Introduction to Decision Tree - I

Contents

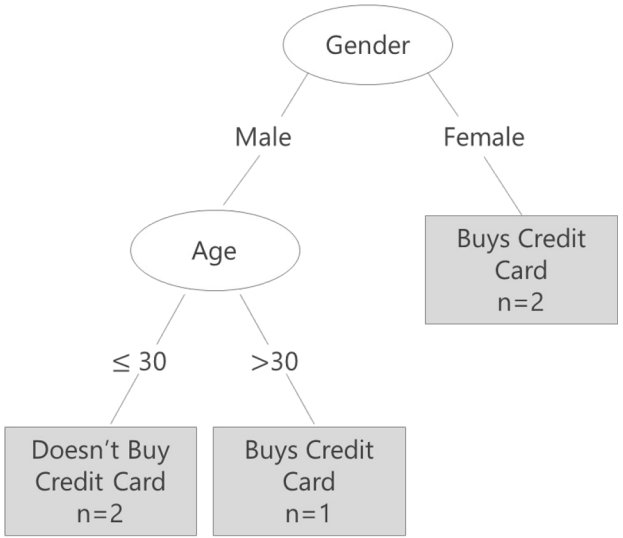
Introduction to Decision Tree

- One of the popular predictive modeling techniques, **Decision Tree** uses data mining techniques for predicting a class or value
- Decision Tree **breaks down a data set into smaller subsets and presents association between target variable(dependent) and independent variables as a tree structure.**
- Final result is a **tree with Decision Nodes and Leaf Nodes.**
- A decision node has two or more branches and leaf node represents a classification or decision.

Decision Tree – Basic Framework

Suppose we have information about 5 customers and their decision about buying a credit card. This data can be represented as a Decision Tree:

Customer No.	Gender	Age	Occupation	Buys Credit Card
01	Male	≤30	Student	No
02	Male	≤30	Professional	No
03	Female	≤30	Business	Yes
04	Male	>30	Business	Yes
05	Female	>30	Professional	Yes



- First subset is based on **Gender. All females opt for credit cards.**
- Males, however, can be split further, based on **Age. Male customers of age ≤30 years do not buy a credit card, whereas those >30 do buy cards.**

What is CART ?

The term **Classification And Regression Tree (CART)** analysis is an umbrella term used to refer to predictive decision tree procedures, first introduced by Breiman et al.

Classification Tree

When the predicted outcome is the class to which the data belongs

In such cases, **dependent variable is categorical**

Independent variables can be either continuous or categorical or both

Regression Tree

When the predicted outcome can be considered a real number

In such cases, **dependent variable is continuous**

Independent variables can be either continuous or categorical or both

Decision Tree Algorithms

Trees used for regression and trees used for classification have some similarities - but also some differences, such as the procedure used to determine where to split.

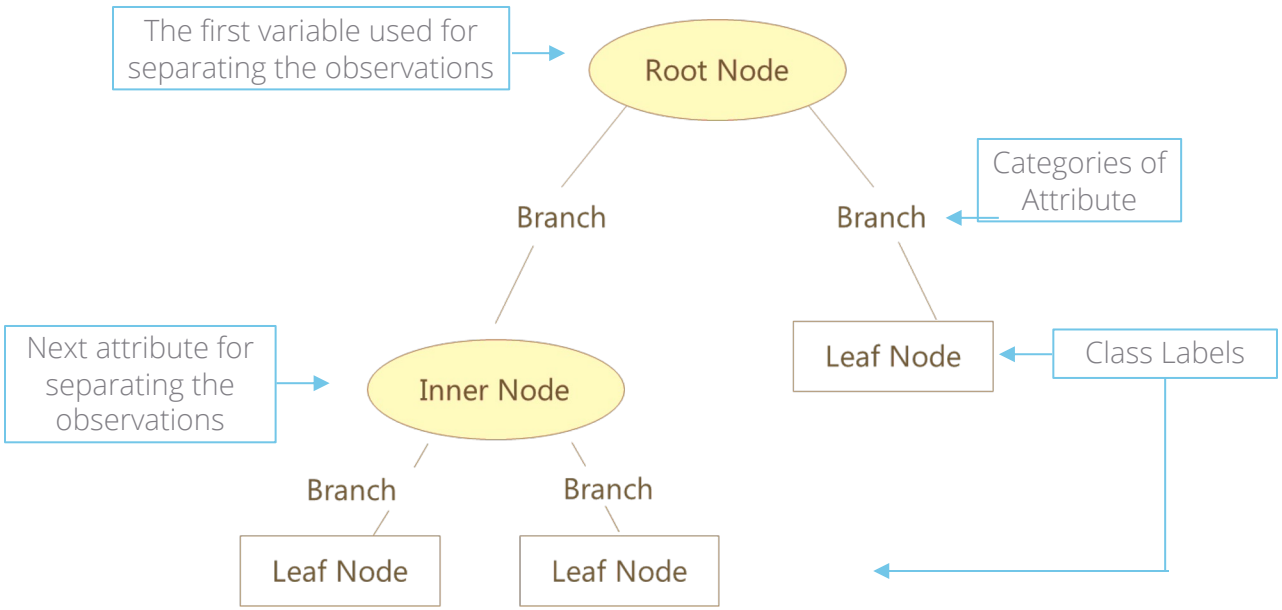
There are many specific decision-tree algorithms. Notable ones include:

1. ID3 (Iterative Dichotomiser 3)
2. C4.5 (Successor of ID3)
3. CART (Classification And Regression Tree)
4. CHAID (CHi-squared Automatic Interaction Detector)
   Performs multi-level splits when computing classification trees
5. MARS (Multivariate Adaptive Regression Splines)
   Extends decision trees to handle numerical data better
6. Conditional Inference Trees
   Statistics-based approach that uses non-parametric tests as splitting criteria, corrected for multiple testing to avoid overfitting. This approach results in unbiased predictor selection and does not require pruning

Decision Tree – Basic Components

Component	Description	Alternate terms
Root node	Has no incoming edges and zero or more outgoing edges	Parent node
Inner node	Each has exactly one incoming edge and two or more outgoing edges	Decision nodes / Child nodes
Leaf node	Each has exactly one incoming edges and no outgoing edges	Terminal nodes
Branches	Categories of attributes	Edges

Decision Tree – Basic Components



* Class labels show observations belong to which class. The leaf node also shows Number of observations and Error rate (Actual classification vs classification given by the tree)

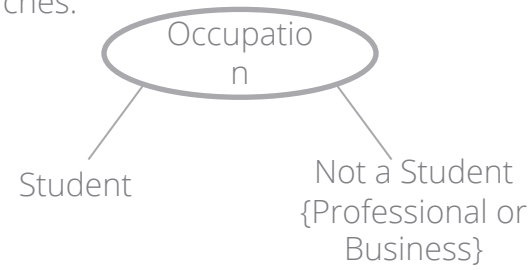
Binary and Non Binary Trees

Consider the same example illustrated earlier.

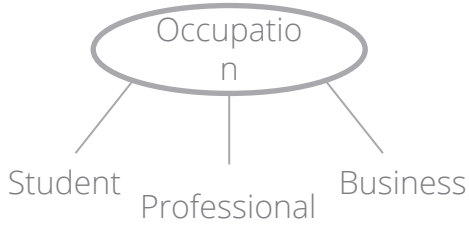
Occupation is the nominal attribute under consideration, with three distinct categories.

Customer No.	Occupation	Buys Credit Card
01	Student	No
02	Professional	No
03	Business	Yes
04	Business	Yes
05	Professional	Yes

There are two ways in which a decision node can be split into further branches:



Binary Tree



Non-Binary Tree

Introduction to CHAID

The acronym **CHAID stands for Chi Square Automatic Interaction Detection**

CHAID is ,

- Developed in South Africa and was published in 1980 by Gordon V. Kass, who had completed a PhD thesis on this topic
- A type of decision tree technique used for classification problem
- An **exploratory method used to study the relationship between a dependent variable and a series of independent variables**
- **Builds non-binary trees** (trees where more than 2 branches can attach to the root node or any node)
- Can be effectively applied in case of large data sets

Introduction to CHAID

- Chi-square test of independence is used at each step to determine the strength of relationship between dependent and independent variables.
- The independent variables should be categorical so if data contains continuous predictors then they must be converted to categorical variables.
- Chi-square test is used to measure association between dependent and independent variables as well as amongst independent variables.
- Each pair is assessed to identify the least significantly different via a Bonferroni adjusted p-value which is calculated for merged cross table.

? **Why are we doing this?**
The CHAID algorithm is based on Chi-square test of independence, which essentially is a test for checking association between two categorical variables.

Case Study – Employee Churn Model

Background

- A company has comprehensive database of its past and present workforce, with information on their demographics, education, experience and hiring background as well as their work profile. The management wishes to see if this data can be used for predictive analysis, to control attrition levels.

