Decision Tree – Classification & Regression Tree

Learn How to Use Decision Tree for Predictive Modeling

Contents

Entropy

- Entropy measures the homogeneity of a sample or the degree of uncertainty. It is used as a parameter for checking the amount of uncertainty associated with a set of probabilities.
- Entropy lies between 0 and 1

  If the sample is completely homogeneous the entropy is 0 and if the sample is equally divided it has entropy of 1
- Entropy can be of two types, for each category and at the variable level
- Entropy of a category is calculated as:

$$\boxed{-\ P1 * \log_2(P1) - \ P2 * \log_2(P2)}$$

where,

P1 is the proportion of class 1

P2 is the proportion of class 2

Entropy of a Category

Let us consider survey data from three cities depicting shopper's preferred brand

City	Brand A Voters	Brand B Voters	Number of Voters	% of votes for Brand A	
Delhi	90	310	400	22.5%	77.5%
Chennai	10	90	100	10%	90%
Mumbai	100	100	200	50%	50%

Entropy for each city is calculated as:

Delhi: $-\ 0.225 * \log_2(0.225) - 0.775 * \log_2(0.775) = \mathbf{0.76919}$

Chennai: $-\ 0.1 * \log_2(0.1) - 0.9 * \log_2(0.9) = \mathbf{0.46900}$

Mumbai: $-\ 0.5 * \log_2(0.5) - 0.5 * \log_2(0.5) = \mathbf{1}$

Entropy at the Variable Level

- Entropy at the variable level can be derived by **adding weighted averages of all classes**
- Weights are the **proportion of respondents in each class to total respondents**

In the example under consideration,

Weights for the categories are

Delhi: $400/700 = \mathbf{0.5714}$

Chennai: $100/700 = \mathbf{0.1428}$

Mumbai: $200/700 = \mathbf{0.2857}$

Entropy at the variable level is

$$0.57 * 0.76919 + 0.14 * 0.46900 + 0.29 * 1 = \mathbf{0.79225}$$

Information Gain

- Information Gain is based on the decrease in entropy after a dataset is split on an attribute
- Constructing a decision tree is about finding attribute that returns the highest information gain

$$\text{Information Gain} = \text{Entropy of Sample (Dependent Variable)} - \text{Average Entropy of Any of the Independent Variable}$$

- Information gain can be interpreted as ability of reducing the uncertainty (Entropy) and hence increase predictability

Information Gain

City	Brand A Voters	Brand B Voters	Number of Voters	% of votes for Brand A	% of votes for Brand B
Delhi	90	310	400	22.5%	77.5%
Chennai	10	90	100	10%	90%
Mumbai	100	100	200	50%	50%

Entropy for complete sample is calculated as follows:

P1 = (Total Brand A Voters/Total Voters)

P2 = (Total Brand B Voters/Total Voters)

$$\text{Entropy} = -(0.286) * \log_2(0.286) - (0.714) * \log_2(0.714) = \mathbf{0.86312}$$

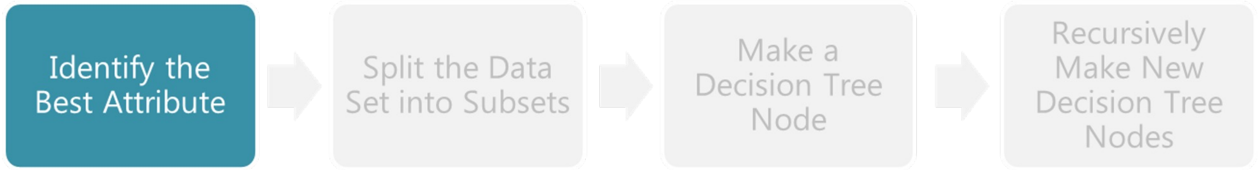
Information Gain

Entropy at the variable level (Weighted average)

$$0.86312 - \underset{\uparrow}{0.79225} = \mathbf{0.070868}$$

Information Gain and ID3 Algorithm

- Let us now go back to the basic ID3 algorithm; Step 1 of which is 'Identify the Best Attribute'



