CUNY School of Professional Studies

SPS.CUNY.EDU

Lecture 11
2020 Spring Data-622
Decision Tree
Raman Kannan

Instructor Email Address: Raman.Kannan@sps.cuny.edu

Acknowledgements:
Generous support from IBM Power Systems Academic Initiative
IBM PSAI provides computing infrastructure for free

Script for Algorithms

Develop the Intuition
Understand the assumptions
Develop the mathematics
Run the algorithms
Learn to interpret the result/output
Predict using the model
Learn to determine the performance
Distinguish training/testing error
Differentiate between overfitting/underfitting
Techniques to improve performance

Intuition (Tree)

A tree is a hierarchical system. Root, stem, branches(internal decision nodes) and then the leaves.

In Classification we are trying to learn how to label never seen before data given some labeled data to learn.

At each juncture, we evaluate the attributes, and first find the attributes that provide the most information toward identifying the label.

We use that attribute to navigate the branch and reach the node. When a node results in all instances that belong to a single class we have reached a terminal/leaf node.

We want a short tree – we may choose a shorter tree than a tree with more splits to avoid over-fitting.

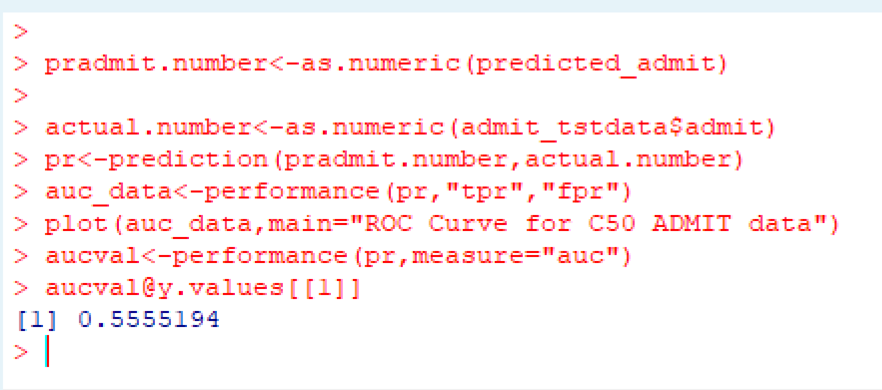
How can we capture the above in a procedure? We need a formal Mechanism to determine the attribute. Decision Tree use Entropy, Information Gain/OR Gini Index to identify such attributes.

Generate a Model (C50)

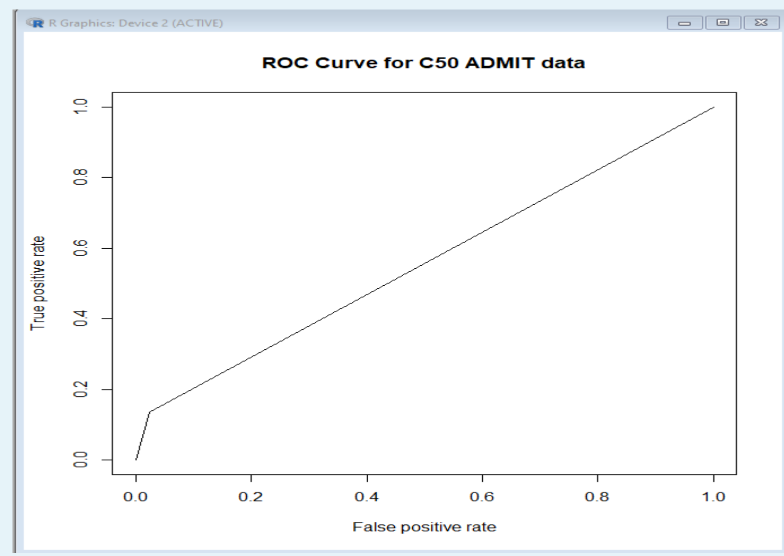
```
path<-"C:/Users/rk215/cuny/L11-tree/binary.csv"
admit_data<-read.csv(path,head=TRUE);
head(admit_data)

#make some columns factors
fad<-data.frame(as.factor(admit_data$admit),admit_data$gre,
admit_data$gpa,as.factor(admit_data$rank))
names(fad)<-names(admit_data)

# create test and training set
set.seed(43)
tstset<-sample(400,120,replace=FALSE) # 30% hold out test set
admit_trdata<-fad[-tstset,]
admit_tstdata<-fad[tstset,]

# generate model
C50_model<-C5.0(admit~.,data=admit_trdata)
```