Objective

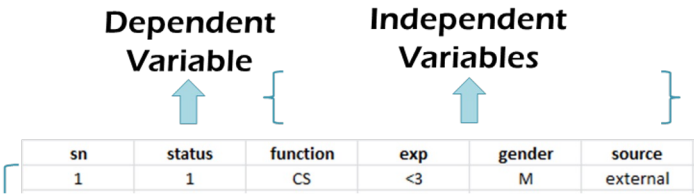
- To develop an Employee Churn model via Decision Tree

Available Information

- Sample size is 83
- **Gender, Experience Level** (<3, 3-5 and >5 years), **Function** (Marketing, Finance, Client Servicing (CS)) **and Source** (Internal or External) are independent variables
- **Status** is the dependent variable (=1 if employee left within 18 months from joining date)

Data Snapshot

EMPLOYEE CHURN DATA



Columns	Description	Type	Measurement	Possible values
sn	Serial Number	-	-	-
status	= 1 If the Employee Left Within 18 Months of Joining = 0 Otherwise	Binary	1,0	2
function	Employee Job Profile	Categorical	CS, FINANCE, MARKETING	3
exp	Experience in Years	Categorical	<3,3-5,>5	3
gender	Gender of the Employee	Categorical	M,F	2
source	Whether the Employee was Appointed via Internal or External Links	categorical	external, internal	2

CHAID Algorithm

- CHAID algorithm has three main steps:

Step 1: Preparing Predictors

- All continuous predictor variables are converted to categorical variables. Example: Experience Level (<3, 3-5 and >5 years)
- The categorical variables are considered with the natural categories
- Example: Function (Marketing, Finance, Client Servicing)

CHAID Algorithm

Step 2: Merging Categories

- In this step categories of predictors are assessed for possible merging
- For every predictor all possible pairs of categories are considered
- Example: For predictor "function", Marketing-Finance, Marketing-CS and Finance-CS are assessed pairwise
- A chi-squared test of independence is used to decide whether categories in the pair are to be combined or not
- The level of significance value used in the testing is named as alpha-to-merge
- If the adjusted p-value (Bonferroni) for a particular pair is greater than alpha-to-merge, then the categories are merged together
- This merging is continued until no categories can be merged further

CHAID Algorithm

Step 3: Selecting the Split Variable

- Chi-square test of independence between dependent variable and each of predictors is performed.
- Predictor variable with the smallest adjusted p-value (Bonferroni) is considered as a first split variable and appears at first node.
- If there is tie between 2 predictors then the predictor with largest chi-square statistic value is considered as the split variable.
- The level of significance value is named as alpha-to-split is decided apriori.
- The process is repeated at each node using subset of data defined by the node.
- If the smallest (Bonferroni) adjusted p-value for any predictor is greater than some alpha-to-split value, then no further splits will be performed, and the respective node is a terminal node.

This process is continued until no further splits can be performed

Bonferroni Adjustment in CHAID

- Bonferroni adjustment acts as a safeguard against false decisions in case of multiple hypothesis testing problems.
- If n dependent or independent tests are performed simultaneously at level of significance α, then there is a chance of false rejection or acceptance of null hypothesis.
- To control the false rejection/acceptance, α (Probability of type I error) is adjusted to α/n, making the decision rule stricter .
- Instead of adjusting the α value , one can also adjust the p-value obtained for each test .
- $p_{adj} = \min(p_{raw} * n, 1)$, p_{raw} is the p-value for individual test , tested at α.

In CHAID algorithm, both the steps (Merging /Splitting) , include multiple chi-square tests, hence Bonferroni adjusted P-values are used .

Package “CHAID” in R

```
# Importing the Data

empdata<-read.csv("EMPLOYEE CHURN DATA.csv",header=T)


# Decision Tree Using Package "CHAID"

install.packages("partykit")

install.packages("CHAID",
                 repos="http://R-Forge.R-project.org",type="source")
library(CHAID)

library(partykit)
```

Package **“CHAID“** is no longer available on CRAN. **“CHAID“** can be downloaded from R-Forge Repository. We also install package **“partykit”** because CHAID is built upon **“partykit”**.

Package “CHAID” in R

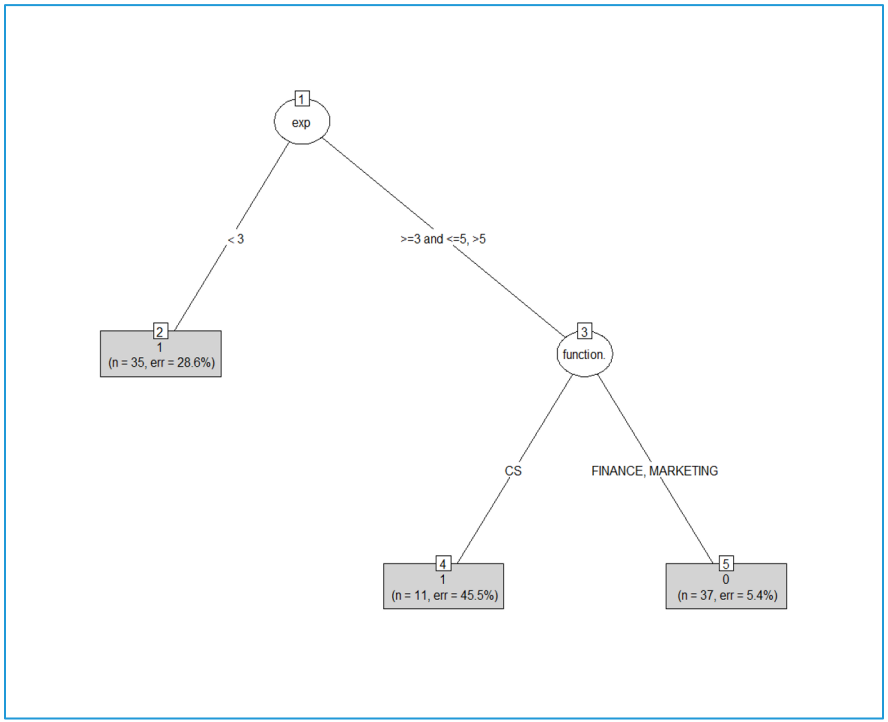
```
# Coverting all variables into factor variables
empdata$status<-as.factor(empdata$status)
empdata$function.<-as.factor(empdata$function.)
empdata$exp<-as.factor(empdata$exp)
empdata$gender<-as.factor(empdata$gender)
empdata$source<-as.factor(empdata$source)
```

```
# Decision Tree Using Package "CHAID"
tree<-chaid(formula=status~function.+exp+gender+source,data=empdata)
tree
```

```
> tree

Model formula:
status ~ function. + exp + gender + source

Fitted party:
[1] root
|   [2] exp <3: 1 (n = 35, err = 28.6%)
|   [3] exp >=3 and <=5, >5
|   |   [4] function. in CS: 1 (n = 11, err = 45.5%)
|   |   [5] function. in FINANCE, MARKETING: 0 (n = 37, err = 5.4%)

Number of inner nodes:    2
Number of terminal nodes: 3
```

Decision Tree Using Package “CHAID”

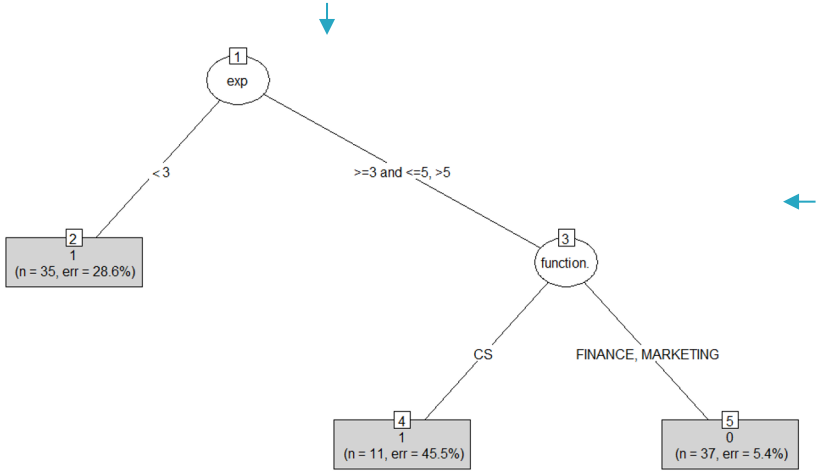
```
# Plotting chaid output
plot(tree,type="simple")
```



There are 35 employees with exp<3. Out of which 25 Left within 18 months. 25/35=0.714. Err=28.6%

Decision Tree Interpretation – Case Study

Exp is the first split variable. 71% of employees with less than 3 years of experience left the organisation within the first 18 months of joining.



Next split variable is Function. Out of the more experienced employees those working in Client Servicing are more likely to leave.
55% of client servicing employees left the organisation despite having more than 3 years of overall work experience.

With an error rate of 5%, employees with more than 3 years of experience working in the finance and marketing departments are least likely to leave within first 18 months of joining

Predicted Probability and ROC Curve

```
# Predicted Probability
predtree<-predict(tree,empdata,type="prob")


head(predtree)
```

- **predict()** returns predicted values.

```
# Output
          0         1
1 0.2857143 0.7142857
2 0.2857143 0.7142857
3 0.4545455 0.5454545
4 0.4545455 0.5454545
5 0.2857143 0.7142857
6 0.4545455 0.5454545
```

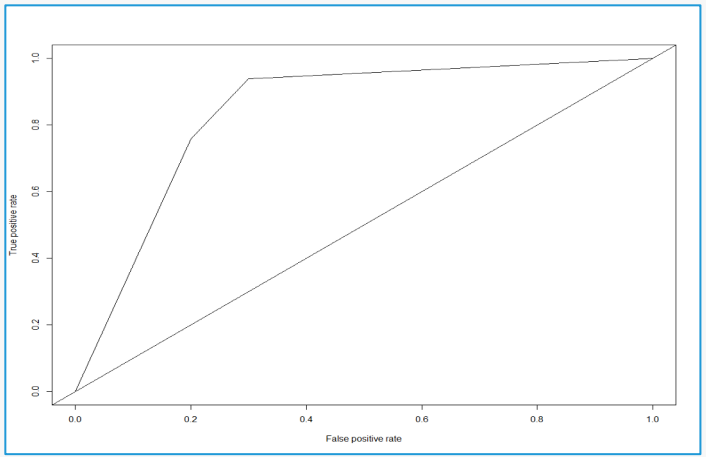
- First column gives probability for Y=0 and second column gives probability of Y=1

Predicted Probability and ROC Curve

```
#ROC Curve and Area Under ROC Curve
install.packages("ROCR")
library(ROCR)

pred<-prediction(predtree[,2],empdata$status)
perf<-performance(pred,"tpr","fpr")

plot(perf)

abline(0,1)
```

- **prediction()** function prepares data required for ROC curve.
- **performance()** function creates performance objects, "tpr“ (True positive rate), "fpr” (False positive rate).
- **plot()** function plots the objects created using performance
- **abline()** adds a straight line to the plot.

