BDCT - Lecture 5

Decision trees using R

This tutorial will be an introduction to construct decision trees in R. We will follow series of commands that constitute the program, that would construct the tree. It builds on the same data set used in the regression programming exercise – pregnancy. We will also touch upon using packages in R.

Lecture 5 - Pregnancy example revisited

We will revisit the pregnancy example that was part of the exercise sheet using R. We will use the same set of features to construct a tree. It builds on a R package *rpart* for the purpose.

The question is to predict if the customer is pregnant or not that would involve a series of nodes where decisions are made. We go over each command in the program, and use the following steps while programming:

- Construct a tree based on the input data.
- Display the splits from the tree.
- Visualize the results of the tree.

For your reference, the entire program is also given at the end of this document. After each chunk of code, you also see the output that is produced on running them.

Loading Packages in R

R, by default, comes with many of the standard packages. Packages are collection of functions in R that are stored in what is referred to a library. There are many other packages available that would need to be initially downloaded and installed in the machine through some simple commands. Once they are installed, they would need to be loaded into the session to use it. These packages increase the statistical capacity of R, and also is an active part of the open source programming community through CRAN.

To load the packages to create a tree with our dataset, we first load the package *rpart*, and *xlsx*. If the packages are not currently installed in the machine, run the following code. Note that we used *xlsx* earlier to load an Excel file in to R.

```r
install.packages("xlsx")
install.packages("rpart")
```

If the packages are already installed, then load them in R using. Note, even if you have installed the package earlier, it would need to be loaded in order to use its functions.

```r
library(xlsx) #Loading the package needed to read Excel file
```

```
## Loading required package: rJava
## Loading required package: xlsxjars
```

```r
library(rpart) #Loading the package needed to construct and prune trees
```

```
## Warning: package 'rpart' was built under R version 3.1.2
```

Reading the Excel file

We read the Excel data that was provided into the data frame *lecture5data* . The variable name *DIRNAME* stores the directory, and *XLFILENAME* contains the file name in which the excel file is on my computer. The variable *FILENAME* stores the name of the absolute file name.

```
DIRNAME <- "/Users/rvijayaraghavan/Desktop/rajesh/Personal/Teaching/DataDrivenSrikantClass/"
XLFILENAME <- "03 Assignment Workbook - Regression_10-6.xlsx"
FILENAME <- paste0(DIRNAME,XLFILENAME)
lecture5data <- read.xlsx(FILENAME, sheetName="Training Data")
```

Next, we create the dummy variables as we did in lecture 3:

```
lecture5data$Male <- ifelse(lecture5data$Implied.Gender == "M", 1, 0)
lecture5data$Female <- ifelse(lecture5data$Implied.Gender == "F", 1, 0)
lecture5data$Home <- ifelse(lecture5data$Home.Apt..PO.Box == "H", 1, 0)
lecture5data$Apt <- ifelse(lecture5data$Home.Apt..PO.Box == "A", 1,0)
```

Constructing the Tree

To contruct the tree, the R code is similar to that of regression. The command used is *rpart()*, with the outcome variable *PREGNANT* along with the same set of features used in regression. The argument *method = "class"* in the rpart function specifies that we are interested in constructing a classification tree. The *rpart* uses the recursive partitioning algorithm while parsing the tree. The output is generated in the variable *fit*.

For simplicity, we will first begin with an example using just four features that could potential predict if some one is pregnant or not.

```
fit <- rpart(PREGNANT ~ Birth.Control + Feminine.Hygiene  +  Folic.Acid + Prenatal.Vitamins ,
    method="class", data=lecture5data)
```

R at this point stores the tree that was created.

