MIS 64036: Business Analytics

Lecture VIII

Rouzbeh Razavi, PhD

Agenda

- Tree-based Methods
- R implementation

Agenda

- Tree-based Methods
- R implementation

Tree-based Methods

- Here we describe *tree-based* methods for regression and classification.
- These involve *stratifying* or *segmenting* the predictor space into a number of simple regions.
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as *decision-tree* methods.

The Basics of Decision Trees

- Decision trees can be applied to both regression and classification problems.
- We first consider regression problems, and then move on to classification.

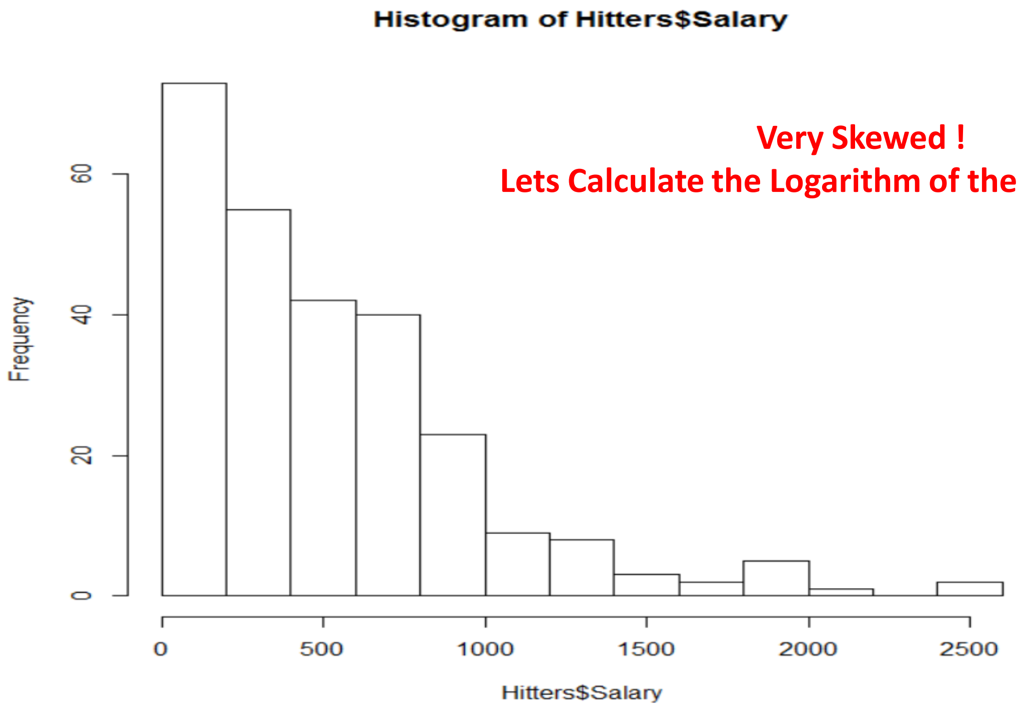
The Basics of Decision Trees

- Decision trees can be applied to both regression and classification problems.
- We first consider regression problems, and then move on to classification.

Hitters Dataset: Salary of Baseball Players

```
> library(ISLR)
> summary(Hitters)
     AtBat             Hits            HmRun             Runs              RBI              Walks       
 Min.   : 16.0   Min.   :  1   Min.   : 0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.00  
 1st Qu.:255.2   1st Qu.: 64   1st Qu.: 4.00   1st Qu.: 30.25   1st Qu.: 28.00   1st Qu.: 22.00  
 Median :379.5   Median : 96   Median : 8.00   Median : 48.00   Median : 44.00   Median : 35.00  
 Mean   :380.9   Mean   :101   Mean   :10.77   Mean   : 50.91   Mean   : 48.03   Mean   : 38.74  
 3rd Qu.:512.0   3rd Qu.:137   3rd Qu.:16.00   3rd Qu.: 69.00   3rd Qu.: 64.75   3rd Qu.: 53.00  
 Max.   :687.0   Max.   :238   Max.   :40.00   Max.   :130.00   Max.   :121.00   Max.   :105.00  

     Years            CAtBat           CHits            CHmRun           CRuns             CRBI        
 Min.   : 1.000   Min.   :   19.0   Min.   :   4.0   Min.   :  0.00   Min.   :   1.0   Min.   :   0.00  
 1st Qu.: 4.000   1st Qu.:  816.8   1st Qu.: 209.0   1st Qu.: 14.00   1st Qu.: 100.2   1st Qu.:  88.75  
 Median : 6.000   Median : 1928.0   Median : 508.0   Median : 37.50   Median : 247.0   Median : 220.50  
 Mean   : 7.444   Mean   : 2648.7   Mean   : 717.6   Mean   : 69.49   Mean   : 358.8   Mean   : 330.12  
 3rd Qu.:11.000   3rd Qu.: 3924.2   3rd Qu.:1059.2   3rd Qu.: 90.00   3rd Qu.: 526.2   3rd Qu.: 426.25  
 Max.   :24.000   Max.   :14053.0   Max.   :4256.0   Max.   :548.00   Max.   :2165.0   Max.   :1659.00  

     CWalks          League  Division    PutOuts          Assists          Errors          Salary      
 Min.   :   0.00   A:175   E:157    Min.   :   0.0   Min.   :  0.0   Min.   : 0.00   Min.   :  67.5  
 1st Qu.:  67.25   N:147   W:165    1st Qu.: 109.2   1st Qu.:  7.0   1st Qu.: 3.00   1st Qu.: 190.0  
 Median : 170.50                    Median : 212.0   Median : 39.5   Median : 6.00   Median : 425.0  
 Mean   : 260.24                    Mean   : 288.9   Mean   :106.9   Mean   : 8.04   Mean   : 535.9  
 3rd Qu.: 339.25                    3rd Qu.: 325.0   3rd Qu.:166.0   3rd Qu.:11.00   3rd Qu.: 750.0  
 Max.   :1566.00                    Max.   :1378.0   Max.   :492.0   Max.   :32.00   Max.   :2460.0  
                                                                                     NA's   :59      
```