Predicted Probability and ROC Curve




```
auc<-performance(pred,"auc")
auc@y.values
[[1]]
[1] 0.8393939
```

- **The estimated area under ROC curve is 0.8394**

Quick Recap

Decision Tree	• Breaks down a data set into smaller subsets and presents association between target variable(dependent) and independent variables as a tree structure. • A decision tree is made of root node, internal nodes and terminal nodes, joined by branches.
Types of Decision Tree	• The term Classification And Regression Tree (CART) analysis is an umbrella term used to refer to predictive decision tree procedures • Classification Tree Categorical dependent variable • Regression Tree Continuous dependent variable
Decision Tree Algorithms	• ID3 (Iterative Dichotomiser 3) • C4.5 (Successor of ID3) • CART (Classification And Regression Tree) • CHAID (CHi-squared Automatic Interaction Detector) • MARS (Multivariate Adaptive Regression Splines) • Conditional Inference Trees
CHAID Tree	• **`chaid()`** in package **"CHAID"** fits a classification tree by CHAID algorithm.

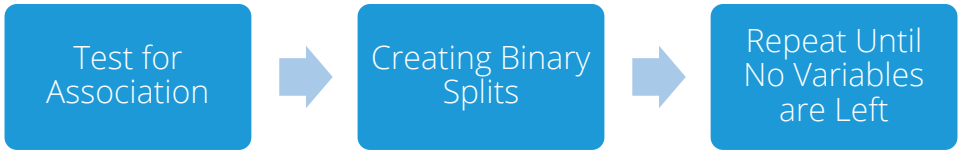
Decision Tree - II

Learn Classification and Prediction via Data Mining

Contents

Conditional Inference Tree Algorithm

- Conditional Inference (CI) Tree algorithm can also be divided into three main steps:



Step 1: Test for Association

- The algorithm tests if any independent variables are associated with the given response variable, and chooses the variable that has the strongest association with the response, i.e. Variable with the smallest p-value based on permutation test is chosen

Conditional Inference Tree Algorithm

Step 2: Split Variables

- The algorithm makes a binary split in this variable, dividing the dataset into two subsets
- In case of a binary predictor with values A and B, one subset will contain all observations with value A, and the other will contain all cases with value B. If a variable has more levels, one group may have values A and B, and the other may contain observations with C
- If the variable is quantitative, the range of its values can be split into two, e.g. values from 0 to 100 can be split into two subsets: from 0 to 50 and from 51 to 100; OR 0-30 and 31 to 100, and so on.

Conditional Inference Tree Algorithm

Step 3: Repeat Until No Variables are Left

- The first two steps are repeated for each subset until there are no variables that are associated with the outcome at the pre-defined level of statistical significance. This is why the algorithm is called recursive.

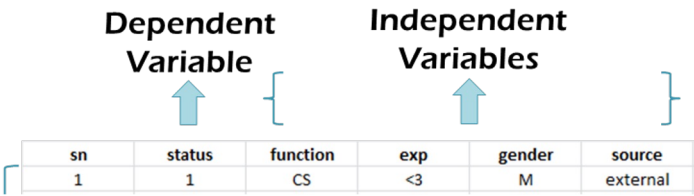
Tests Used in CI Algorithm

- Conditional Inference algorithm can be used for Classification as well as Regression Models.
- Structure of the algorithm remains the same, tests used for checking variable association change as per variable type.

Dependent Variable	Independent Variables	Test
Categorical	Categorical	Chi-square
Continuous	Continuous	Correlation
Continuous	Categorical	ANOVA

Data Snapshot

EMPLOYEE CHURN DATA



Columns	Description	Type	Measurement	Possible values
sn	Serial Number	-	-	-
status	= 1 If the Employee Left Within 18 Months of Joining = 0 Otherwise	Binary	1,0	2
function	Employee Job Profile	Categorical	CS, FINANCE, MARKETING	3
exp	Experience in Years	Categorical	<3,3-5,>5	3
gender	Gender of the Employee	Categorical	M,F	2
source	Whether the Employee was Appointed via Internal or External Links	categorical	external, internal	2

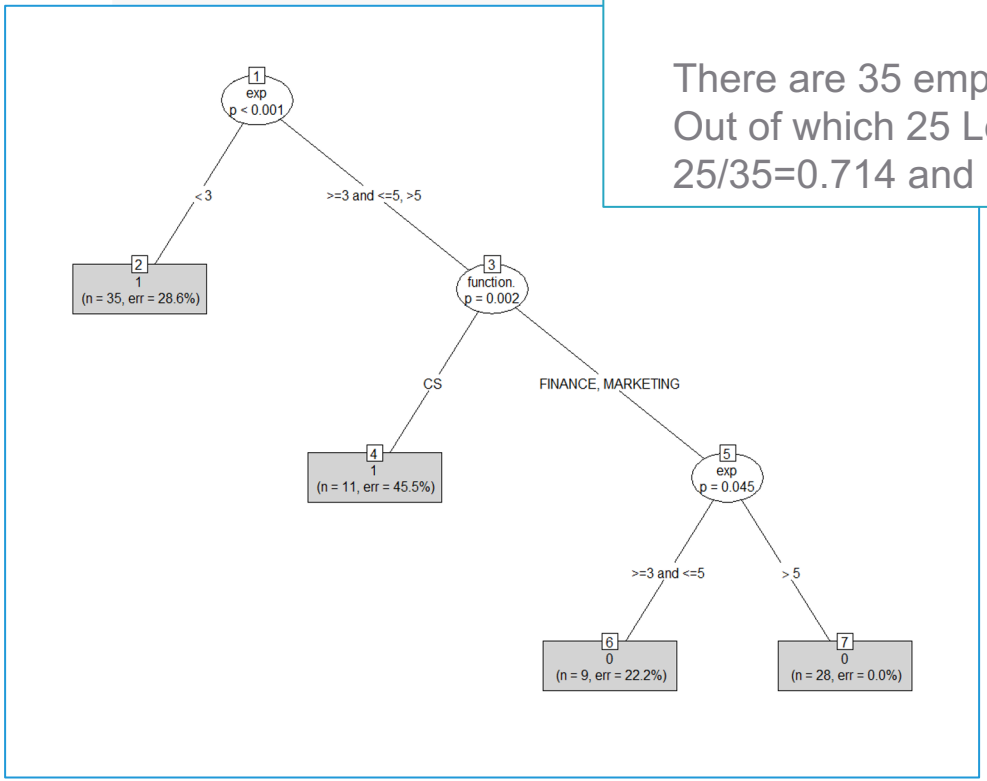
CHAID-like Implementation in Package “partykit”

```
# Decision Tree Using Package "partykit"
library(partykit)
empdata<-read.csv("EMPLOYEE CHURN DATA.csv",header=T)

empdata$status<-as.factor(empdata$status)
empdata$function.<-as.factor(empdata$function.)
empdata$exp<-as.factor(empdata$exp)
empdata$gender<-as.factor(empdata$gender)
empdata$source<-as.factor(empdata$source)

ctree<-
partykit::ctree(formula=status~function.+exp+gender+source
```

- We are instructing R to use the improved version of **ctree()** from package **“partykit”** by specifying **partykit::ctree()** in the command.
- formula= specifies dependent and independent variables

Decision Tree in Package “partykit”

```
plot(ctree,type="simple")
```



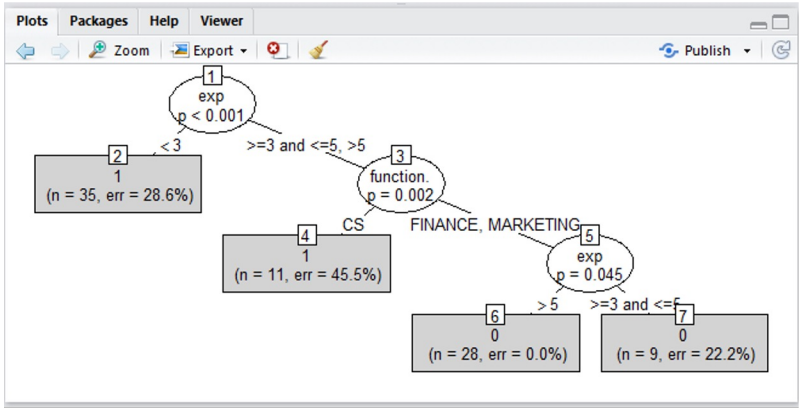
There are 35 employees with exp<3.
Out of which 25 Left within 18 months.
25/35=0.714 and Err=28.6%

Get an Edge!

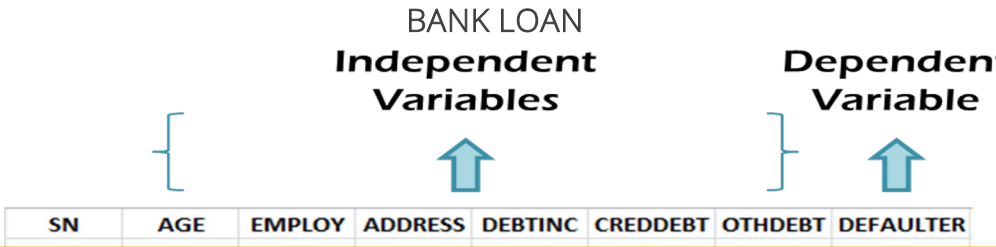
- In case of large data, default tree plot may end up looking congested and difficult to interpret. Adjust the aesthetics of the tree plot for better results. Add argument **gp** (graphical parameter) in the **plot()** function.

```
plot(ctree,type="simple", gp=gpar(cex=0.8))
```

We have used **gp=gpar()**from package **grid** to decrease the text size



Data Snapshot



Column	Description	Type	Measurement	Possible Values
SN	Serial Number		-	-
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	Continuous	-	Positive value
DEFAULTER	Whether customer defaulted on loan	Binary	1(Defaulter), 0(Non-Defaulter)	2

Decision Tree for Continuous & Categorical Independent Variables

```
# ctree() for Continuous Independent Variables

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

str(bankloan)
```

- **str()** is used to check the structure of all variables.
- We convert DEFAULTER and AGE to factor variables using **as.factor()** as in our data these 2 variables are categorical.