Displaying the results

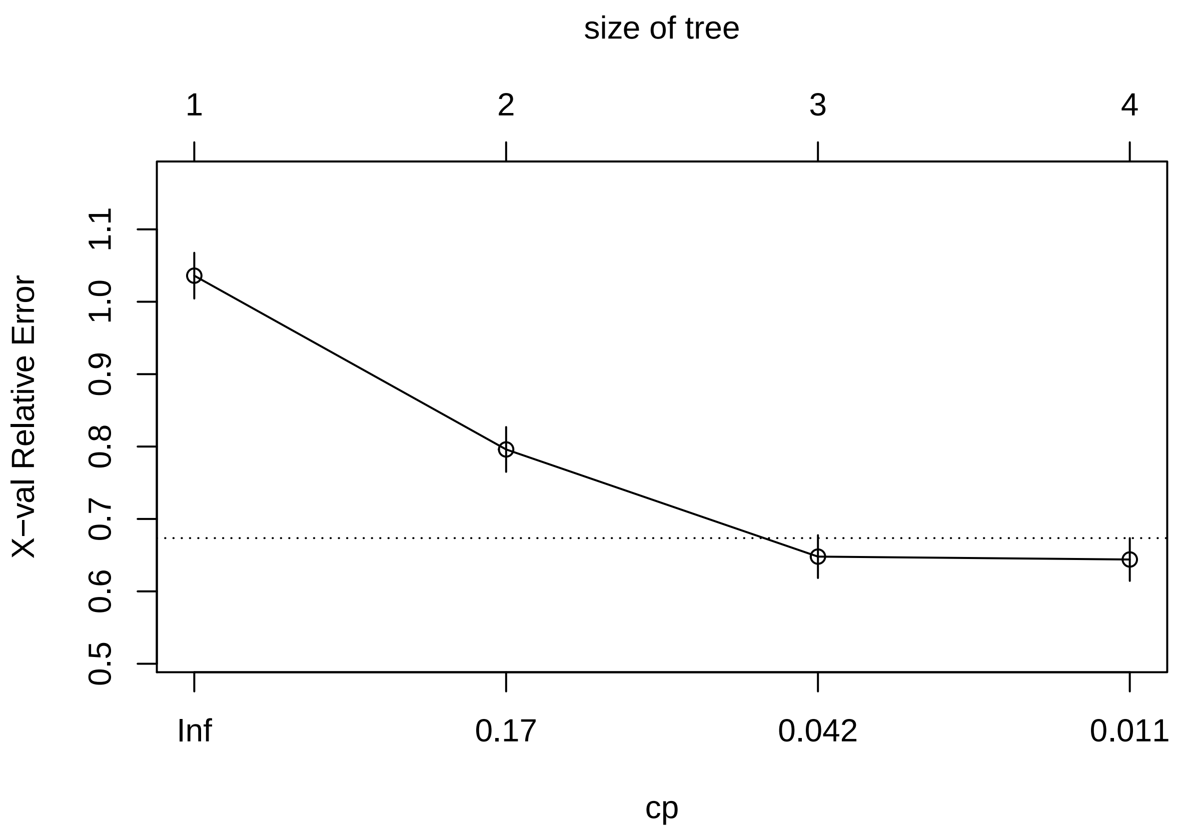
Once the above command is run, the output is stored in the variable *fit*. The following commands are used to display the results, and then to visualize the results from the tree generated using *rpart*. The cross validation results help in choosing the tree size that minimizes the cross validated error – that is displayed in the *xerror* column.

```
printcp(fit) # display the results
```

```
##
## Classification tree:
## rpart(formula = PREGNANT ~ Birth.Control + Feminine.Hygiene +
##     Folic.Acid + Prenatal.Vitamins, data = lecture5data, method = "class")
##
## Variables actually used in tree construction:
## [1] Birth.Control     Folic.Acid        Prenatal.Vitamins
##
## Root node error: 500/1000 = 0.5
```

```
##
## n= 1000
##
##      CP nsplit rel error xerror  xstd
## 1 0.204      0      1.00   1.04 0.032
## 2 0.148      1      0.80   0.80 0.031
## 3 0.012      2      0.65   0.65 0.030
## 4 0.010      3      0.64   0.64 0.030
```

```r
plotcp(fit) # visualize cross-validation results
```



The command *summary* provides the summary of the tree along with the splits.