- Information Gain value is used to determine which attribute is the "best" – the attribute with most information gain is chosen
- Information gain for a variable is high when that variable has the low entropy at the variable level (Weighted average)
- Low entropy for a variable implies the classification based on that attribute is fairly homogenous , hence this attribute is selected as the first best attribute
- The same process is repeated till no attributes remain

CART Algorithm

- Classification and Regression Tree (CART) algorithm generates a binary decision tree, by splitting a node into two branches
- Root node contains the complete sample (training data)
- The splits are univariate – each split depends on the value of only one predictor variable

The algorithm can be divided into three steps:

Gini impurity is used as the splitting criteria in CART

CART Algorithm

- As the name suggests, this algorithm can be used for both classification and regression purposes. Splitting criteria for the dependent variable depends on its format:

	Categorical	Continuous
Dependent Variable	Gini Impurity, Twoing Criterion and Ordered Twoing Criterion (When it is ordinal)	Least Squares Deviation for the impurity measures

- Splits are determined as follows:

	Nominal	Ordinal or Continuous
Independent Variable		

Gini Impurity

- Gini Impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset

$$\boxed{\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2}$$

where

$p(i|t)$: Fraction of records belonging to class i at node t

- It reaches its minimum (zero) when all cases in the node fall into a single target category
- Attribute with the smaller Gini index is considered for the split

Package "rpart“ in R

- Package "rpart" uses methods which implement many ideas found in the CART algorithm proposed by Breiman, Friedman, Olshen and Stone.
- Recursive splitting in “rpart“ is based on the concept of impurity and the package offers two methods for quantifying impurity :
  - Entropy (Also called Information Gain)
  - Gini impurity

> The algorithm works by making the best possible choice at each particular stage, without any consideration of whether those choices remain optimal in future stages. That is, it makes a locally optimal decision at each stage. Such a choice may turn out to be sub-optimal in the overall scheme of things. The algorithm does not find a globally optimal tree.

Case Study – Predicting Loan Defaulters

Background

- The bank possesses demographic and transactional data of its loan customers. If the bank has a robust model to predict defaulters it can undertake better resource allocation.

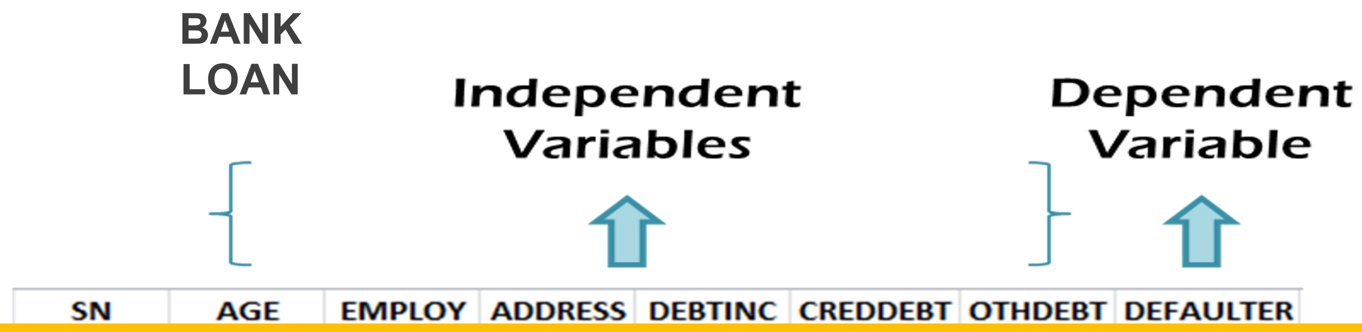
Objective

- To predict whether the customer applying for the loan will be a defaulter

Available Information

- Sample size is 700
- Age group, Years at current address, Years at current employer, Debt to Income Ratio, Credit Card Debts, Other Debts are the independent variables
- Defaulter (=1 if defaulter, 0 otherwise) is the dependent variable

Data Snapshot



Column	Description	Type	Measurement	Possible Values
SN	Serial Number	Numeric	-	-
AGE	Age Groups	Categorical	1(<28 years),2(28-40 years),3(>40 years)	3
EMPLOY	Number of years customer working at current employer	Continuous	-	Positive value
ADDRESS	Number of years customer staying at current address	Continuous	-	Positive value
DEBTINC	Debt to Income Ratio	Continuous	-	Positive value
CREDDEBT	Credit to Debit Ratio	Continuous	-	Positive value
OTHDEBT	Other Debt	Continuou	-	Positive value

