Number of terminal nodes:  5
## Residual mean deviance:  1.18 = 466.2 / 395
## Misclassification error rate: 0.325 = 130 / 400
```

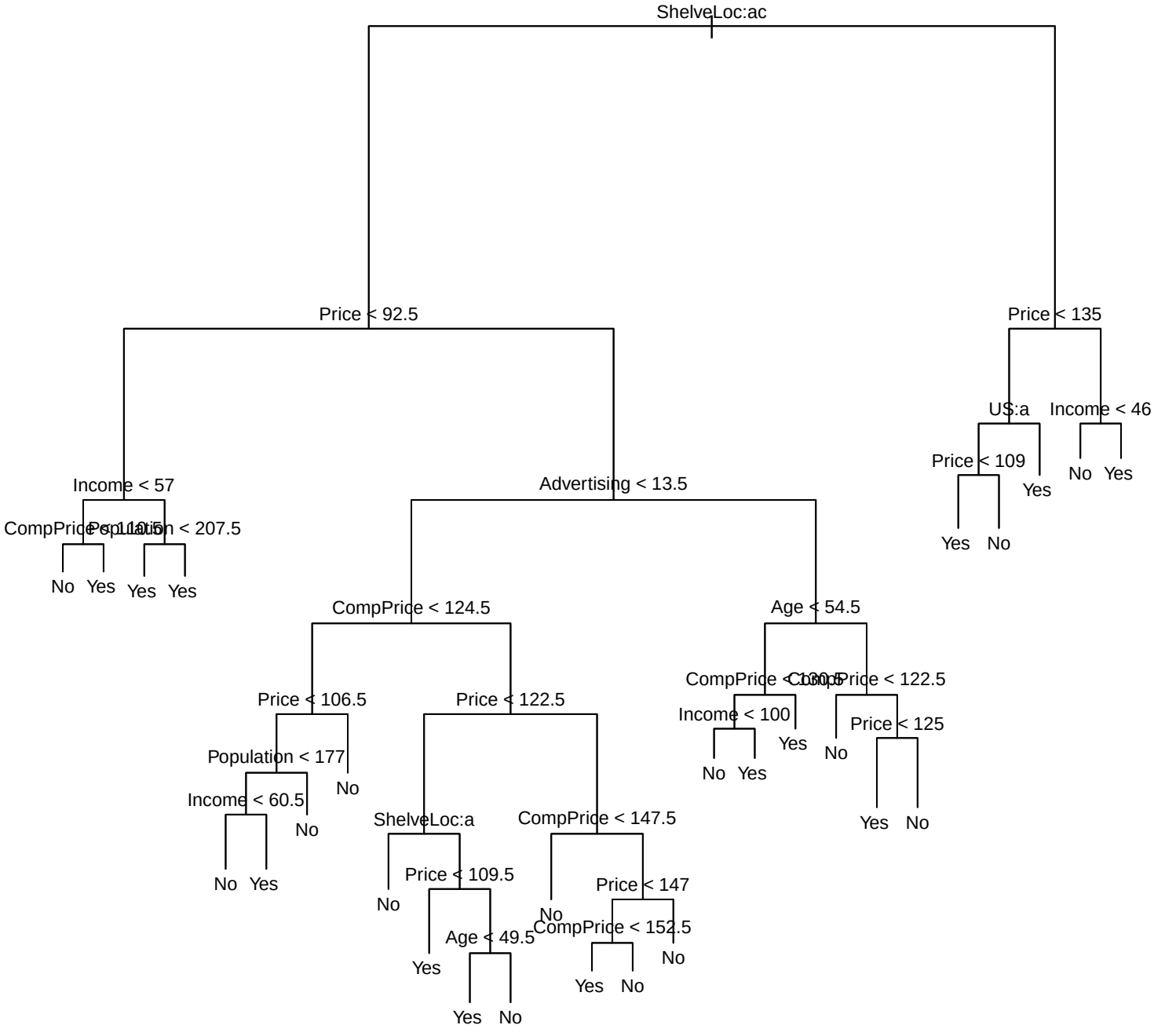
All Variables

```
tree.carseats =tree(High ~  . -Sales, data=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:  0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

Overfitting?

Tree


ShelveLoc:ac
Price < 92.5
Price < 135
Income < 57
Advertising < 13.5
US:a
Income < 46
Price < 109
No
Yes
Yes
No
Yes
No
Yes
Yes
CompPrice < 124.5
Age < 54.5
Price < 106.5
Price < 122.5
Income < 100
Price < 125
Yes
No
No
Yes
Yes
No
Population < 177
No
Income < 60.5
No
No
Yes
ShelveLoc:a
CompPrice < 147.5
Price < 109.5
Price < 147
No
No
Age < 49.5
CompPrice < 152.5
No
Yes
Yes
No
Yes
No