Hitters Dataset: Salary of Baseball Players



Very Skewed !

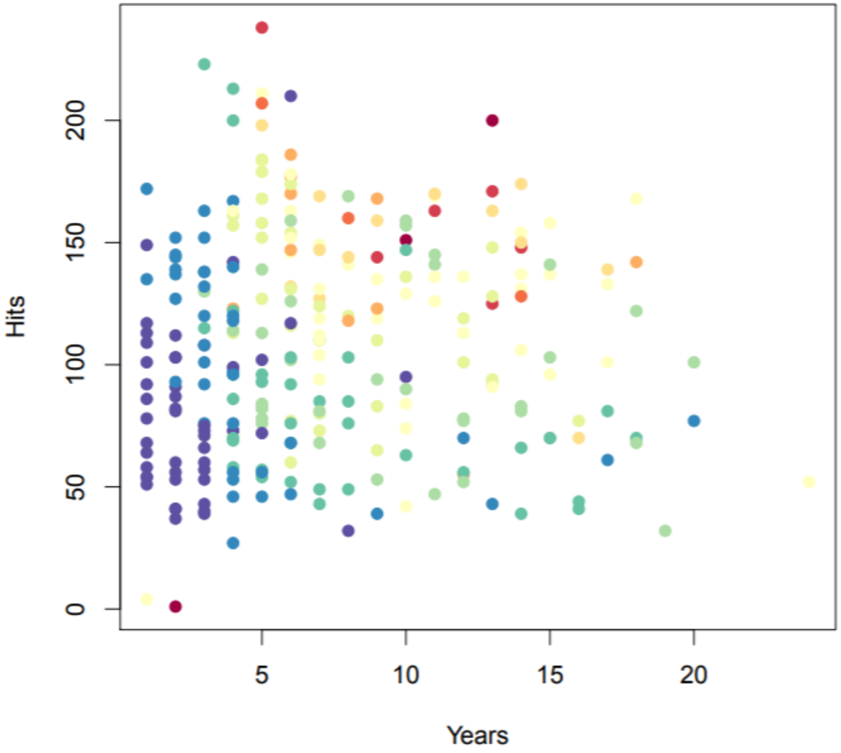
Lets Calculate the Logarithm of the Salary Values

Hitters Dataset: Salary of Baseball Players

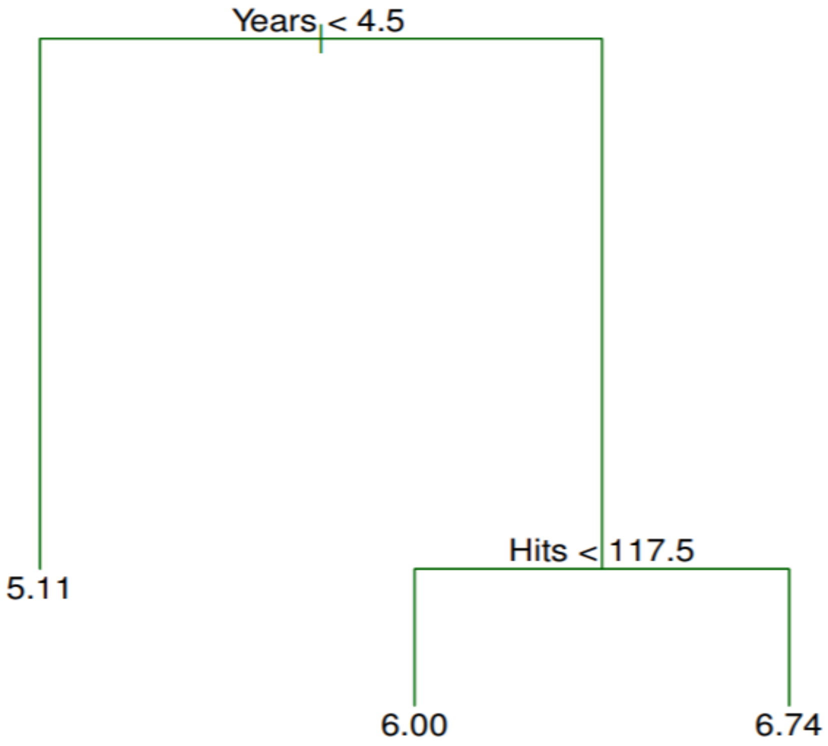
hist(log(Hitters$Salary))

Baseball salary data: how would you stratify it?