Performance

```
predicted_admit<-predict(C50_model,admit_tstdata[,-1])
head(predicted_admit)

pradmit.number<-as.numeric(predicted_admit)

actual.number<-as.numeric(admit_tstdata$admit)
pr<-prediction(pradmit.number,actual.number)
auc_data<-performance(pr,"tpr","fpr")
plot(auc_data,main="ROC Curve for C50 ADMIT data")
aucval<-performance(pr,measure="auc")
aucval@y.values[[1]]
```



```
>
> pradmit.number<-as.numeric(predicted_admit)
>
> actual.number<-as.numeric(admit_tstdata$admit)
> pr<-prediction(pradmit.number,actual.number)
> auc_data<-performance(pr,"tpr","fpr")
> plot(auc_data,main="ROC Curve for C50 ADMIT data")
> aucval<-performance(pr,measure="auc")
> aucval@y.values[[1]]
[1] 0.5555194
>
```

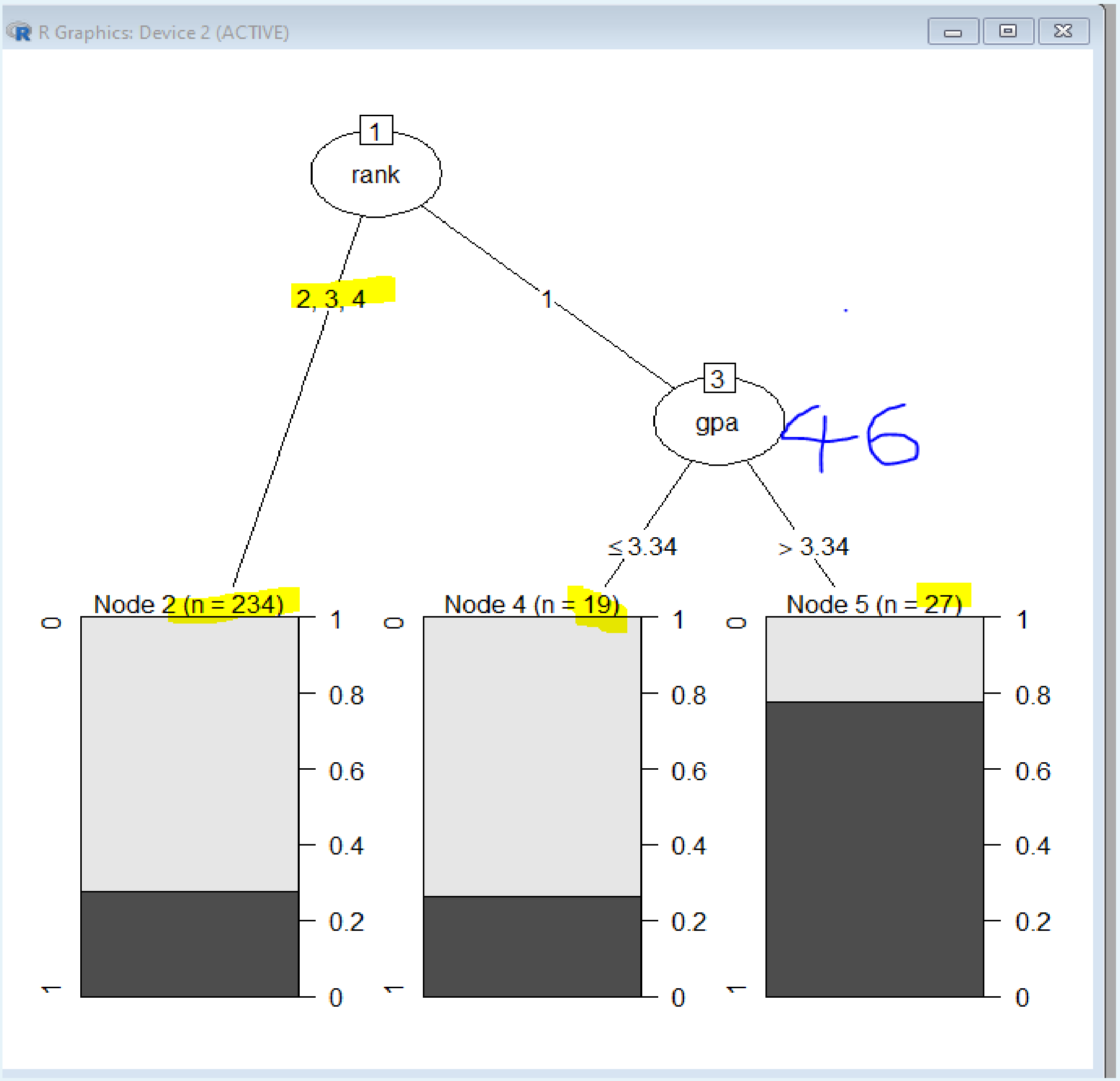


Visualizing the Tree

plot(C50_model)
admit_trdata[admit_trdata$gpa>3.34&admit_trdata$rank==1,]
dim(admit_trdata[admit_trdata$gpa>3.34&admit_trdata$rank==1,])
nrow(admit_trdata[admit_trdata$rank==1,])
table(admit_trdata$rank)
sum(table(admit_trdata$rank))-nrow(admit_trdata[admit_trdata$rank==1,])

```
> admit_trdata[admit_trdata$gpa>3.34&admit_trdata$rank==1,]
    admit gre  gpa rank
3       1 800 4.00    1
13      1 760 4.00    1
15      1 700 4.00    1
26      1 800 3.66    1
27      1 620 3.61    1