Classification Error

```
set.seed (2)
train=sample (1: nrow(Carseats ), 200)
Carseats.test=Carseats [-train ,]

tree.carseats =tree(High ~  . -Sales,
                    data=Carseats,subset=train )
tree.pred=predict (tree.carseats ,Carseats.test ,type ="cla
table(tree.pred , Carseats.test$High)

##
## tree.pred No Yes
##       No   86   27
##       Yes  30   57
```

```
 (30 + 27) /200    # classification error

## [1] 0.285
```

Cost-Complexity Pruning

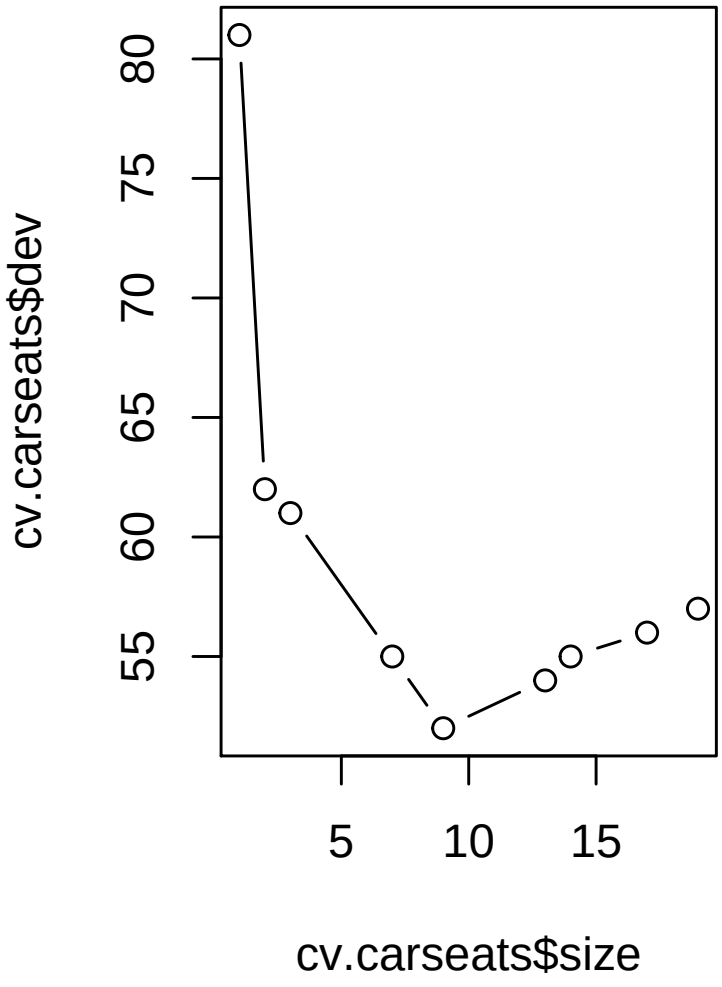
1. Grow a large tree on training data, stopping when each terminal node has fewer than some minimum number of observations
2. Prediction for region m is the Class c that $max_c \hat{\pi}_{mc}$
3. Snip off the least important splits via cost-complexity pruning to the tree in order to obtain a sequence of best subtrees indexed by cost parameter k,