Salary is color-coded from low (blue, green) to high (yellow,red)



Decision tree for these data



Details of previous figure

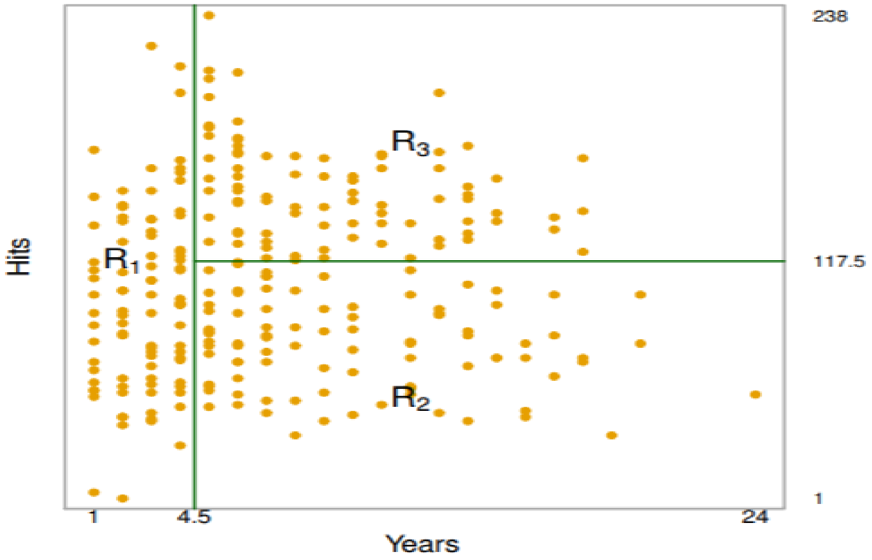
- For the Hitters data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year.

- At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to Years<4.5, and the right-hand branch corresponds to Years>=4.5.

- The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.

Results

- Overall, the tree stratifies or segments the players into three regions of predictor space: R_1 ={X | `Years`< 4.5}, R_2 ={X | `Years`>=4.5, `Hits`<117.5}, and R_3 ={X | `Years`>=4.5, `Hits`>=117.5}.



Terminology for Trees

- In keeping with the *tree* analogy, the regions R_1, R_2, and R_3 are known as *terminal nodes*
- Decision trees are typically drawn *upside down*, in the sense that the leaves are at the bottom of the tree.
- The points along the tree where the predictor space is split are referred to as *internal nodes*
- In the hitters tree, the two internal nodes are indicated by the text `Years`<4.5 and `Hits`<117.5.

Interpretation of Results

- `Years` is the most important factor in determining `Salary`, and players with less experience earn lower salaries than more experienced players.
- Given that a player is less experienced, the number of `Hits` that he made in the previous year seems to play little role in his `Salary`.
- But among players who have been in the major leagues for five or more years, the number of `Hits` made in the previous year does affect `Salary`, and players who made more `Hits` last year tend to have higher salaries.
- Surely an over-simplification, but compared to a regression model, it is easy to display, interpret and explain

Details of the tree-building process

1. We divide the predictor space — that is, the set of possible values for $X_1, X_2, \ldots, X_p$ — into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

More details of the tree-building process

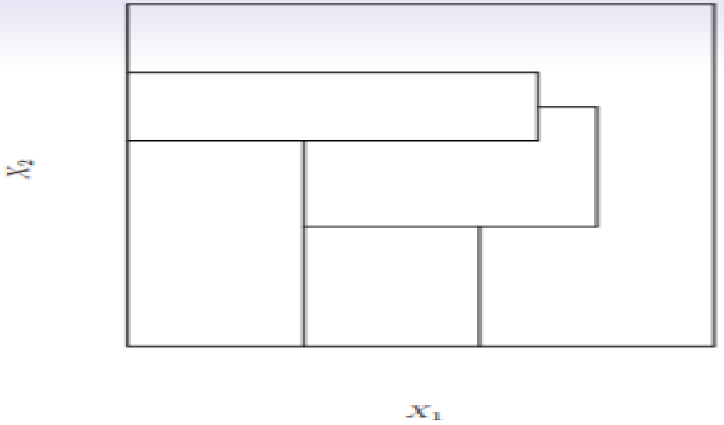
- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or *boxes*, for simplicity and for ease of interpretation of the resulting predictive model.
- The goal is to find boxes $R_1, \ldots, R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