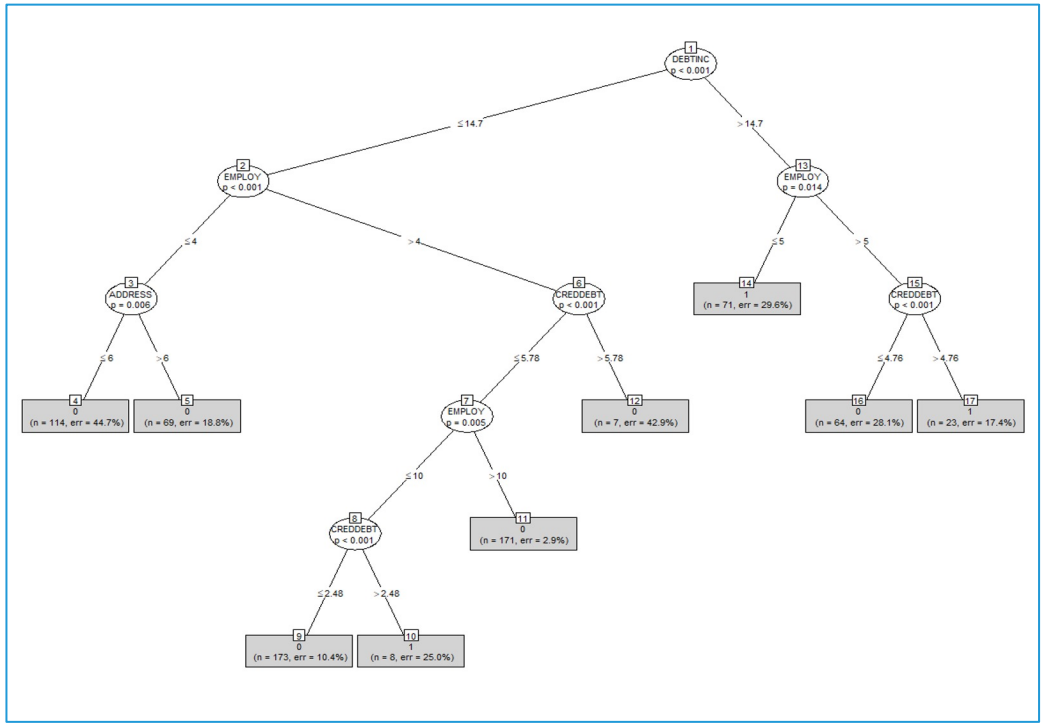
```
# Output
```

```
> str(bankloan)
'data.frame':   700 obs. of  8 variables:
 $ SN        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ AGE       : int  3 1 2 3 1 3 2 3 1 2 ...
 $ EMPLOY    : int  17 10 15 15 2 5 20 12 3 0 ...
 $ ADDRESS   : int  12 6 14 14 0 5 9 11 4 13 ...
 $ DEBTINC   : num  9.3 17.3 5.5 2.9 17.3 10.2 30.6 3.6 24.4 19.7 ...
 $ CREDDEBT  : num  11.36 1.36 0.86 2.66 1.79 ...
 $ OTHDEBT   : num  5.01 4 2.17 0.82 3.06 ...
 $ DEFAULTER: int  1 0 0 0 1 0 0 0 1 0 ...
```

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

bankctree<-partykit::ctree(DEFAULTER~AGE+EMPLOY+ADDRESS+DEBTINC+
                           CREDDEBT+OTHDEBT, data=bankloan)
```

Decision Tree for Continuous & Categorical Independent Variables

```
plot(bankctree,type="simple", gp=gpar(cex=0.7))
```



Interpretation

- AGE and OTHDEBT do not appear in the tree.
- 114 customers with DEBTIC >14.7, employed for ≤ 5 years are mainly DEFAULTERS

Quick Recap

CI Tree	• **`partykit::ctree()`** in package “partykit” yields conditional inference trees for continuous & categorical independent variables

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:

$$-\ P1 * \log_2(P1) -\ P2 * \log_2(P2)$$

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:

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

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

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

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

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

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

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

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)

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

Information Gain and ID3 Algorithm

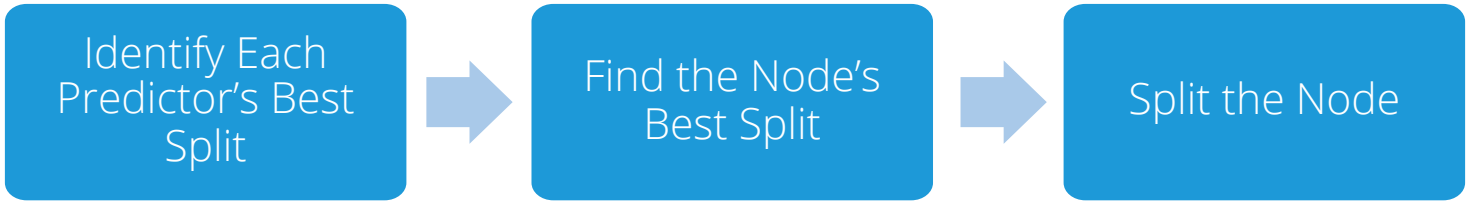
- Let us now go back to the basic ID3 algorithm; Step 1 of which is 'Identify the Best Attribute'

- 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

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

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.

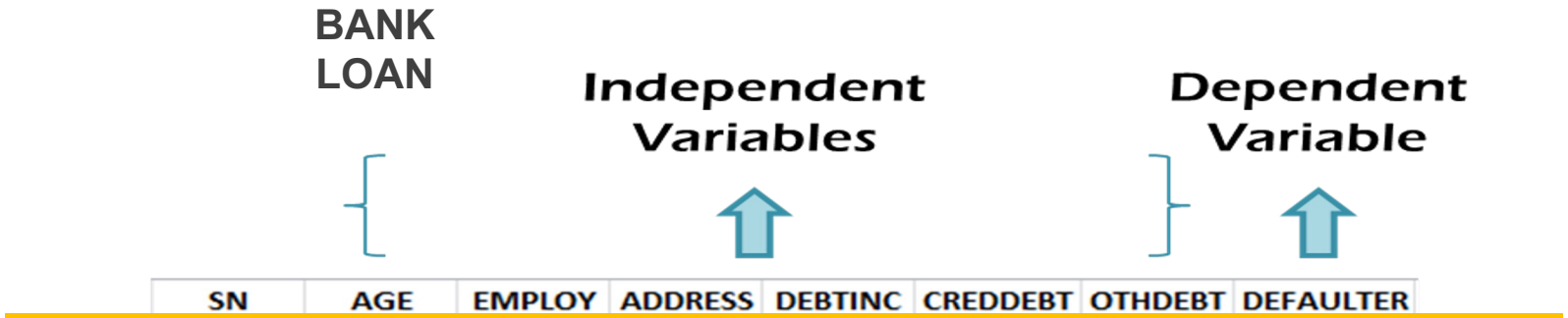
Objective

- 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")
```

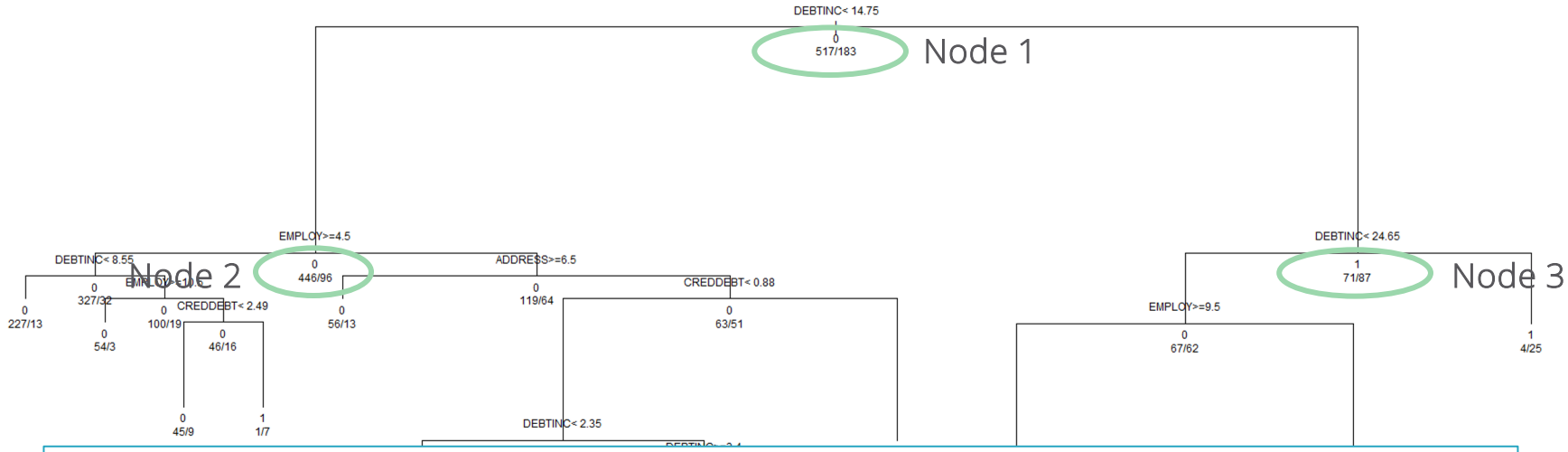
- **rpart()** fits an **rpart** model.
- First argument in the function is the formula (With no interaction terms)
- **data=** giving the data object.
- **method=** uses one of the four options **"anova", "poisson", "class"** or **"exp"**, depending on the type of the dependent variable. For classification 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 :