69      0 580 3.69    1
80      1 620 4.00    1
92      1 720 3.64    1
119     1 800 3.70    1
127     1 600 3.54    1
140     1 600 3.58    1
151     1 800 3.74    1
166     0 700 4.00    1
203     1 700 4.00    1
205     1 600 3.89    1
207     0 740 3.54    1
208     1 640 3.63    1
235     1 800 3.53    1
242     1 520 3.81    1
278     1 580 3.58    1
326     0 680 3.90    1
336     1 620 3.71    1
362     1 540 3.49    1
365     1 560 3.36    1
369     0 580 4.00    1
374     1 620 3.37    1
384     0 660 4.00    1
> dim(admit_trdata[admit_trdata$gpa>3.34&admit_trdata$rank==1,])
[1] 27  4
> nrow(admit_trdata[admit_trdata$rank==1,])
[1] 46
> table(admit_trdata$rank)

  1   2   3   4
 46 101  91  42
> sum(table(admit_trdata$rank))-nrow(admit_trdata[admit_trdata$rank==1,])
[1] 234
>
```

Plot



How did the algo know or learn to split first on rank?

How did it decide to split next on gpa?

Why did it stop at 234 nodes in Node 2?

Tree:: Entropy (H)

The idea of Entropy is from Themodynamics in Physics and Information Theory.

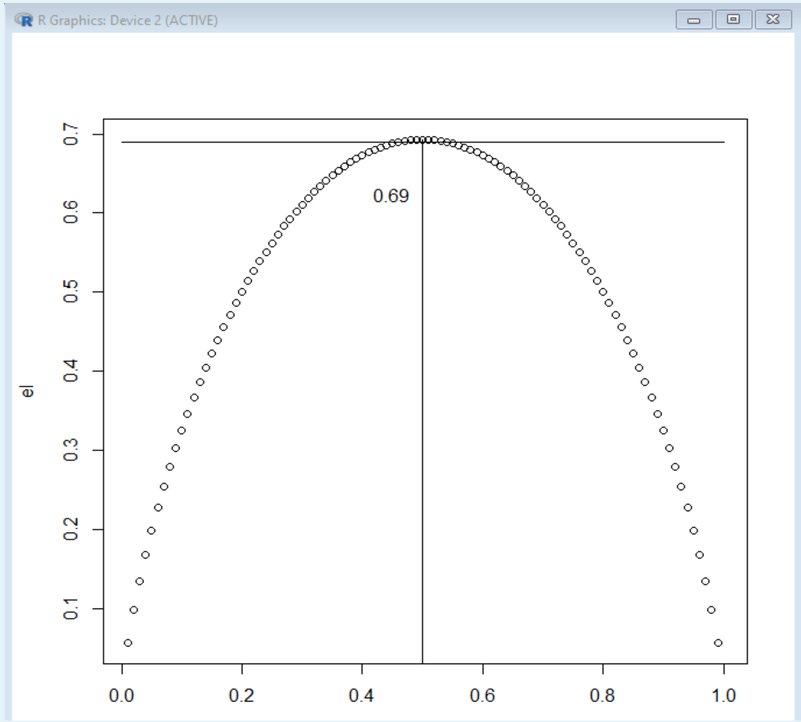
$$H(X) = - \Sigma\, p(x_i) \log (p(x_i)) + (1-p(x_i))*\log (1-p(x_i))$$

Here p is probability, a positive with a range of 0 to 1. Let us plot this and understand H visually.