Package “rpart” in R

```
# Install & load Package “rpart”
# Import Data
install.packages("rpart")
library(rpart)
```

- ❑ The package name stands for Recursive Partitioning and Regression Trees.
- ❑ It can generate both classification and regression trees in R.
- ❑ The package uses CART algorithm by default and can implement the algorithm using information gain or Gini impurity as the splitting criteria.

```
bankloan<-read.csv("BANK LOAN.csv",header=T)
str(bankloan)

# Convert Defaulter & Age variables to factor

bankloan$DEFAULTER<-as.factor(bankloan$DEFAULTER)
bankloan$AGE<-as.factor(bankloan$AGE)
```

Classification Tree Using Gini Impurity

```
# Classification Tree in "rpart" Using Gini Impurity

rpart_c<-rpart(DEFAULTER~AGE+EMPLOY+ADDRESS+DEBTINC+CREDDEBT+OTHDEBT,
               data=bankloan, method="class")
```

- **rpart()** fits an **rpart** model.
- First argument in the function is the formula (With no interaction terms)
- **data=** giving the data object.
- **method=** uses one of the four options **"anova", "poisson", "class"** or **"exp"**, depending on the type of the dependent variable. For classification tree, we have specified **method=class**
- Splitting function parameters can be specified using **parms=list()**. When nothing else is specified, the function uses **"gini"** as the splitting criteria.

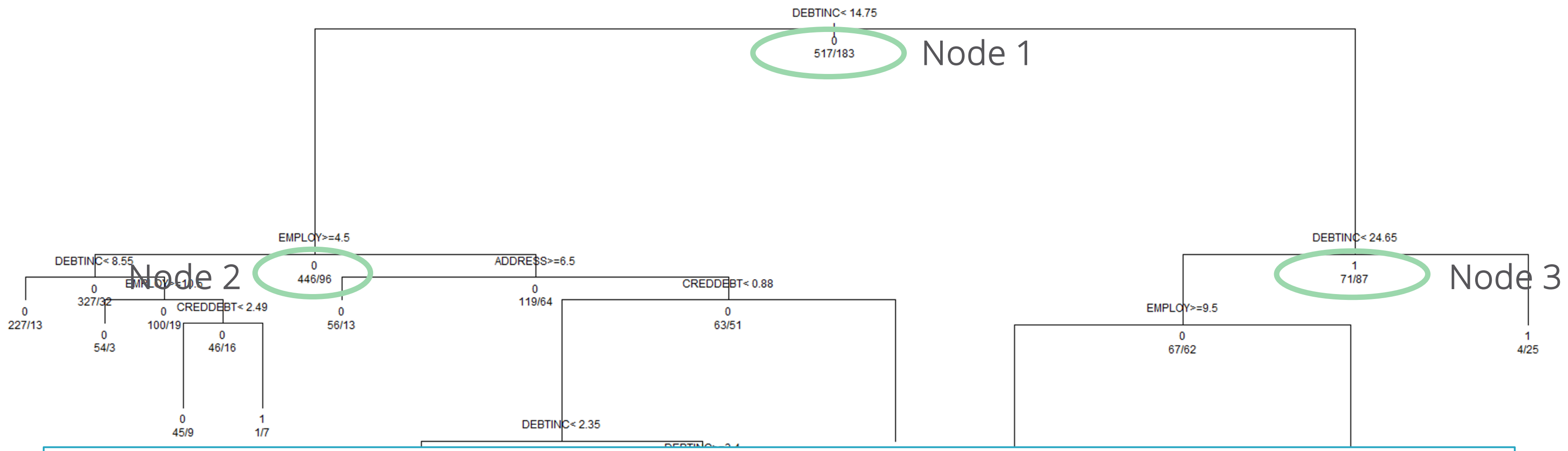
Classification Tree Using Gini Impurity

```
# Classification Tree in "rpart" Using Gini Impurity
```

```
par(mfrow=c(1,1),xpd=NA)
plot(rpart_c)
text(rpart_c,splits=T,use.n=T,all=T,cex=0.75)
```

- **par()** is used to ensure the plot fits with correct margins.
- **xpd=NA** clips all plotting to the device region.
- **plot()** produces an unlabelled plot.
- **text()** command is used for adding labels.
- **splits=, use.n=, all=** are logical arguments which tell R which values to show in the plot.
- **cex=** controls text proportion.