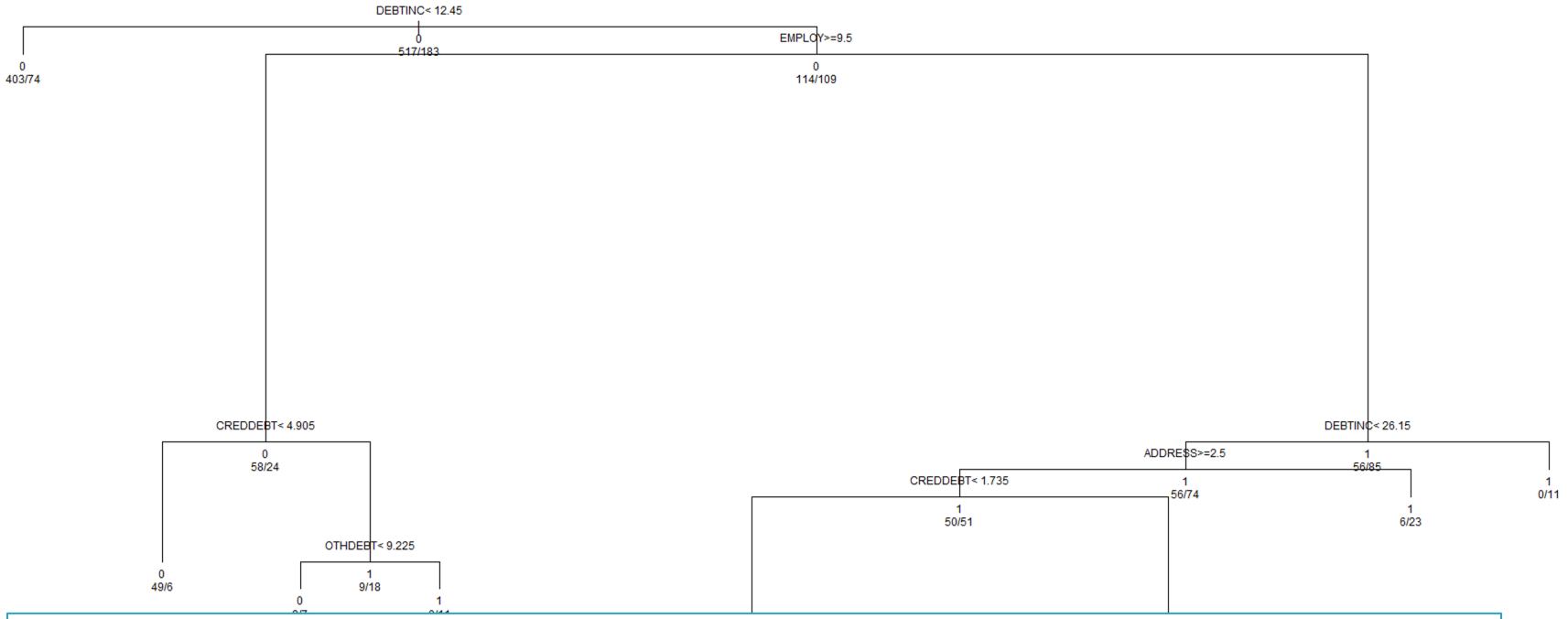
- Due to a large number of continuous predictors, a tree with several nodes and branches is generated.
- Tree starts with all 700 observations. 517 are non-defaulters (0) and the remaining 183 are defaulters (1).
- DEBTINC is the first split variable, left branch is <14.75 and right branch is >14.75. 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"))

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

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

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

vations

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

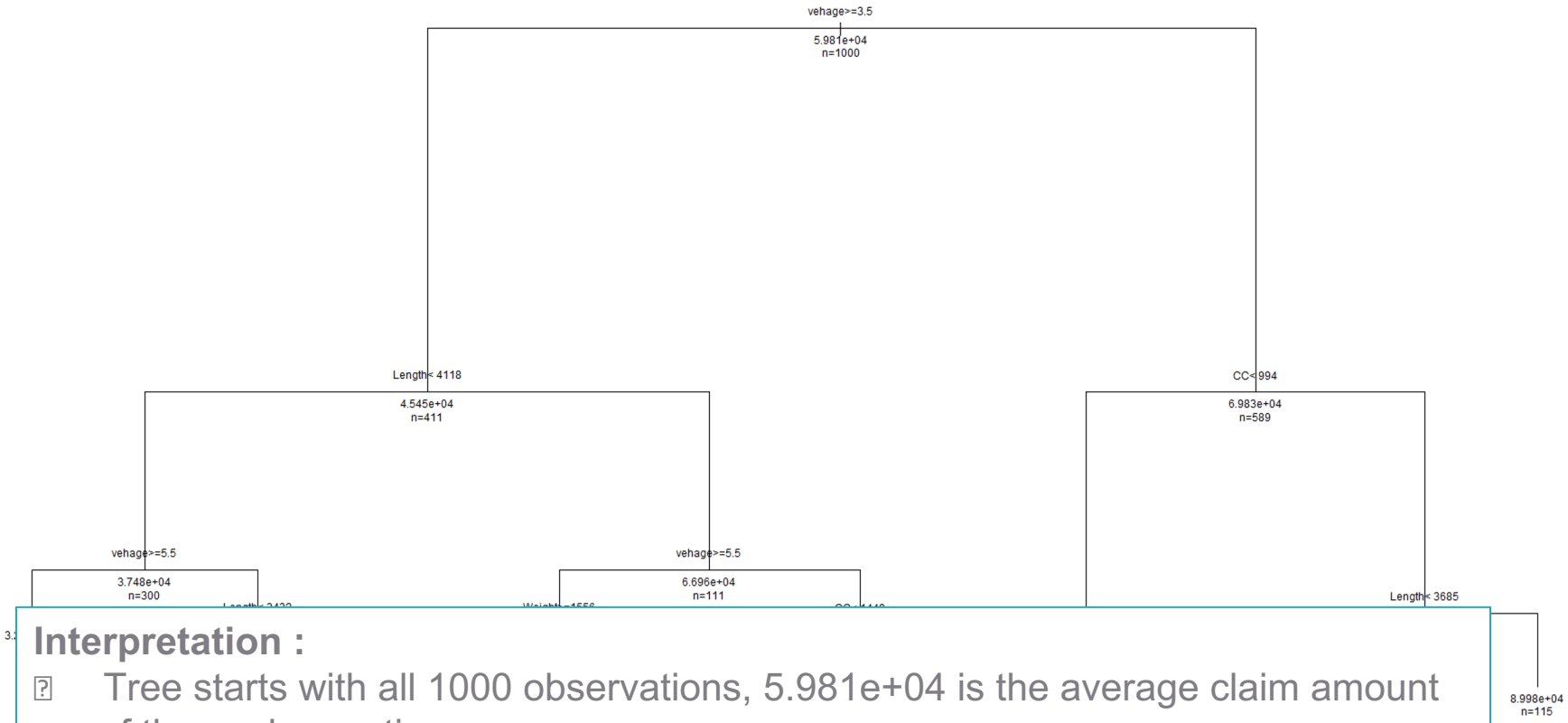
Columns	Description	Type	Measurement	Possible values
vehage	Age of the vehicle at the time of claim	integer	Years	positive values
CC	Engine capacity	numeric	cc	positive values
Length	Length of the vehicle	numeric	mm	positive values
Weight	Weight of the vehicle	numeric	kg	positive values
claimamt	Claim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

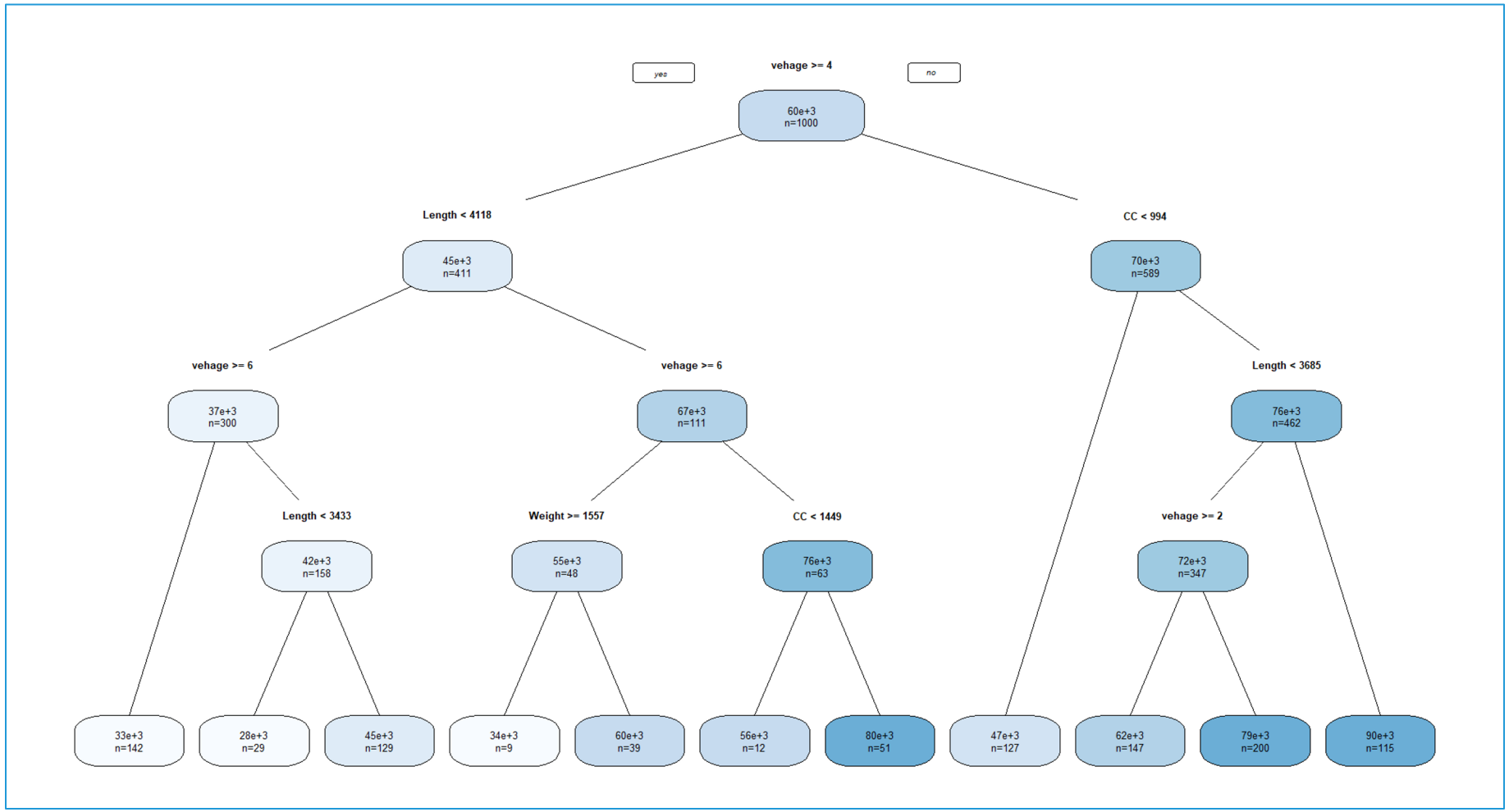
type= the function gives five possible types of plots. 0 is default, 1 labels all nodes, not just leaves