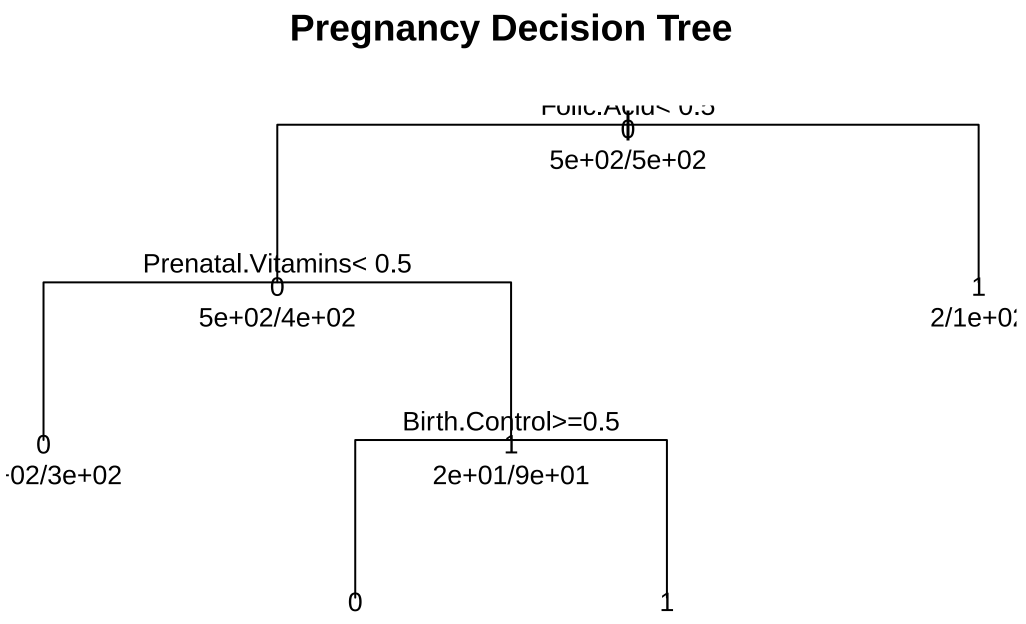
```r
summary(fit) # detailed summary of splits
```

```
## Call:
## rpart(formula = PREGNANT ~ Birth.Control + Feminine.Hygiene +
##     Folic.Acid + Prenatal.Vitamins, data = lecture5data, method = "class")
##   n= 1000
##
##      CP nsplit rel error xerror    xstd
## 1 0.204      0     1.000  1.036 0.03160
## 2 0.148      1     0.796  0.796 0.03096
## 3 0.012      2     0.648  0.648 0.02960
## 4 0.010      3     0.636  0.644 0.02955
##
## Variable importance
##        Folic.Acid Prenatal.Vitamins     Birth.Control
##                53                38                 9
##
## Node number 1: 1000 observations,    complexity param=0.204
##   predicted class=0  expected loss=0.5  P(node) =1
##     class counts:   500   500
##    probabilities: 0.500 0.500
##   left son=2 (894 obs) right son=3 (106 obs)
##   Primary splits:
##       Folic.Acid        < 0.5 to the left,  improve=54.89, (0 missing)
##       Birth.Control     < 0.5 to the right, improve=43.21, (0 missing)
##       Prenatal.Vitamins < 0.5 to the left,  improve=41.28, (0 missing)
##       Feminine.Hygiene  < 0.5 to the right, improve=29.83, (0 missing)
##
## Node number 2: 894 observations,    complexity param=0.148
##   predicted class=0  expected loss=0.443  P(node) =0.894
##     class counts:   498   396
##    probabilities: 0.557 0.443
##   left son=4 (788 obs) right son=5 (106 obs)
##   Primary splits:
##       Prenatal.Vitamins < 0.5 to the left,  improve=39.67, (0 missing)
##       Birth.Control     < 0.5 to the right, improve=36.60, (0 missing)
##       Feminine.Hygiene  < 0.5 to the right, improve=25.83, (0 missing)
##
## Node number 3: 106 observations
##   predicted class=1  expected loss=0.01887  P(node) =0.106
##     class counts:     2   104
##    probabilities: 0.019 0.981
##
## Node number 4: 788 observations
##   predicted class=0  expected loss=0.3883  P(node) =0.788
##     class counts:   482   306
##    probabilities: 0.612 0.388
##
## Node number 5: 106 observations,    complexity param=0.012
##   predicted class=1  expected loss=0.1509  P(node) =0.106
##     class counts:    16    90
##    probabilities: 0.151 0.849
##   left son=10 (10 obs) right son=11 (96 obs)
##   Primary splits:
##       Birth.Control    < 0.5 to the right, improve=9.303, (0 missing)
##       Feminine.Hygiene < 0.5 to the right, improve=4.450, (0 missing)
##
## Node number 10: 10 observations
##   predicted class=0  expected loss=0.2  P(node) =0.01
##     class counts:     8     2
##    probabilities: 0.800 0.200
##
## Node number 11: 96 observations
##   predicted class=1  expected loss=0.08333  P(node) =0.096
##     class counts:     8    88
##    probabilities: 0.083 0.917
```

The above commands show number of statistic in understanding how the tree was constructed, and about their splits. They also show ratios that could be used to calculate the probabilities while estimating their entropies.