Classification Tree Interpretation



Interpretation :

- Due to a large number of continuous predictors, a tree with several nodes and branches is generated.
- Tree starts with all 700 observations. 517 are non-defaulters (0) and the remaining 183 are defaulters (1).
- DEBTINC is the first split variable, left branch is <14.75 and right branch is >14.75. 542/700 have DEBTINC<14.75.
- EMPLOY is the second split on left branch, which further divides 542 obs. into 446 non-defaulters (0) and the remaining 96 are defaulters (1).
- The algorithm progresses till no further variable split is left.

Classification Tree Using Information Gain

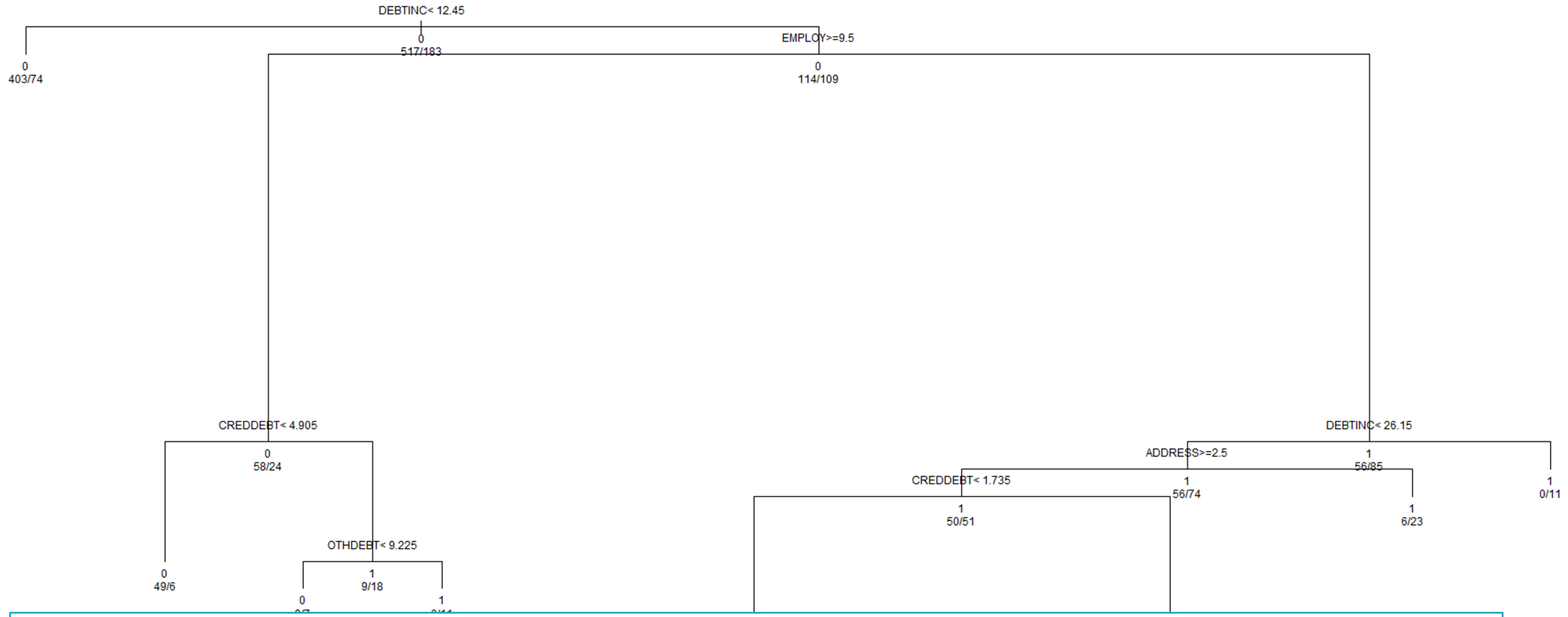
```
# Classification Tree in “rpart” Using Information Gain
```

```
rpart_inf<-rpart(DEFAULTER~AGE+EMPLOY+ADDRESS+
                         DEBTINC+CREDDEBT+OTHDEBT,
               data=bankloan, method="class",
               parms=list(split="information"))
```

parms=list(split=) is used to specify information gain as the splitting parameter.

```
par(mfrow=c(1,1),xpd=NA)
plot(rpart_inf)
text(rpart_inf,splits=T,use.n=T,all=T,cex=0.75)
```