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
CC < 994
45e+3
n=411
70e+3
n=589
vehage >= 6
vehage >= 6
Length < 3685
37e+3
n=300
67e+3
n=111
76e+3
n=462
Length < 3433
Weight >= 1557
CC < 1449
vehage >= 2
42e+3
n=158
55e+3
n=48
76e+3
n=63
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

Random Forest Method I

Learn How Ensemble Learning Can be Used for Predictive Modeling

Contents

Bootstrapping

Bootstrapping is a method for estimating the sampling distribution of an estimator by resampling with replacement from the original sample

- The method is especially useful in situations where sampling distribution of estimator is not standard distribution.
- The method can be used in any statistical inference problem.
- The use of the term 'bootstrap' comes from the phrase “To pull oneself up by one's bootstraps ” - generally interpreted as succeeding in spite of limited resources.

Bootstrapping

- Many conventional statistical methods of analysis make assumptions about normality, including correlation, regression, t tests, and analysis of variance. When these assumptions are violated, such methods may fail.

- Bootstrapping, a data-based simulation method, is steadily becoming more popular as a statistical methodology. It is intended to simplify the calculation of statistical inferences, sometimes in situations where no analytical answer can be obtained.

- As computer processors become faster and more powerful, the time and effort required for bootstrapping decreases to levels where it becomes a viable alternative to standard parametric techniques.

Bootstrapping

Suppose the original sample is 12, 23, 11, 29, 34, 38, 41, 45, 6

Median = 29.00

We now draw multiple random samples using Bootstrapping

	Sample 1	Sample 2	Sample 3			Sample B
	23	11	6			41
	23	29	45			45
	29	11	11			11
	29	29	29			34
	34	34	11			34
	38	34	38			38
	41	41	41			41
	45	45	41			45
	41	11	6			6
Median	34.00	29.00	29.00			38.00

Sampling distribution of sample median is generated

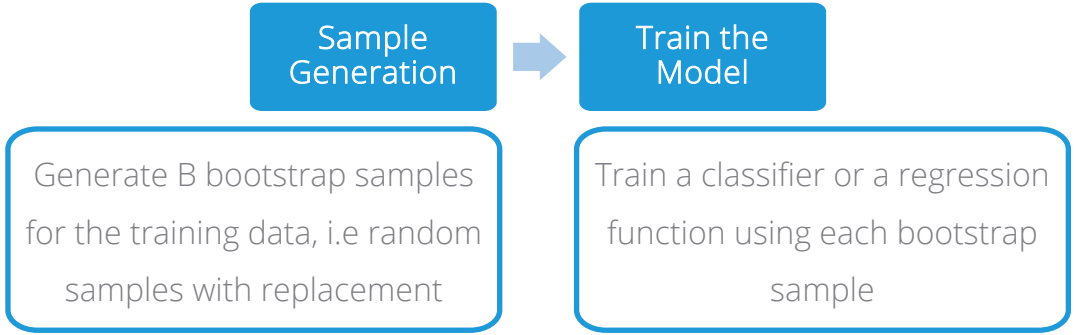
Assuming B=1000, **25th** value and **975th** value will provide 95% confidence interval for median

Bagging

- The term “Bagging” was Introduced by Breiman (1996).
- “Bagging” stands for “Bootstrap Aggregating”.
- It is an ensemble method: a method of combining results from multiple resamples.
- Ensemble method can also be applied by using different classifiers for a given sample.

Bagging Method Framework

Bagging has two basic steps:

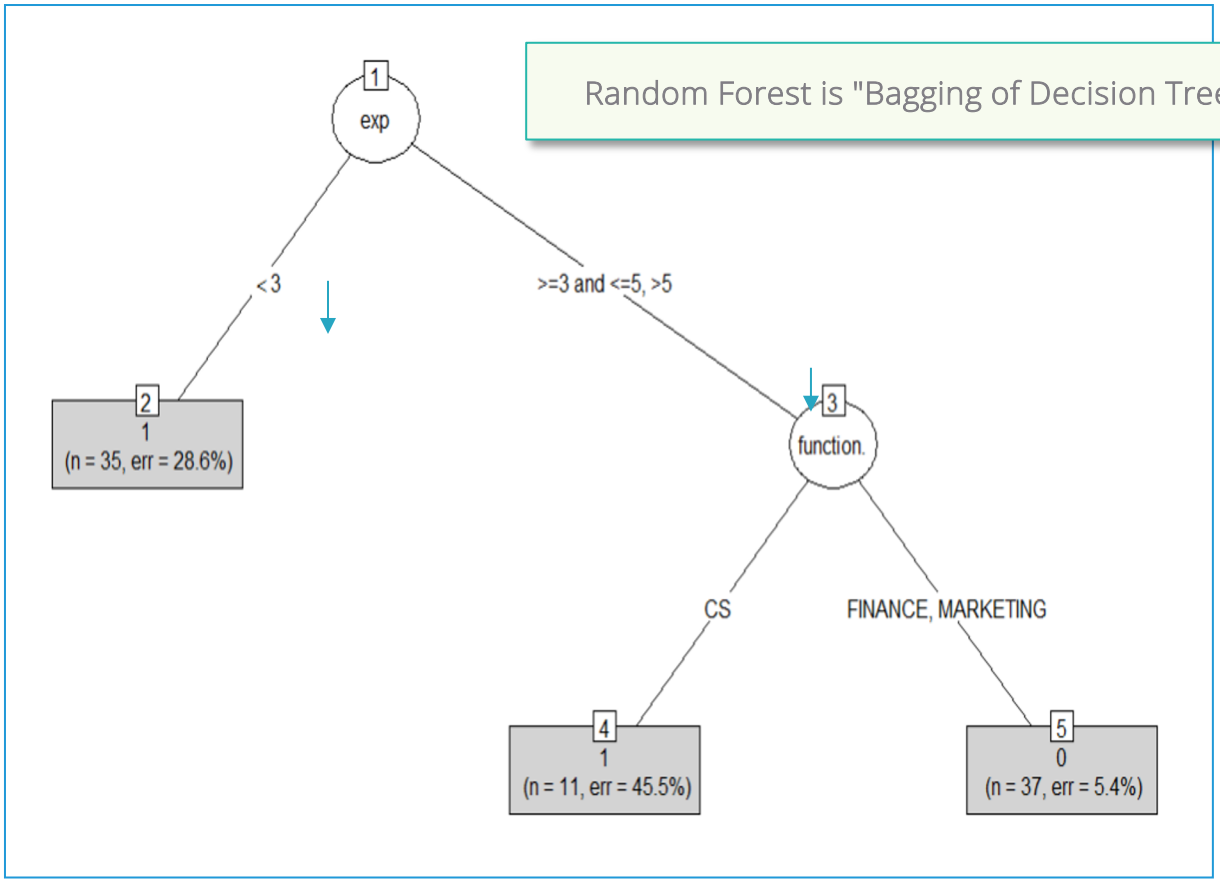


Model for **classification** Majority vote on the classification

Model for **regression** Average of the predicted value

Bagging improves performance for unstable classifiers which vary significantly with small changes in the data set

Re-look at the CHAID Decision Tree



Random Forest Classifier

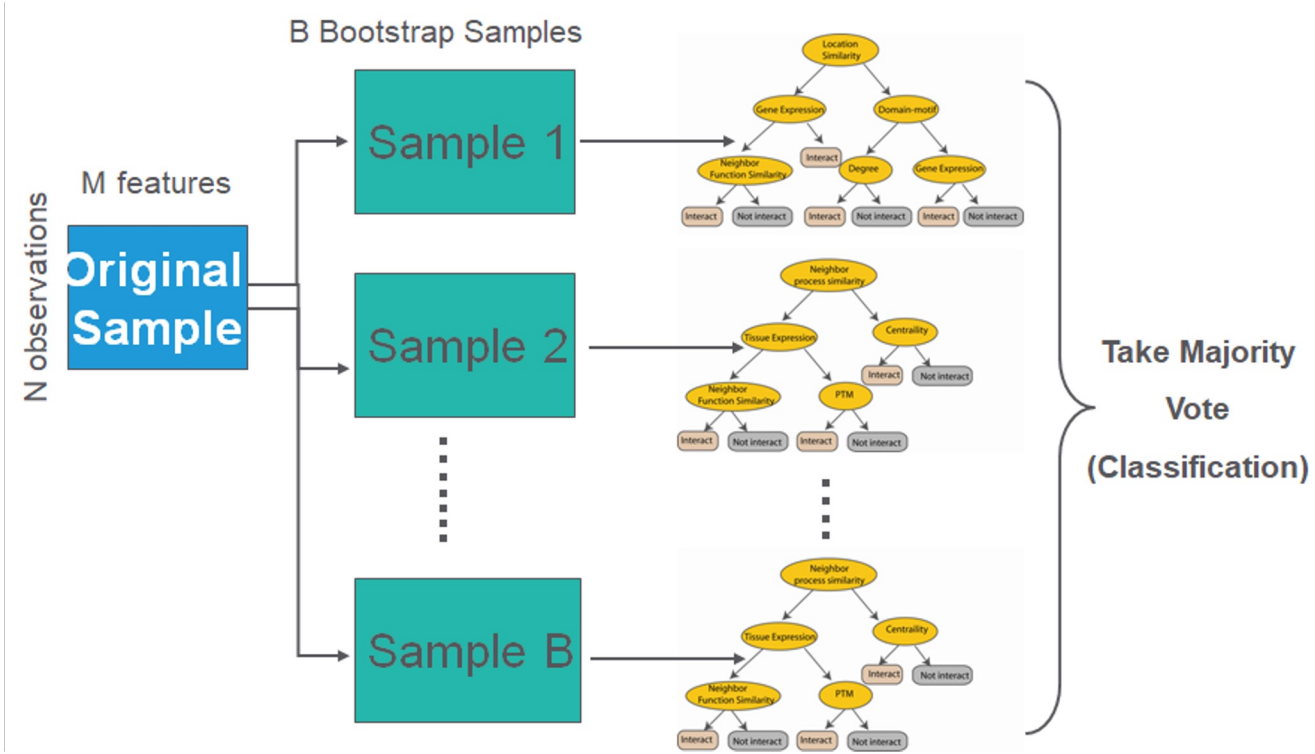
Random Forest (or random forests) is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees.