As the next step, to visualize the tree, we use the *plot()* function. This takes the output *fit* that was run in the previous step and plots the tree that was created.

```
# plot tree
plot(fit, uniform=TRUE, main="Pregnancy Decision Tree")
text(fit, use.n=TRUE, all=TRUE, cex=.8)
```



In the first step, we use four features to construct a tree and understand which features have the highest entropy. We will now proceed to include all the features from the dataset.

Analysis with all features

Now revisit the example with all the features. We will now construct a tree to include all the features, including the four that we used in the prior step.

```
fit <- rpart(PREGNANT ~ Female + Male + Home + Apt + Pregnancy.Test + Birth.Control
                + Feminine.Hygiene  +  Folic.Acid + Prenatal.Vitamins + Prenatal.Yoga +
                  Body.Pillow + Ginger.Ale + Sea.Bands + Stopped.buying.ciggies +
                  Cigarettes + Smoking.Cessation + Stopped.buying.wine +
                  Wine + Maternity.Clothes,
    method="class", data=lecture5data)
```

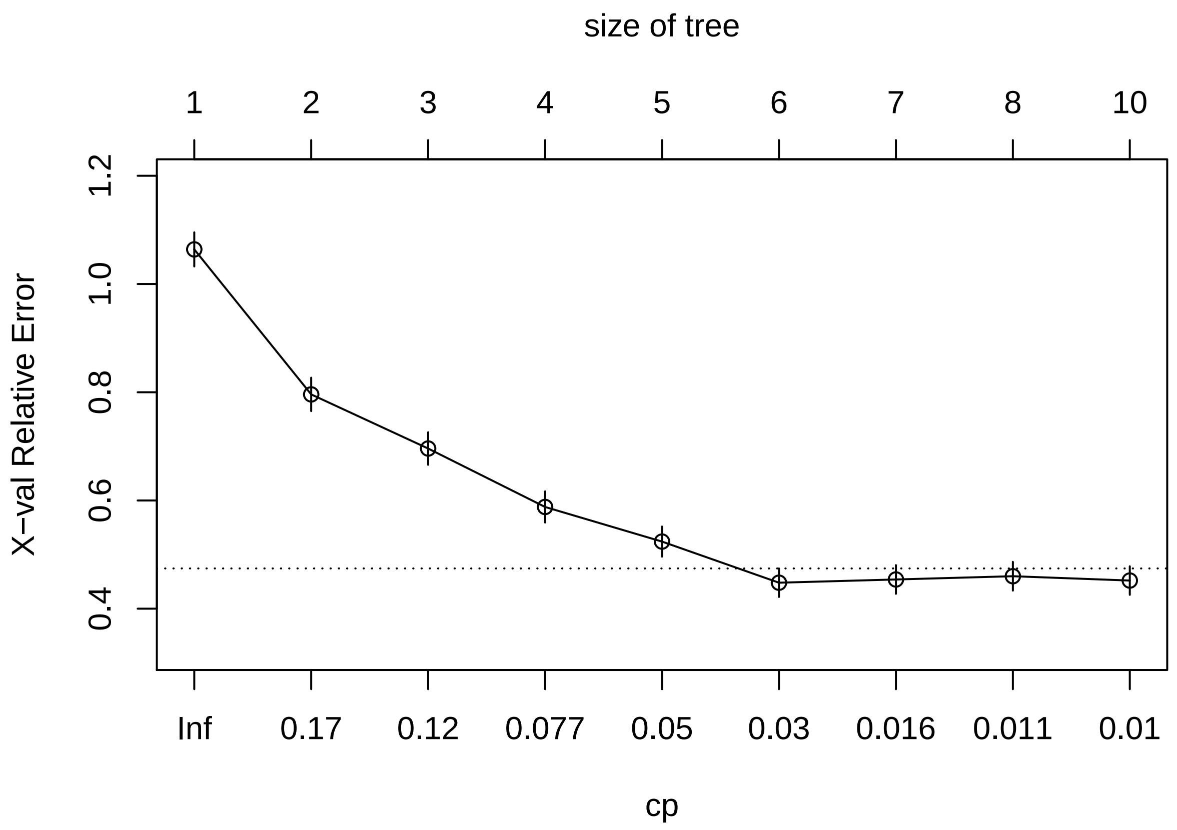
Displaying the results

To display the results from the tree model created. The cross validation results help in choosing the tree size that minimizes the cross validated error – that is displayed in the *xerror* column.

```r
printcp(fit) # display the results
```

```
##
## Classification tree:
## rpart(formula = PREGNANT ~ Female + Male + Home + Apt + Pregnancy.Test +
##     Birth.Control + Feminine.Hygiene + Folic.Acid + Prenatal.Vitamins +
##     Prenatal.Yoga + Body.Pillow + Ginger.Ale + Sea.Bands + Stopped.buying.ciggies +
##     Cigarettes + Smoking.Cessation + Stopped.buying.wine + Wine +
##     Maternity.Clothes, data = lecture5data, method = "class")
##
## Variables actually used in tree construction:
## [1] Birth.Control          Feminine.Hygiene       Folic.Acid
## [4] Ginger.Ale             Maternity.Clothes      Pregnancy.Test
## [7] Prenatal.Vitamins      Stopped.buying.ciggies Stopped.buying.wine
##
## Root node error: 500/1000 = 0.5
##
## n= 1000
##
##      CP nsplit rel error xerror  xstd
## 1 0.204      0      1.00   1.06 0.032
## 2 0.148      1      0.80   0.80 0.031
## 3 0.098      2      0.65   0.70 0.030
## 4 0.060      3      0.55   0.59 0.029
## 5 0.042      4      0.49   0.52 0.028
## 6 0.022      5      0.45   0.45 0.026
## 7 0.012      6      0.43   0.45 0.026
## 8 0.011      7      0.41   0.46 0.027
## 9 0.010      9      0.39   0.45 0.026
```

```r
plotcp(fit) # visualize cross-validation results
```