Classification Tree Using Information Gain



Interpretation :

- Tree starts with all 700 observations. 517 are non-defaulters (0) and the remaining 183 are defaulters (1).
- DEBTINC is the first split variable, left branch is <12.45 and right branch is >12.45.
- 477/700 have DEBTINC<12.45 which further divides into 403 non-defaulters (0) and the remaining 74 are defaulters (1).

Case Study – Modeling Motor Insurance Claims

Background

- A car insurance company collects range of information from their customers at the time of buying and claiming insurance. The company wishes to check if any of the information gathered can be used to model and predict the claim amounts

Objective

- To model motor insurance claim amount based on vehicle related information collected at the time of registering and claiming insurance

Available Information

- Sample size is 1000
- Independent Variables: Vehicle Information – Vehicle Age, Engine Capacity, Length and Weight of the Vehicle
- Dependent Variable: Claim Amount

Data Snapshot

Motor Claims

Independent variables | Dependent variable

vehage	CC	Length	Weight	claimamt
4	1495	4250	1023	72000
2	1061	3495	875	72000
2	1405	3675	980	50400
7	1298	4090	930	39960
2	1495	4250	1023	106800
1	1086	3565	854	69592.8

vations

Columns	Description	Type	Measurement	Possible values
vehage	Age of the vehicle at the time of claim	integer	Years	positive values
CC	Engine capacity	numeric	cc	positive values
Length	Length of the vehicle	numeric	mm	positive values
Weight	Weight of the vehicle	numeric	kg	positive values
claimamt	Claim amount	numeric	INR	positive values

Regression Tree in Package “rpart”

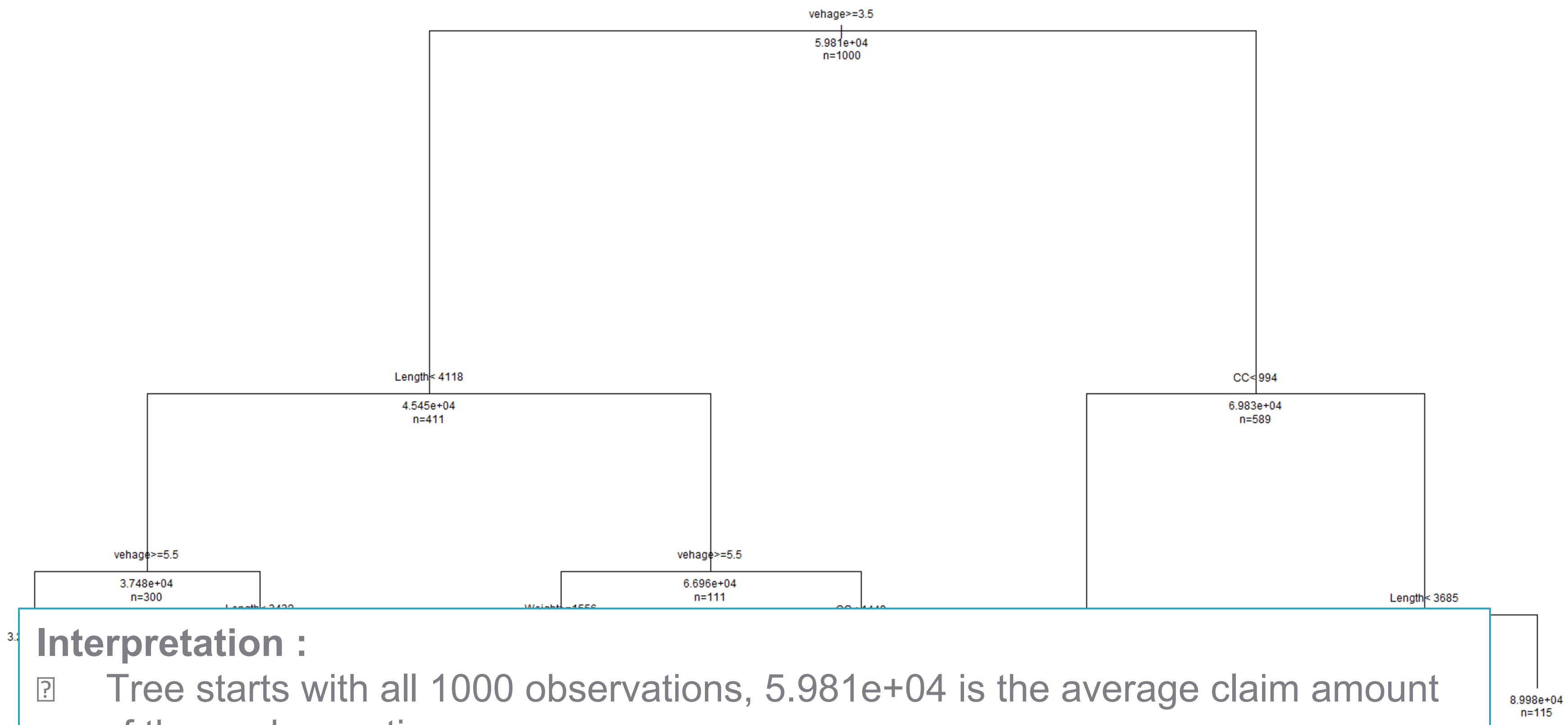
```
# Regression Tree in “rpart”
# Since we intend to plot a regression tree, new data having
# continuous dependent variable is imported.
motor<-read.csv("Motor Claims.csv",header=T)
str(motor)

rpart_r<-rpart(claimamt~vehage+CC+Length+Weight,
                data=motor, method="anova",
                parms = list(split="information"))

par(mfrow=c(1,1),xpd=NA)
plot(rpart_r)
text(rpart_r,splits=T,use.n=T,all=T,cex=0.75)
```