H peaks when p=0.5

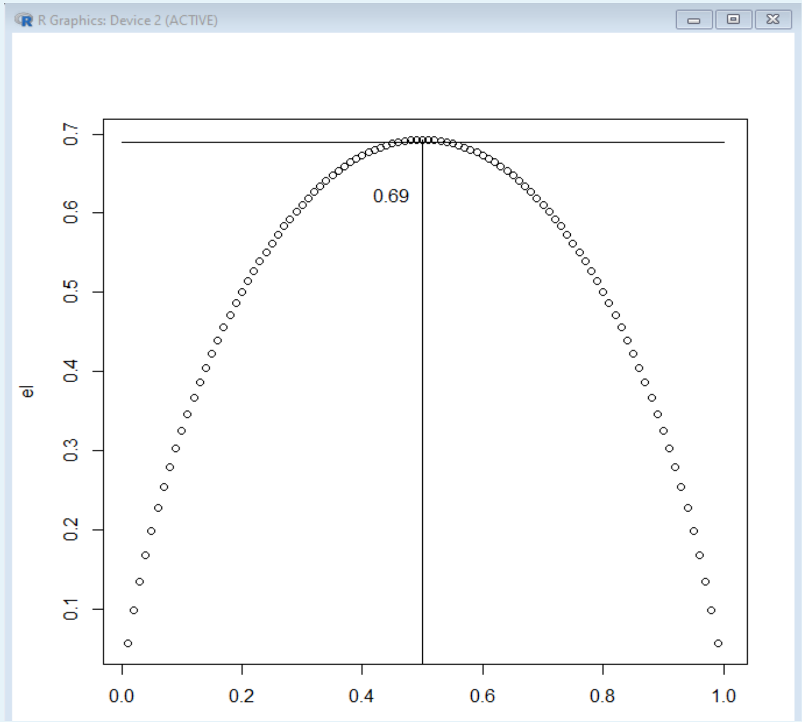
Why?

What happens when I Tell you u can go this or that way?
Is that help at all?



Tree:: Entropy (H)

```
> entropy
Error: object 'entropy' not found
> entropy <- function (p)
+ {
+  return ( -1 * ( p * log (p) + ( 1 -p) * log (1-p) ) ) ;
+ }
>
> x <- seq ( 0.01,1,0.01)
>
> el <-lapply(x,FUN=entropy)
>
> plot(x,el)
>
> eln<-as.numeric(el)
> maxe<-max(eln[!is.nan(eln)])
> slbl<-as.character(round(max(eln[1:99]),2))
> text(x=0.45,y=0.9*maxe,slbl)
>
>  vertical_y<-seq(0,maxe,maxe/10)
> vertical_x<-rep(0.5,length(vertical_y))
> lines(vertical_x,vertical_y[1:length(vertical_x)])
>
> horizontal_x<-seq(0.0,1,0.1)
> horizontal_y<-rep(round(maxe,2),11)
> lines(horizontal_x, horizontal_y)
>
```



Information Gain

Since, Entropy is a measure of uncertainty in the system, the entropy maxes out at p=0.5, as there is max uncertainty at that probability.

Information Gain achieved by following a certain branch is then equal to the reduction in entropy.