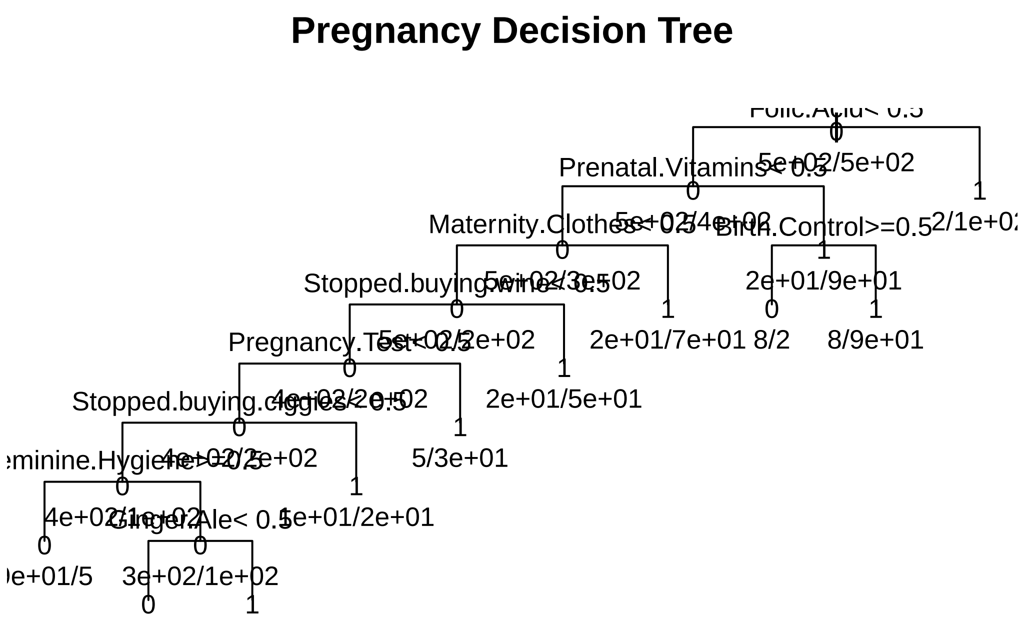
The command *summary* provides the summary of the tree along with the splits.

```
summary(fit) # detailed summary of splits
```

```
## Call:
## rpart(formula = PREGNANT ~ Female + Male + Home + Apt + Pregnancy.Test +
##     Birth.Control + Feminine.Hygiene + Folic.Acid + Prenatal.Vitamins +
##     Prenatal.Yoga + Body.Pillow + Ginger.Ale + Sea.Bands + Stopped.buying.ciggies +
##     Cigarettes + Smoking.Cessation + Stopped.buying.wine + Wine +
##     Maternity.Clothes, data = lecture5data, method = "class")
##   n= 1000
##
##      CP nsplit rel error xerror    xstd
## 1 0.204      0     1.000  1.064 0.03156
## 2 0.148      1     0.796  0.796 0.03096
## 3 0.098      2     0.648  0.696 0.03013
## 4 0.060      3     0.550  0.588 0.02881
## 5 0.042      4     0.490  0.524 0.02781
## 6 0.022      5     0.448  0.448 0.02637
## 7 0.012      6     0.426  0.454 0.02649
## 8 0.011      7     0.414  0.460 0.02662
## 9 0.010      9     0.392  0.452 0.02645
##
## Variable importance
##             Folic.Acid      Prenatal.Vitamins      Maternity.Clothes
##                     27                     19                     15
##    Stopped.buying.wine         Pregnancy.Test Stopped.buying.ciggies
##                     11                      9                      6
##             Ginger.Ale          Birth.Control       Feminine.Hygiene
##                      5                      5                      4
##
## Node number 1: 1000 observations,    complexity param=0.204
##   predicted class=0  expected loss=0.5  P(node) =1
##     class counts:   500   500
##    probabilities: 0.500 0.500
##   left son=2 (894 obs) right son=3 (106 obs)
##   Primary splits:
##       Folic.Acid        < 0.5 to the left,  improve=54.89, (0 missing)
##       Birth.Control     < 0.5 to the right, improve=43.21, (0 missing)
##       Prenatal.Vitamins < 0.5 to the left,  improve=41.28, (0 missing)
##       Maternity.Clothes < 0.5 to the left,  improve=36.37, (0 missing)
##       Wine              < 0.5 to the right, improve=31.93, (0 missing)
##
## Node number 2: 894 observations,    complexity param=0.148
##   predicted class=0  expected loss=0.443  P(node) =0.894
##     class counts:   498   396
##    probabilities: 0.557 0.443
##   left son=4 (788 obs) right son=5 (106 obs)
##   Primary splits:
##       Prenatal.Vitamins   < 0.5 to the left,  improve=39.67, (0 missing)
##       Birth.Control       < 0.5 to the right, improve=36.60, (0 missing)
##       Maternity.Clothes   < 0.5 to the left,  improve=33.97, (0 missing)
##       Pregnancy.Test      < 0.5 to the left,  improve=28.23, (0 missing)
##       Stopped.buying.wine < 0.5 to the left,  improve=27.18, (0 missing)
##
## Node number 3: 106 observations
##   predicted class=1  expected loss=0.01887  P(node) =0.106
##     class counts:     2   104
##    probabilities: 0.019 0.981
##
## Node number 4: 788 observations,    complexity param=0.098