method="anova" ensures R generates a regression tree in rpart().

Regression Tree in Package “rpart”



Interpretation :

- Tree starts with all 1000 observations, 5.981e+04 is the average claim amount of these observations.
- vehage is the first split variable, left branch is >=3.5 and right branch is <3.5.
- 411 have vehage >=3.5 which has 4.545e+04 average claim amount .
- The process continues till there is no variable left for splitting.

Get an Edge!

> Simple plot of an rpart object isn't the most efficient in terms of understanding and interpretation. Use package **"rpart.plot"** for a better visualisation of rpart trees.

```
install.packages("rpart.plot")
library(rpart.plot)
rpart.plot(rpart_r,type=1,extra=1,branch=0,cex=0.8)
```

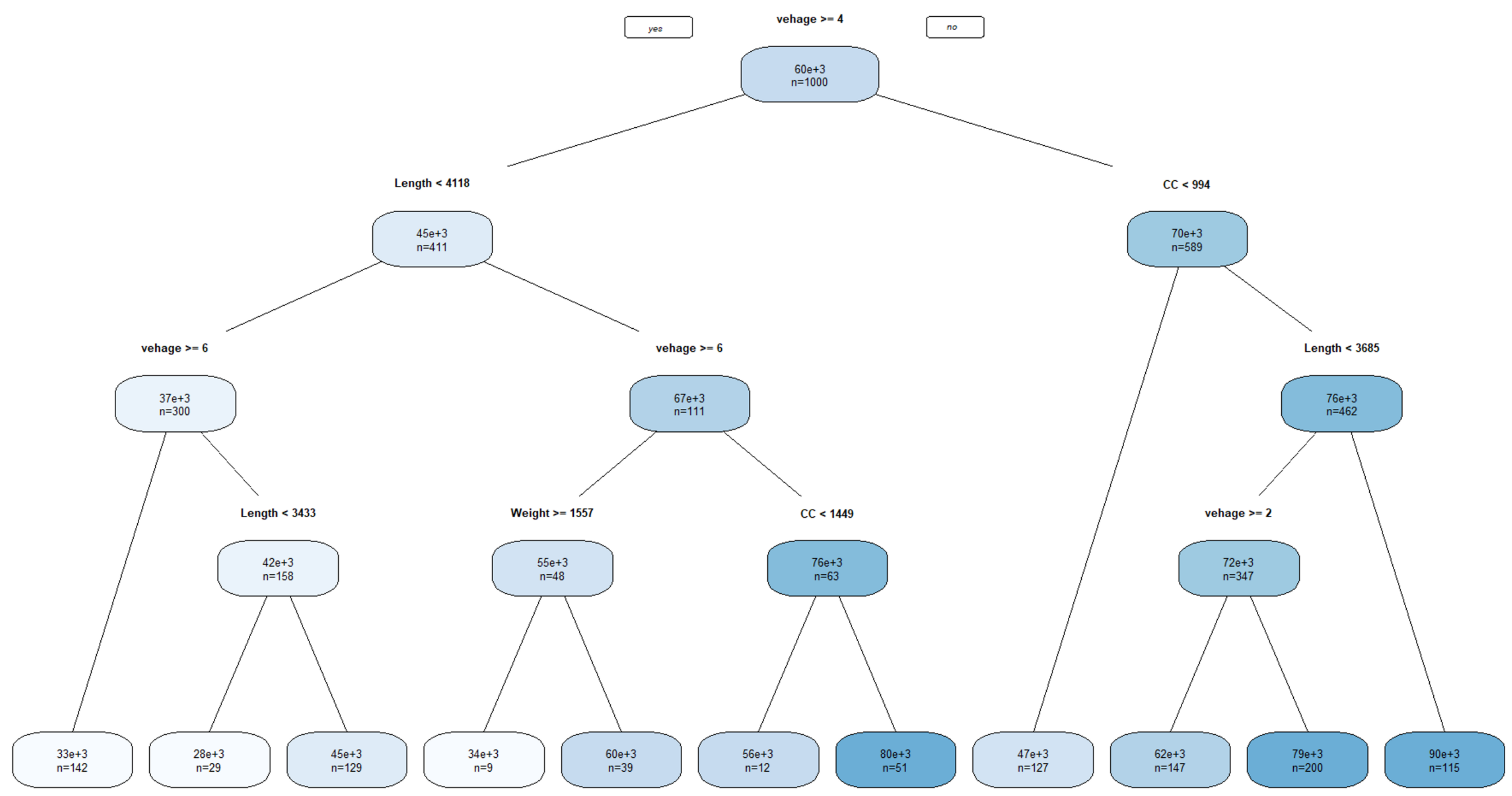
type= the function gives five possible types of plots. 0 is default, 1 labels all nodes, not just leaves

extra= gives extra information about the nodes. 1 displays the number of observations that fall in the node

branch= controls the shape of the branch lines. 0 plots V shaped branches

cex= controls text proportion

Get an Edge!



yes
vehage >= 4
no
60e+3
n=1000
Length < 4118
45e+3
n=411
CC < 994
70e+3
n=589