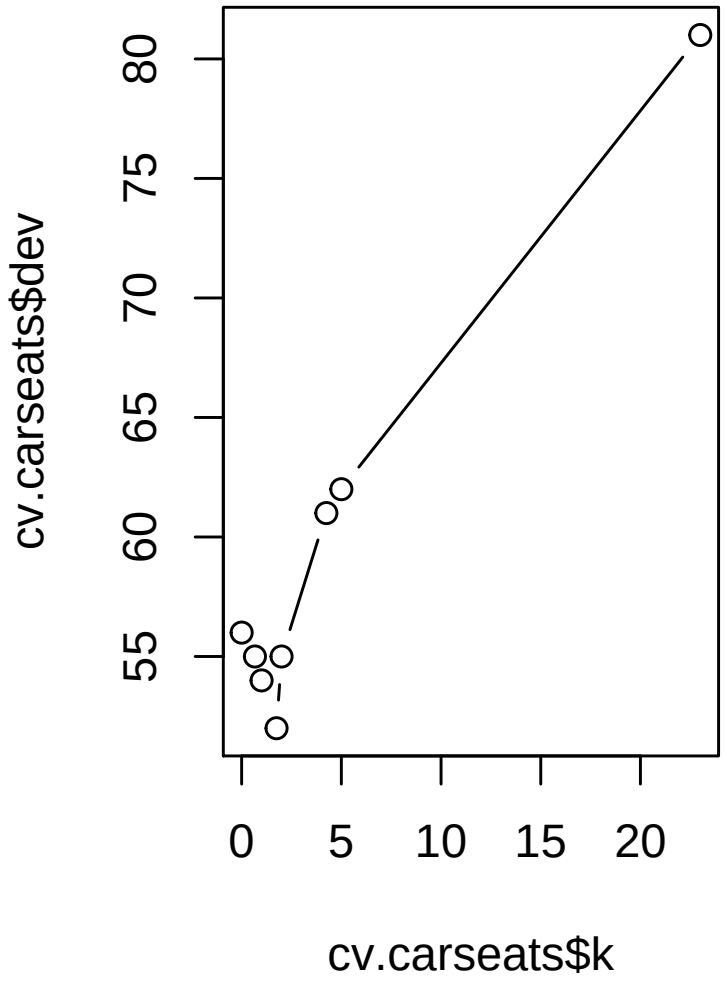
$$\frac{N_{miss}}{N} - k|T|$$

missclassification error penalized by number of terminal nodes
4. Using K-fold cross validation, compute average cost-complexity for each k
5. Pick subtree with smallest penalized error

Pruning via Cross Validation)

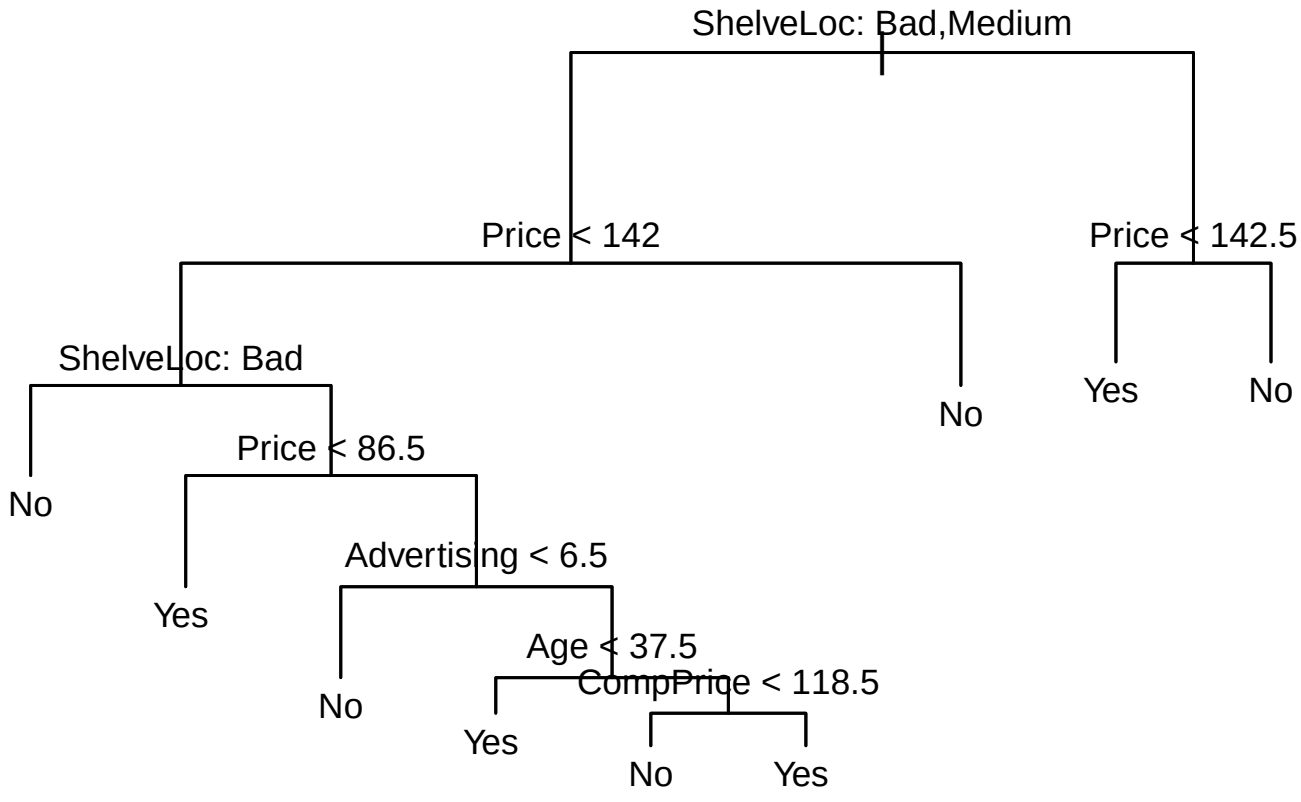
```
set.seed(2)
cv.carseats = cv.tree(tree.carseats, FUN=prune.misclass)
```





Pruned

```
prune.carseats = prune.misclass(tree.carseats ,best =9)
```



Miss-classification after Selection

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

##
## tree.pred No Yes
##       No   94   24
##       Yes  22   60
```

```
(94 +60)/200  # classified Correctly

## [1] 0.77
```

Next Class

Trees are simple to understand, but not as competitive with other supervised learning approaches for prediction/classification.
Ways to improving Trees through multiple trees in some ensemble:

- Bagging
- Random Forests
- Boosting
- BART (Bayesian Additive Regression Trees)

Combining trees will yield improved prediction accuracy, but with loss of interpretability.