##   predicted class=0  expected loss=0.3883  P(node) =0.788
##     class counts:   482   306
##    probabilities: 0.612 0.388
##   left son=8 (699 obs) right son=9 (89 obs)
##   Primary splits:
##       Maternity.Clothes   < 0.5 to the left,  improve=30.05, (0 missing)
##       Birth.Control       < 0.5 to the right, improve=25.39, (0 missing)
##       Pregnancy.Test      < 0.5 to the left,  improve=23.92, (0 missing)
##       Stopped.buying.wine < 0.5 to the left,  improve=23.01, (0 missing)
##       Wine                < 0.5 to the right, improve=21.98, (0 missing)
##
## Node number 5: 106 observations,    complexity param=0.012
##   predicted class=1  expected loss=0.1509  P(node) =0.106
##     class counts:    16    90
##    probabilities: 0.151 0.849
##   left son=10 (10 obs) right son=11 (96 obs)
##   Primary splits:
##       Birth.Control          < 0.5 to the right, improve=9.303, (0 missing)
##       Feminine.Hygiene       < 0.5 to the right, improve=4.450, (0 missing)
##       Maternity.Clothes      < 0.5 to the left,  improve=1.055, (0 missing)
##       Home                   < 0.5 to the left,  improve=1.034, (0 missing)
##       Stopped.buying.ciggies < 0.5 to the left,  improve=0.988, (0 missing)
##
## Node number 8: 699 observations,    complexity param=0.06
##   predicted class=0  expected loss=0.3391  P(node) =0.699
##     class counts:   462   237
##    probabilities: 0.661 0.339
##   left son=16 (631 obs) right son=17 (68 obs)
##   Primary splits:
##       Stopped.buying.wine < 0.5 to the left,  improve=21.93, (0 missing)
##       Pregnancy.Test      < 0.5 to the left,  improve=20.69, (0 missing)
##       Birth.Control       < 0.5 to the right, improve=17.45, (0 missing)
##       Feminine.Hygiene    < 0.5 to the right, improve=17.36, (0 missing)
##       Wine                < 0.5 to the right, improve=17.36, (0 missing)
##
## Node number 9: 89 observations
##   predicted class=1  expected loss=0.2247  P(node) =0.089
##     class counts:    20    69
##    probabilities: 0.225 0.775
##
## Node number 10: 10 observations
##   predicted class=0  expected loss=0.2  P(node) =0.01
##     class counts:     8     2
##    probabilities: 0.800 0.200
##
## Node number 11: 96 observations
##   predicted class=1  expected loss=0.08333  P(node) =0.096
##     class counts:     8    88
##    probabilities: 0.083 0.917
##
## Node number 16: 631 observations,    complexity param=0.042
##   predicted class=0  expected loss=0.2979  P(node) =0.631
##     class counts:   443   188
##    probabilities: 0.702 0.298
##   left son=32 (600 obs) right son=33 (31 obs)
##   Primary splits:
##       Pregnancy.Test   < 0.5 to the left,  improve=19.07, (0 missing)
##       Birth.Control    < 0.5 to the right, improve=13.12, (0 missing)
##       Feminine.Hygiene < 0.5 to the right, improve=12.80, (0 missing)
##       Wine             < 0.5 to the right, improve=12.61, (0 missing)
##       Ginger.Ale       < 0.5 to the