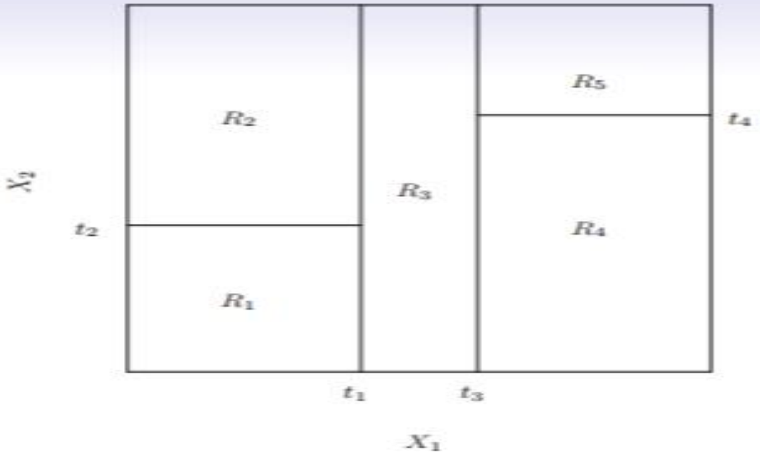
where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

Predictions

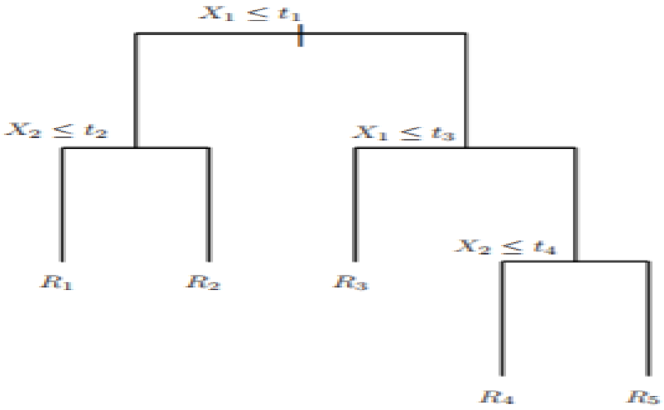
- We predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.
- A five-region example of this approach is shown in the next slide.



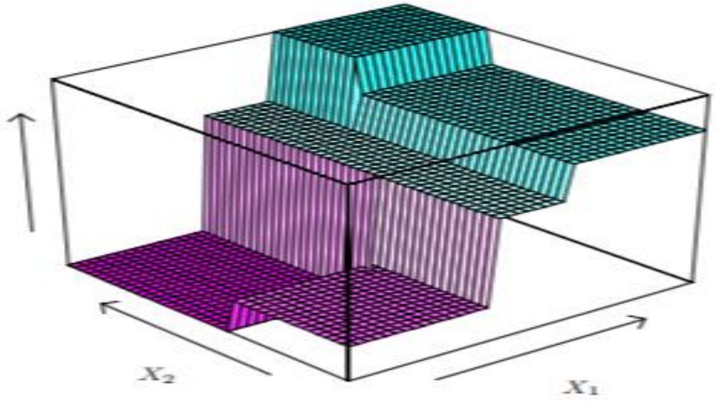
X_2
X_1



R_1
R_2
R_3
R_4
R_5
t_1
t_2
t_3
t_4
X_1
X_2



$X_1 \le t_1$
$X_2 \le t_2$
$X_1 \le t_3$
$X_2 \le t_4$
R_1
R_2
R_3
R_4
R_5



X_2
X_1

Details of previous figure

Top Left: A partition of two-dimensional feature space that could not result from recursive binary splitting.

Top Right: The output of recursive binary splitting on a two-dimensional example.

Bottom Left: A tree corresponding to the partition in the top right panel.

Bottom Right: A perspective plot of the prediction surface corresponding to that tree.

Classification Trees

- Very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one.
- For a classification tree, we predict that each observation belongs to the *most commonly occurring class* of training observations in the region to which it belongs.

Agenda

- Tree-based Methods
- R implementation

R Implementation

- We use the ‘rpart’ library from R to implement Decisions Trees (both for classification and regression)

- The function rpart() has a parameter called **method**. If the method is set to ‘anova’ the model will do regression. If the method is set to ‘class’ the model will be a classifier. There is also an optional control parameter, **minsplit** with default value of 30, which says hominy observation we should have at least at each node before attempting to split it further.

- Install the library using (make sure you have internet connectivity)

install.packages('rpart')

R Implementation

- Additional functions:

print(*Model*)	print results
summary(*Model*)	detailed results
plot(*Model*)	plot decision tree
text(*Model*)	label the decision tree plot

where 'Model' is the name of the rpart model.