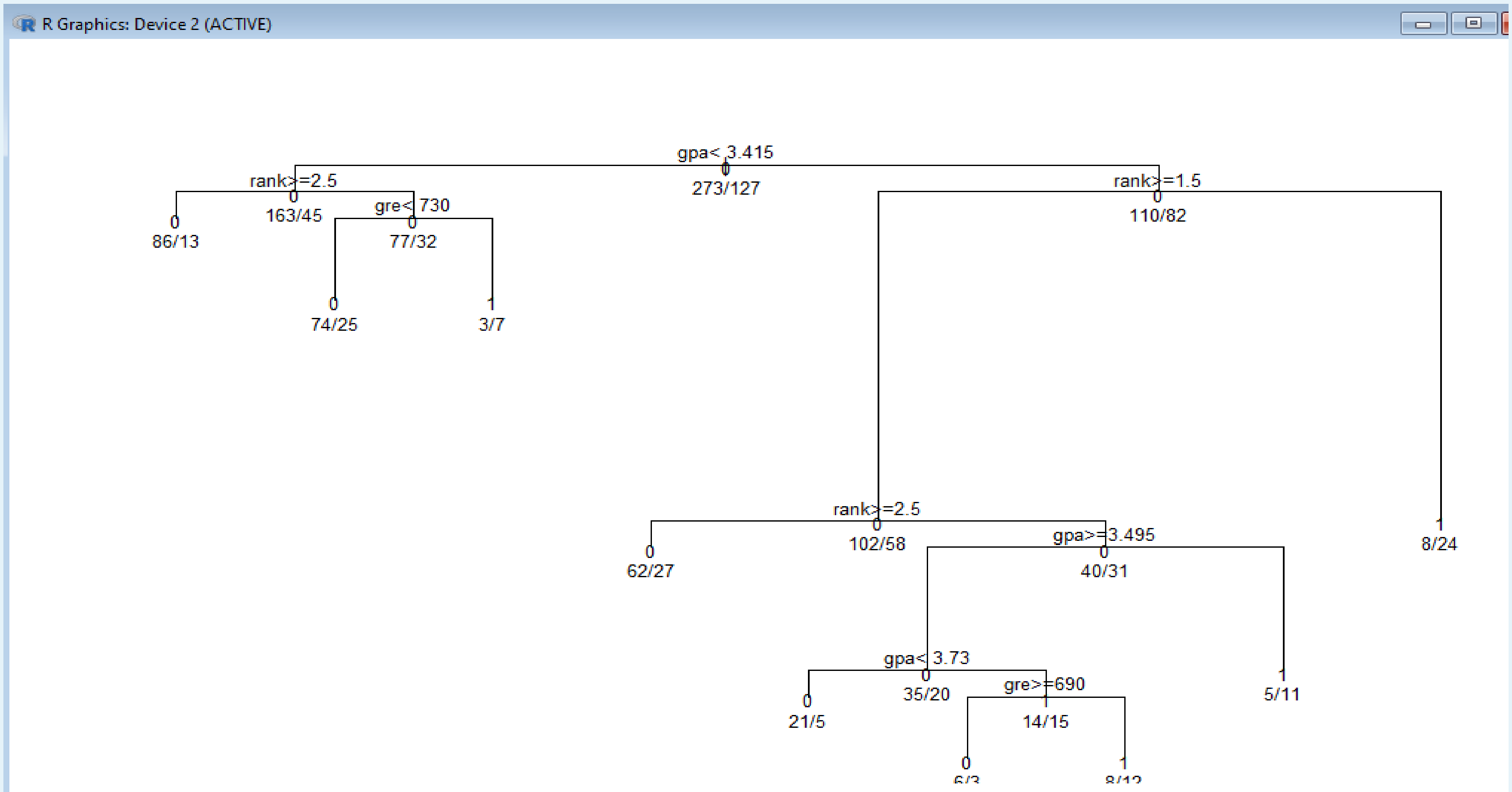
Calculate the entropy for each branch and find the difference. Select the one with the lower entropy, would you agree?

Please review this site https://www.dezyre.com/data-science-in-r-programming-tutorial/decision-tree-tutorial for a good review of Entropy, Information Gain, Gain Ratio and Gini Index.