left,  improve=11.22, (0 missing)
##
## Node number 17: 68 observations
##   predicted class=1  expected loss=0.2794  P(node) =0.068
##     class counts:    19    49
##    probabilities: 0.279 0.721
##
## Node number 32: 600 observations,    complexity param=0.022
##   predicted class=0  expected loss=0.27  P(node) =0.6
##     class counts:   438   162
##    probabilities: 0.730 0.270
##   left son=64 (561 obs) right son=65 (39 obs)
##   Primary splits:
##       Stopped.buying.ciggies < 0.5 to the left,  improve=11.480, (0 missing)
##       Feminine.Hygiene       < 0.5 to the right, improve=10.770, (0 missing)
##       Wine                  < 0.5 to the right, improve=10.520, (0 missing)
##       Birth.Control         < 0.5 to the right, improve=10.480, (0 missing)
##       Ginger.Ale            < 0.5 to the left,  improve= 9.374, (0 missing)
##
## Node number 33: 31 observations
##   predicted class=1  expected loss=0.1613  P(node) =0.031
##     class counts:     5    26
##    probabilities: 0.161 0.839
##
## Node number 64: 561 observations,    complexity param=0.011
##   predicted class=0  expected loss=0.2442  P(node) =0.561
##     class counts:   424   137
##    probabilities: 0.756 0.244
##   left son=128 (99 obs) right son=129 (462 obs)
##   Primary splits:
##       Feminine.Hygiene  < 0.5 to the right, improve=9.021, (0 missing)
##       Wine              < 0.5 to the right, improve=8.569, (0 missing)
##       Ginger.Ale        < 0.5 to the left,  improve=8.024, (0 missing)
##       Birth.Control     < 0.5 to the right, improve=7.743, (0 missing)
##       Smoking.Cessation < 0.5 to the left,  improve=5.690, (0 missing)
##
## Node number 65: 39 observations
##   predicted class=1  expected loss=0.359  P(node) =0.039
##     class counts:    14    25
##    probabilities: 0.359 0.641
##
## Node number 128: 99 observations
##   predicted class=0  expected loss=0.05051  P(node) =0.099
##     class counts:    94     5
##    probabilities: 0.949 0.051
##
## Node number 129: 462 observations,    complexity param=0.011
##   predicted class=0  expected loss=0.2857  P(node) =0.462
##     class counts:   330   132
##    probabilities: 0.714 0.286
##   left son=258 (437 obs) right son=259 (25 obs)
##   Primary splits:
##       Ginger.Ale        < 0.5 to the left,  improve=9.970, (0 missing)
##       Birth.Control     < 0.5 to the right, improve=8.660, (0 missing)
##       Wine              < 0.5 to the right, improve=8.592, (0 missing)
##       Smoking.Cessation < 0.5 to the left,  improve=4.609, (0 missing)
##       Cigarettes        < 0.5 to the right, improve=4.561, (0 missing)
##
## Node number 258: 437 observations
##   predicted class=0  expected loss=0.2609  P(node) =0.437
##     class counts:   323   114
##    probabilities: 0.739 0.261
##
## Node number 259: 25 observations
##   predicted class=1  expected loss=0.28  P(node) =0.025
##     class counts:     7    18
##    probabilities: 0.280 0.720
```