Next, we will try to use decision trees for the earlier problems

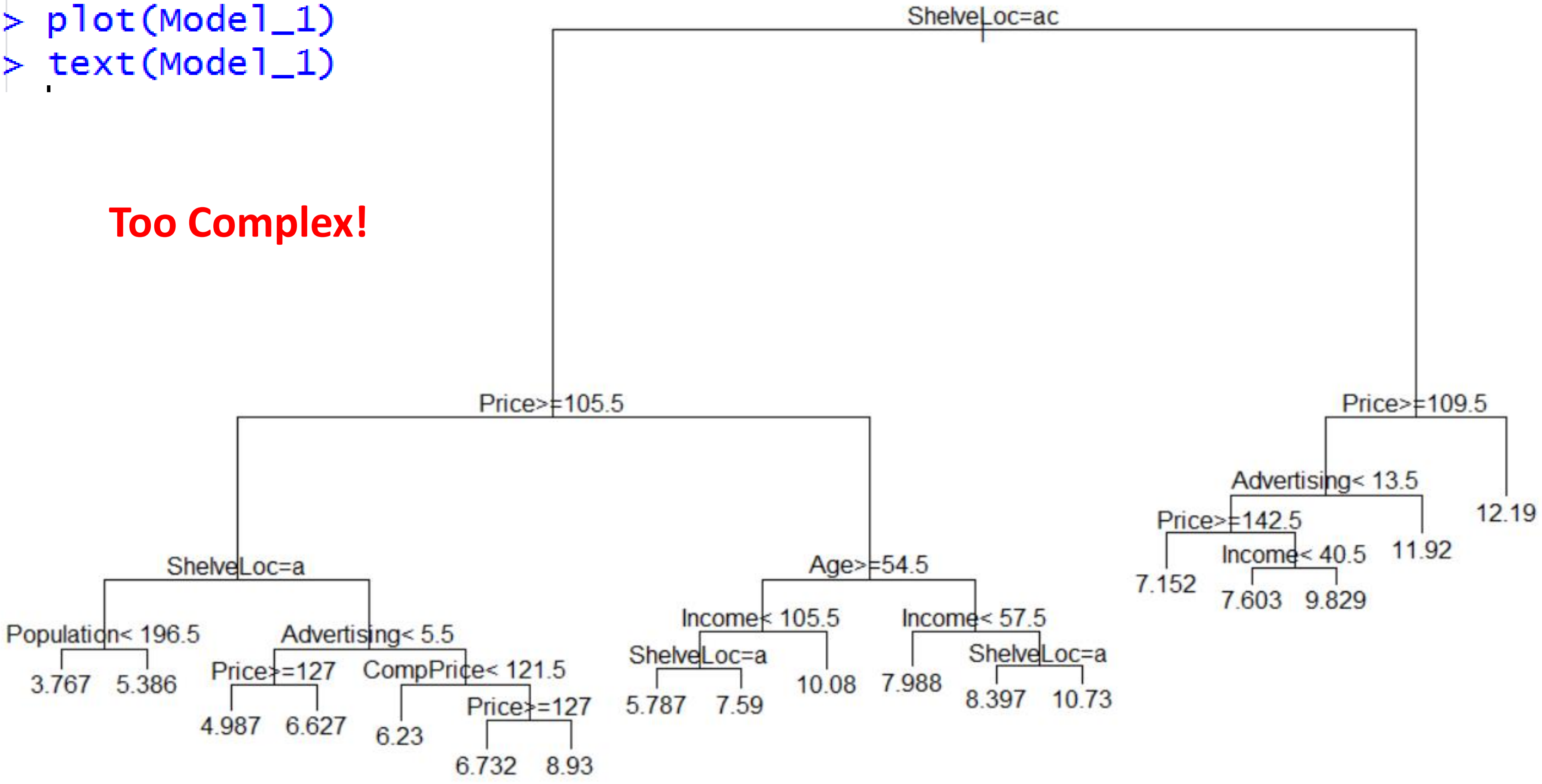
Predicting Sales of Baby Car Seats

```
library(ISLR)  # install.packages('ISLR') if you had errors
MyData<-Carseats[,1:8]
str(MyData) # shows which variables are factor or numerical
Model_1=rpart (Sales~.,data=MyData, method='anova')
 summary(Model_1)
```

```
> Model_1=rpart (Sales~.,data=MyData, method='anova')
> summary(Model_1)
Call:
rpart(formula = Sales ~ ., data = MyData, method = "anova")
  n= 400

           CP nsplit rel error    xerror       xstd
1  0.25051039      0 1.0000000 1.0083138 0.06974448
2  0.10507256      1 0.7494896 0.7563778 0.05127933
3  0.05112059      2 0.6444171 0.6692267 0.04431245
4  0.04567126      3 0.5932965 0.6469999 0.04330420
5  0.03359237      4 0.5476252 0.6021631 0.04173470
6  0.02406279      5 0.5140328 0.5833136 0.04013658
7  0.02394780      6 0.4899700 0.5848473 0.03962839
8  0.02216327      7 0.4660222 0.5853688 0.03965156
9  0.01604252      8 0.4438590 0.5762976 0.03938374
10 0.01402704      9 0.4278165 0.5571913 0.03667444
11 0.01314537     11 0.3997624 0.5549471 0.03889821
12 0.01271091     12 0.3866170 0.5579623 0.03966450
13 0.01214708     13 0.3739061 0.5555587 0.03974475
14 0.01188778     14 0.3617590 0.5541645 0.03952361
15 0.01077845     15 0.3498712 0.5508622 0.03858897
16 0.01050614     16 0.3390928 0.5554305 0.03876887
17 0.01000000     17 0.3285866 0.5583197 0.03871043

Variable importance
  ShelveLoc       Price   CompPrice Advertising      Income         Age  Population
         40          26           9           8           7           6           4
```