vehage >= 6
37e+3
n=300
vehage >= 6
67e+3
n=111
Length < 3685
76e+3
n=462
Length < 3433
42e+3
n=158
Weight >= 1557
55e+3
n=48
CC < 1449
76e+3
n=63
vehage >= 2
72e+3
n=347
33e+3
n=142
28e+3
n=29
45e+3
n=129
34e+3
n=9
60e+3
n=39
56e+3
n=12
80e+3
n=51
47e+3
n=127
62e+3
n=147
79e+3
n=200
90e+3
n=115

Quick Recap

In this session, we learnt about **two major splitting criteria in decision tree, Information Gain and Gini Impurity:**

Decision Tree Algorithms

- ID3 uses top-down, greedy search method to build a classification decision tree
- CART algorithm generates a binary decision tree, by splitting a node into two branches. Root node contains the complete sample.

Entropy, Information Gain and Gini Impurity

- Entropy measures the homogeneity of a sample
- Information Gain is based on the decrease in entropy after a dataset is split on an attribute
- Information Gain= Entropy of Sample − Average Entropy of Any of the Independent Variable
- Gini Impurity measures how often a randomly chosen element from the set would be incorrectly labeled

CART in R

- **`rpart()`** in package "**`rpart`**" generates CART trees
- Use **`method=`** to specify whether to generate classification or regression tree