To visualize the tree, it takes the output *fit* that was run in the previous step. It runs the *plot()* command with the *fit* parameter. Notice that the tree constructed with larger set is much different from what we generated with just the four features.

```r
# plot tree
plot(fit, uniform=TRUE, main="Pregnancy Decision Tree")
text(fit, use.n=TRUE, all=TRUE, cex=.8)
```



As you see, constructing a tree in R using recursive partioning algorithm is simple.

Entire code from above

```r
library(xlsx) #Loading the package needed to read Excel file
library(rpart) #Loading the package needed to construct and prune trees

DIRNAME <- "/Users/rvijayaraghavan/Desktop/rajesh/Personal/Teaching/DataDrivenSrikantClass/"
XLFILENAME <- "03 Assignment Workbook - Regression_10-6.xlsx"
FILENAME <- paste0(DIRNAME,XLFILENAME)
lecture5data <- read.xlsx(FILENAME, sheetName="Training Data")


lecture5data$Male <- ifelse(lecture5data$Implied.Gender == "M", 1, 0)
lecture5data$Female <- ifelse(lecture5data$Implied.Gender == "F", 1, 0)
lecture5data$Home <- ifelse(lecture5data$Home.Apt..PO.Box == "H", 1, 0)
lecture5data$Apt <- ifelse(lecture5data$Home.Apt..PO.Box == "A", 1,0)

fit <- rpart(PREGNANT ~ Birth.Control + Feminine.Hygiene  +  Folic.Acid + Prenatal.Vitamins ,
    method="class", data=lecture5data)

printcp(fit) # display the results
plotcp(fit) # visualize cross-validation results

summary(fit) # detailed summary of splits

plot(fit, uniform=TRUE, main="Pregnancy Decision Tree")
```

```r
text(fit, use.n=TRUE, all=TRUE, cex=.8)



fit <- rpart(PREGNANT ~ Female + Male + Home + Apt + Pregnancy.Test + Birth.Control
                + Feminine.Hygiene  +  Folic.Acid + Prenatal.Vitamins + Prenatal.Yoga +
                  Body.Pillow + Ginger.Ale + Sea.Bands + Stopped.buying.ciggies +
                  Cigarettes + Smoking.Cessation + Stopped.buying.wine +
                  Wine + Maternity.Clothes,
    method="class", data=lecture5data)


printcp(fit) # display the results
plotcp(fit) # visualize cross-validation results

summary(fit) # detailed summary of splits

plot(fit, uniform=TRUE, main="Pregnancy Decision Tree")
text(fit, use.n=TRUE, all=TRUE, cex=.8)
```