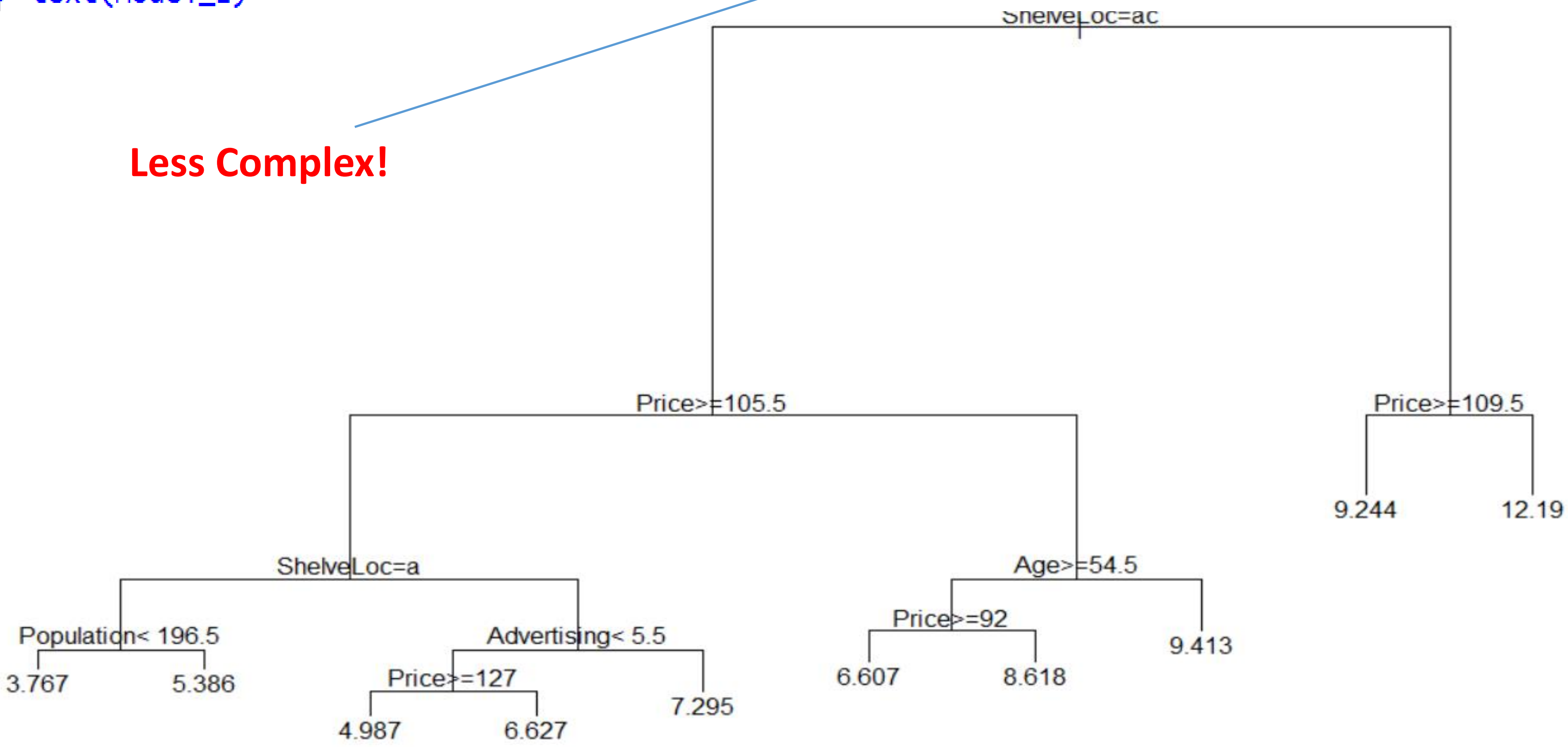
```
> plot(Model_1)
> text(Model_1)
```

Too Complex!



```
> Model_2=rpart (Sales~.,data=MyData, method='anova',control = rpart.control(minsplit = 60))
> plot(Model_2)
> text(Model_2)
```

Less Complex!



```
> summary(Model_2)
Call:
rpart(formula = Sales ~ ., data = MyData, method = "anova", control = rpart.control(
  n= 400

           CP nsplit rel error    xerror       xstd
1  0.25051039      0 1.0000000 1.0085338 0.06966971
2  0.10507256      1 0.7494896 0.7601247 0.05185659
3  0.05112059      2 0.6444171 0.6585943 0.04460677
4  0.04567126      3 0.5932965 0.6652920 0.04486063
5  0.03359237      4 0.5476252 0.6136164 0.04136162
6  0.02216327      5 0.5140328 0.5816088 0.04065888
7  0.01956091      6 0.4918696 0.5867304 0.03845265
8  0.01604252      7 0.4723087 0.5802242 0.03872478
9  0.01214708      8 0.4562661 0.5691229 0.03773806
10 0.01000000      9 0.4441191 0.5708498 0.03770605
```

Don’t worry about these! We don’t cover them in this course

Variable importance. The sum will be 100%

```
Variable importance
  ShelveLoc       Price   CompPrice         Age Advertising  Population      Income
         45          30           8           6           5           5           1
```