- The term came from random decision forests that was first proposed by Tin Kam Ho of Bell Labs in 1995.
- The method combines Breiman's “Bagging" idea and the random selection of features.

Random Forest Algorithm

1	Grow a forest of many trees (R default is 500)
2	Grow each tree on an independent **bootstrap sample*** from the training data
3	At each node: • Select *m* variables **at random** out of all *M* possible variables (independently for each node) • Find the best split on the selected *m* variables
4	Grow the trees to maximum depth
5	Vote/average the trees to get predictions for new data

Random Forest Algorithm



Random Forest Algorithm

- In bootstrap samples of data, approx. 2/3 of original values get included in each sample.
- Fit a tree to its greatest depth determining the split at each node through minimizing the loss function considering a random sample of covariates (size is user specified).
- For each tree,
  - Predict classification of the leftover 1/3 using the tree, and calculate the misclassification rate = *Out of Bag (OOB) Error Rate*
  - For each variable in the tree, permute the variables values and compute the OOB error, compare to the original OOB error, the increase is a indication of the variable's importance .

Random Forest Algorithm

- Aggregate OOB error and importance measures from all trees to determine overall OOB error rate and Variable Importance measure

OOB Error Rate

Calculate the overall percentage of misclassification

Variable Importance

Average increase in OOB error over all trees

Quick Recap

Bootstrapping	• Method for estimating the sampling distribution of an estimator by resampling with replacement from the original sample
Bagging	• "Bagging" stands for "Bootstrap Aggregating" • It is an ensemble method: a method of combining results from multiple resamples
Random Forest Method	• Its an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees • Random forests also work for regression problems • The method combines Breiman's "Bagging" idea and the random selection of features

Random Forest Method II

Learn How Ensemble Learning Can be Used for Predictive Modeling

Contents

Case Study – Employee Churn Model

Background

- A company has comprehensive database of its past and present workforce, with information on their demographics, education, experience and hiring background as well as their work profile. The management wishes to see if this data can be used for predictive analysis, to control attrition levels.

Objective

- To develop an Employee Churn model via Random Forest Method

Available Information

- Sample size is 83
- **Gender, Experience Level** (<3, 3-5 and >5 years), **Function** (Marketing, Finance, Client Servicing (CS)) **and Source** (Internal or External) are independent variables
- **Status** is the dependent variable (=1 if employee left within 18 months from joining date)

Data Snapshot

EMPLOYEE CHURN DATA

Dependent Variable

Independent Variables

sn	status	function	exp	gender	source
1	1	CS	<3	M	external

Columns	Description	Type	Measurement	Possible values
sn	Serial Number	-	-	-
status	= 1 If the Employee Left Within 18 Months of Joining = 0 Otherwise	Integer	1,0	2
function	Employee Job Profile	Character	CS, FINANCE, MARKETING	3
exp	Experience in Years	Character	<3,3-5,>5	3
gender	Gender of the Employee	Character	M,F	2
source	Whether the Employee was Appointed via Internal or External Links	Character	external, internal	2

Random Forest in R

```
# Installing Package, Importing and Readying the Data

install.packages("randomForest")
library(randomForest)

empdata<-read.csv("EMPLOYEE CHURN DATA.csv",header=T)

empdata$status<-as.factor(empdata$status)
```

Since it's a classification problem, dependent variable is converted to factor variable using **as.factor()**.

Random Forest in R

```
# Run Random Forest
churn_rf<-randomForest(status~function.+exp+gender+source,
data=empdata,
                       mtry=2,ntree=100,importance=TRUE,
cutoff=c(0.6,0.4))
```

- **randomForest()** implements Breiman's random forest algorithm, for classification and regression.
- The first argument in the function is **formula=** describing the model to be fitted. It can also take **x**, data frame or matrix of predictors.
- **data=** gives the data object.
- **mtry=** Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in x) and regression (p/3).
- **ntree=** Specifies the number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
- **importance=** logical, (default is FALSE) tells R whether variable importance is to be assessed or not.
- **cutoff=** This argument is specific to classification only. A vector of length equal to number of classes. The 'winning' class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is 1/k where k is the number of classes (i.e. Majority vote wins).

Random Forest in R – Output

```
# Output
churn_rf
```

```
> churn_rf

Call:
 randomForest(formula = status ~ function. + exp + gende
(0.6, 0.4))
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 2

        OOB estimate of  error rate: 28.92%
Confusion matrix:
   0  1 class.error
0 36 14   0.2800000
1 10 23   0.3030303
```

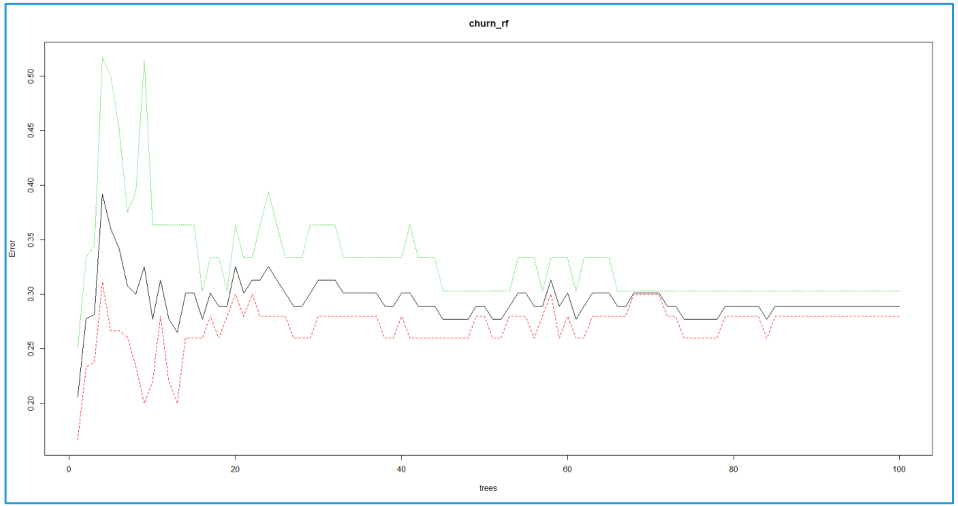
Interpretation :
Model calculates the OOB error.

Note : Since the samples are generated randomly, the outputs will vary slightly for different devices.

Random Forest in R

```
# Decision Trees Error Rate
plot(churn_rf)

# Output
```

plot() of a **randomForest()** object returns a plot of the error rates or MSE of the object.



Interpretation :

- Plot shows error rates for all 100 decision trees.
- Black line shows the overall OOB error rate.
- Coloured lines show error rates for each class.

* In case of regression, the plot shows only the black line, overall OOB MSE error.

Random Forest in R – Prediction

```
# Adding Predictions as a new column to original data
empdata$pred <- predict(churn_rf,empdata)
head(empdata)
```

```
# Output
```

	sn	status	function.	exp	gender	source	pred
1	1	1	CS	<3	M	external	1
2	2	1	CS	<3	M	external	1
3	3	1	CS	>=3 and <=5	M	internal	0
4	4	1	CS	>=3 and <=5	F	internal	0
5	5	1	CS	<3	M	internal	1
6	6	1	CS	>5	M	external	1

Random Forest in R – Variable Importance

```
# Importance Matrix
churn_rf$importance
```

randomForest() object contains variable importance matrix, if **importance=T** in the function.

```
# Output
```

	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
function.	0.06742038	0.04707436	0.056724526	7.391617
exp	0.12162621	0.10965367	0.113936498	12.987221
gender	0.01061312	-0.01354428	0.001747229	1.989162
source	0.02297498	-0.01021166	0.009771908	2.586050