Each M/L algorithm is theoretically founded, is a major paper and many new ideas... as you progress you should familiarize yourself with the concepts and why it works.

rpart

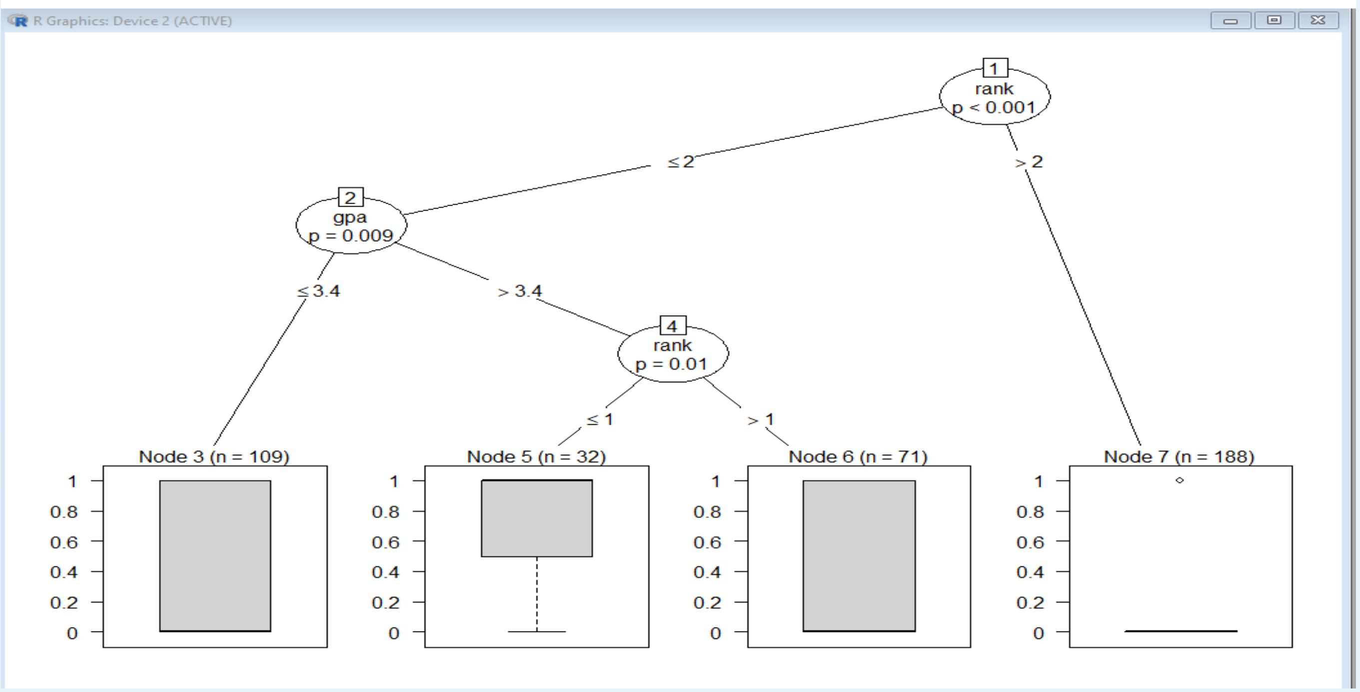
```
require(rpart)
model<-rpart(admit~ .,data=admit_data,method="class")
plot(model)
text(model,use.n=TRUE,all=TRUE,cex=0.8)
```



party

```
require(party)
party.model<-ctree
(admit~ .,data=admit_data)
plot(party.model)
```

```
> party::ctree(admit~.,data=admit_data)

         Conditional inference tree with 4 terminal nodes

Response:  admit
Inputs:  gre, gpa, rank
Number of observations:  400

1) rank <= 2; criterion = 1, statistic = 23.466
  2) gpa <= 3.4; criterion = 0.991, statistic = 8.8
    3)*  weights = 109
  2) gpa > 3.4
    4) rank <= 1; criterion = 0.99, statistic = 8.621
      5)*  weights = 32
    4) rank > 1
      6)*  weights = 71
1) rank > 2
  7)*  weights = 188
```



References

https://www.dezyre.com/data-science-in-r-programming-tutorial/decision-tree-tutorial

http://www.autonlab.org/tutorials/dtree.html
https://www.dezyre.com/data%20science-tutorial/decision-tree-tutorial
[1]See references listed in the doc that comes with rpart package, library/rpart/doc/longintro.pdf
[2] See references listed in the doc that comes with tree package, library/tree/html/tree.html
[2A] Breiman L., Friedman J. H., Olshen R. A., and Stone, C. J. (1984) *Classification and Regression Trees.* Wadsworth.
[2B] Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge. Chapter 7.
[3] http://www.rulequest.com/see5-unix.html

We will now move on to Boosting/Bagging and Ensemble methods.

Play, practice and learn about rpart,partys and tree package which also implement trees.