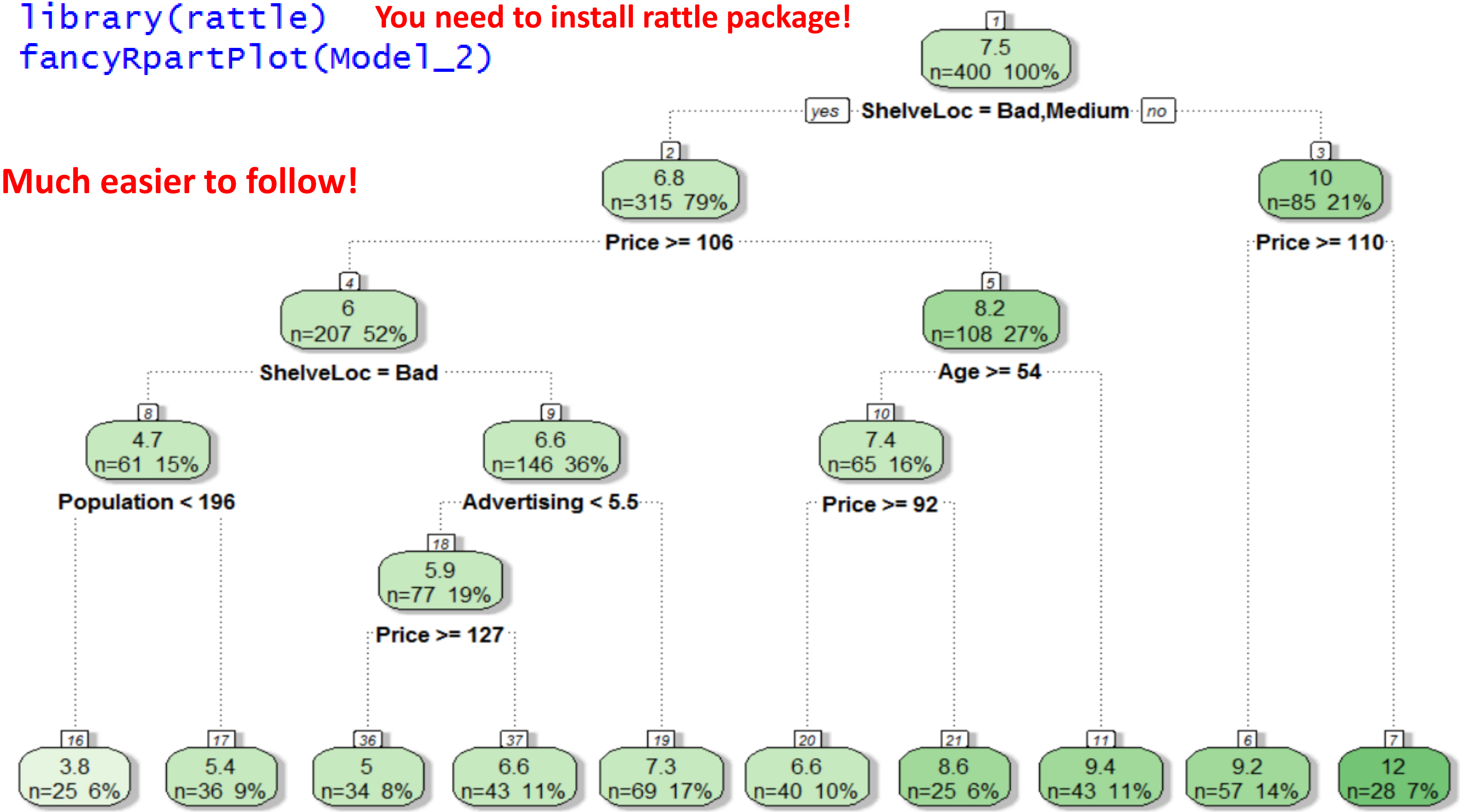
Decision tree Rules! Ugly and difficult to follow!

```
> print(Model_2)
n= 400

node), split, n, deviance, yval
      * denotes terminal node

 1) root 400 3182.27500  7.496325
   2) ShelveLoc=Bad,Medium 315 1859.56000  6.762984
     4) Price>=105.5 207  956.57240  6.018792
       8) ShelveLoc=Bad 61  240.81970  4.722459
        16) Population< 196.5 25   88.22930  3.767200 *
        17) Population>=196.5 36  113.93510  5.385833 *
       9) ShelveLoc=Medium 146  570.41420  6.560411
        18) Advertising< 5.5 77  280.11340  5.902468
          36) Price>=127 34  133.53970  4.986765 *
          37) Price< 127 43   95.52198  6.626512 *
        19) Advertising>=5.5 69  219.77110  7.294638 *
     5) Price< 105.5 108  568.61750  8.189352
      10) Age>=54.5 65  303.05690  7.380154
        20) Price>=92 40  128.69030  6.606500 *
        21) Price< 92 25  112.11840  8.618000 *
      11) Age< 54.5 43  158.66040  9.412558 *
   3) ShelveLoc=Good 85  525.52220 10.214000
     6) Price>=109.5 57  277.26520  9.244386 *
     7) Price< 109.5 28   85.57727 12.187860 *
>
```

```
> library(rattle)
> fancyRpartPlot(Model_2)
```

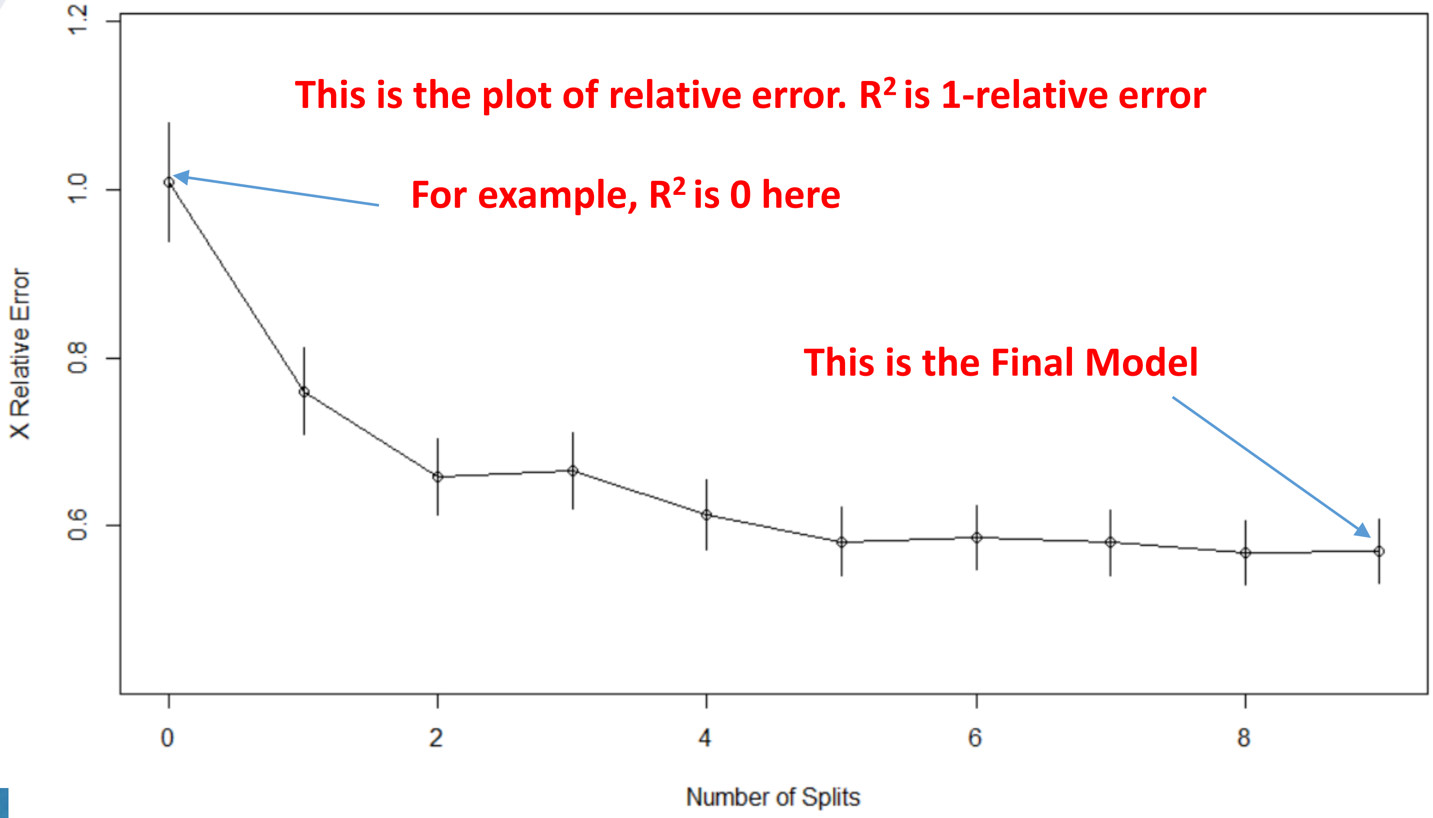
You need to install rattle package!

Much easier to follow!



Where is our beloved R^2 ?

```
> rsq.rpart(Model_2)
```



Where is our beloved R^2?

```
> rsq.rpart(Model_2)

Regression tree:
rpart(formula = Sales ~ ., data = MyData, method

Variables actually used in tree construction:
[1] Advertising Age         Population  Price

Root node error: 3182.3/400 = 7.9557

n= 400

         CP nsplit rel error  xerror     xstd
1  0.250510      0   1.00000 1.00853 0.069670
2  0.105073      1   0.74949 0.76012 0.051857
3  0.051121      2   0.64442 0.65859 0.044607
4  0.045671      3   0.59330 0.66529 0.044861
5  0.033592      4   0.54763 0.61362 0.041362
6  0.022163      5   0.51403 0.58161 0.040659
7  0.019561      6   0.49187 0.58673 0.038453
8  0.016043      7   0.47231 0.58022 0.038725
9  0.012147      8   0.45627 0.56912 0.037738
10 0.010000      9   0.44412 0.57085 0.037706
```

Don't worry bout these two columns

For Final Model:

R^2 = 1-0.444= 0.556

Or 55.6%


ANY
QUESTIONS
?