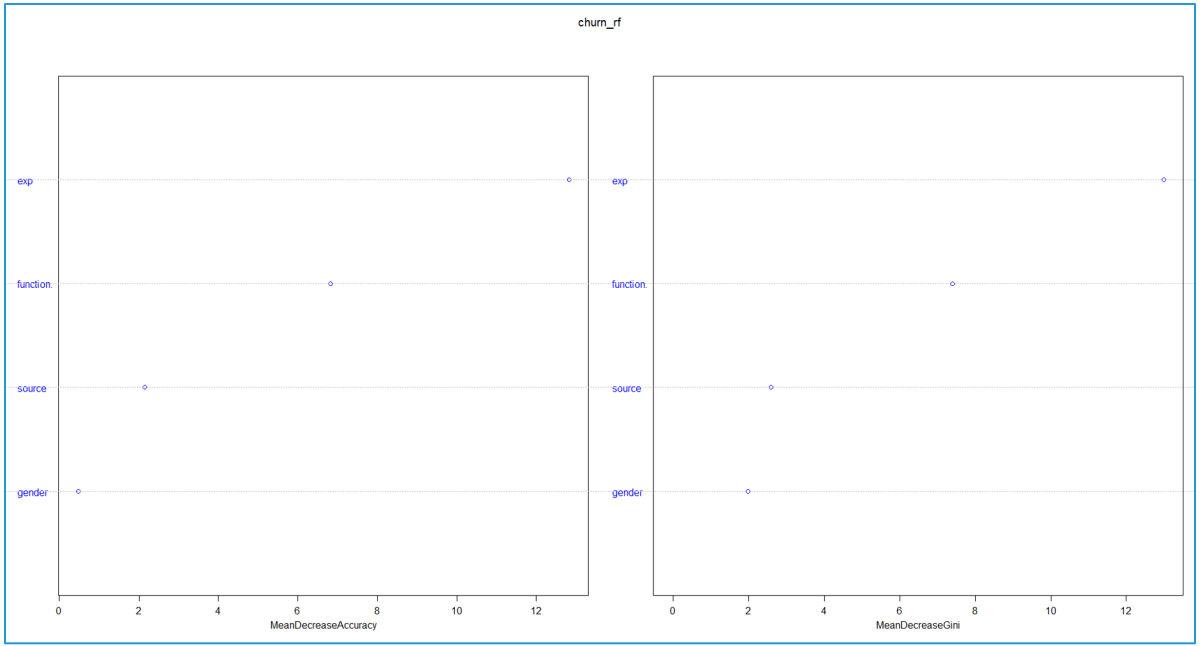
- Experience has highest importance.
- The first two columns are the class-specific measures computed as mean decrease in accuracy.
- The next column is the mean decrease in accuracy over all classes & the mean decrease in Gini index.

```
# Variable Importance Plot
varImpPlot(churn_rf, col="blue")
```

varImpPlot() in package **"randomForest"** returns a dot chart of variable importance measured by a random forest.

Random Forest in R – Variable Importance

```
# Output
```



Quick Recap

Random Forest Method	• Its an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees • Random forests also work for regression problems • The method combines Breiman's “Bagging" idea and the random selection of features
Random Forest in R	• `randomForest()` in package “`randomForest`” runs random forest analysis • The output can generate variable importance and confusion matrix

Market Basket Analysis - I

Contents

About Association Rules

Association Rule Learning

Method for discovering interesting relations between variables in large databases

- Based on the **concept of strong rules**, Rakesh Agrawal introduced association rules for **discovering regularities between products in large-scale transaction data recorded by point-of-sale (POS) systems in supermarkets**
- For example, the rule found in the sales data of a supermarket would indicate that **if a customer buys onions and potatoes together, they are also likely to buy burger**
- Association rule learning method can be applied in many areas such as **web usage mining, fraud detection, continuous production and bioinformatics**

Introduction to Market Basket Analysis

- The most widely used area of application for association rules is **Market Basket Analysis**

> Market Basket Analysis (Association Analysis) is a **mathematical modeling technique based upon the theory that if you buy a certain group of items, you are likely to buy another group of items**

- It is used to **analyze the customer purchasing behavior and helps in increasing the sales and maintain inventory** by focusing on the point of sale transaction data

Market Basket Analysis – Uses

Product Building	• Develop combo offers based on products bought together
Optimisation	• Organise and place associated products/categories nearby inside a store
Advertising and Marketing	• Determine the layout of the catalog of an ecommerce site
Inventory Management	• Control inventory based on product demands and what products sell together

Definitions and Terminology

Term	Definition
Transactions	A set of items (Item set)
Support	Ratio of **number of times two or more items occur together** to **the total number of transactions** Support can be thought of as P(A and B)
Confidence	Conditional probability that **a randomly selected transaction will include Item B given Item A** **P(B|A)** (written as A => B)
Lift	Ratio of the **probability of Items A and B occurring together (Joint probability)** to **the product of P(A) and P(B)**

Get an Edge!

The Famous Story

An article in The Financial Times of London (Feb. 7, 1996) stated,

"The example of what data mining can achieve is the case of a large US supermarket chain which discovered a strong association for many customers between a brand of babies nappies (diapers) and a brand of beer. Most customers who bought the nappies also bought the beer. The best hypothesisers in the world would find it difficult to propose this combination but data mining showed it existed, and the retail outlet was able to exploit it by moving the products closer together on the shelves."

Rule Evaluation – Support

Transaction No.	Item 1	Item 2	Item 3	...
100	Beer	Diaper	Chocolate	
101	Milk	Chocolate	Shampoo	
102	Beer	Wine	Vodka	
103	Beer	Cheese	Diaper	
104	Ice Cream	Diaper	Beer	

A B

Support of {Diaper, Beer}

$$\text{Support} = \frac{\text{No.of transactions containing both A and B}}{\text{Total no.of transactions}} = \frac{3}{5} = 60\%$$

Support of {Diaper, Beer} is 3/5

Rule Evaluation – Confidence

Transaction No.	Item 1	Item 2	Item 3	…
100	Beer	Diaper	Chocolate	
101	Milk	Chocolate	Shampoo	
102	Beer	Wine	Vodka	
103	Beer	Cheese	Diaper	
104	Ice Cream	Diaper	Beer	

$$\text{Confidence for } \{A\} \Rightarrow \{B\} = \frac{\text{No. of transactions containing both A and B}}{\text{No. of transactions containing A}}$$

Confidence for $\{Diaper\} \Rightarrow \{Beer\}$ is 3/3

When Diaper is purchased, the likelihood of Beer purchase is 100%

Confidence for $\{Beer\} \Rightarrow \{Diaper\}$ is 3/4

When Beer is purchased, the likelihood of Diaper purchase is 75%

$\{Diaper\} \Rightarrow \{Beer\}$ is a more important rule according to Confidence

Rule Evaluation – Lift

Transaction No.	Item 1	Item 2	Item 3	Item 4
100	Beer	Diaper	Chocolate	
101	Milk	Chocolate	Shampoo	
102	Beer	Milk	Vodka	Chocolate
103	Beer	Milk	Diaper	Chocolate
104	Milk	Diaper	Beer	

A B

Consider {Chocolate} ⇒ {Milk}

$$\text{Lift} = \frac{P(A \cap B)}{P(A)P(B)} = \frac{3/5}{\left(4/5\right)\left(4/5\right)} = 0.9375$$

Lift < 1 indicates Chocolate is decreasing the chance of Milk purchase
Support and confidence are high but lift is low

Case Study – Groceries Purchase Data

Background

- A typical grocery outlet records point-of-sale transaction data

Objective

- To mine association rules and information about item sets

Available Information

- Total number of transactions is 9835
- Items are aggregated to 169 categories
- Data is collected for 1 month (30-days)

Data Snapshot

Groceries

```
    items
[1] {citrus fruit,
     semi-finished bread,
     margarine,
     ready soups}
[2] {tropical fruit,
     yogurt,
     coffee}
[3] {whole milk}
[4] {pip fruit,
     yogurt,
     cream cheese ,
     meat spreads}
[5] {other vegetables,
```

Columns	Description	Possible values
id	Transaction Id	Positive Integers
items	Set of Items purchased in a transaction	Subset from 169 categories of items

Market Basket Analysis in R

```
#Market Basket Analysis Using Apriori Recommendation

install.packages("arules")
library(arules)

install.packages("arulesViz")
library(arulesViz)



data("Groceries")
```

We will be using two packages for performing Market Basket Analysis in R.
Package **"arules"** stands for 'Association Rules' and it contains functions for mining association rules and frequent itemsets.
Package **"arulesViz"** is used for visualisation.
Install and load these two packages.

- Load the dataset.
- The **Groceries** data set is provided for package **arules** by Michael Hahsler, Kurt Hornik and Thomas Reutterer.*
  The data is of class 'transaction' supported by package **arules**.

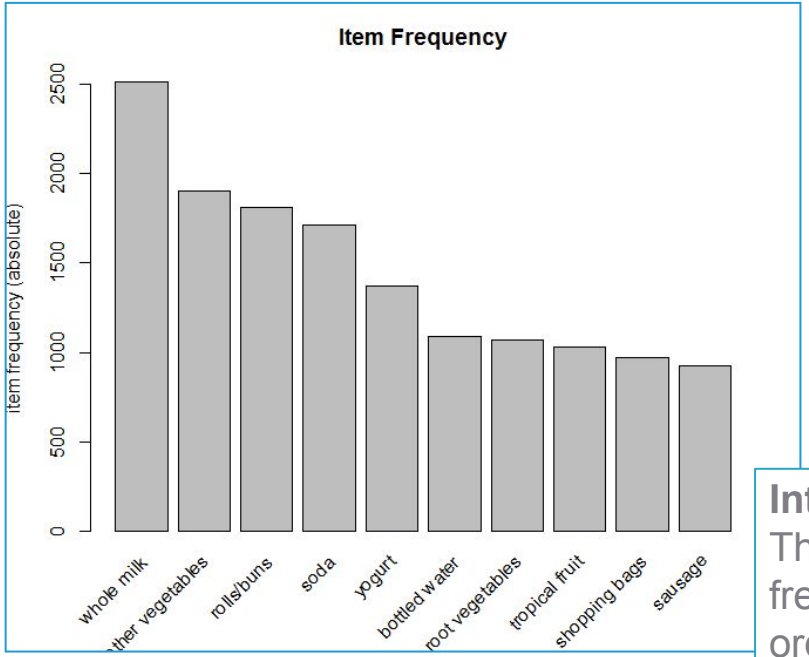
Visualise Item Frequency

```
#Item Frequency Plot
itemFrequencyPlot(Groceries,topN=10,type="absolute",
                  main="Item Frequency")
```

itemFrequencyPlot() calculates item frequency and returns a barplot.
topN= instructs R to plot only top N highest item frequency or lift (Logical, if **lift=TRUE**). It plots values in decreasing order.
type= is a character string indicating whether item frequencies should be displayed relative or absolute. Default is relative.

Item Frequency Plot

```
# Output
```

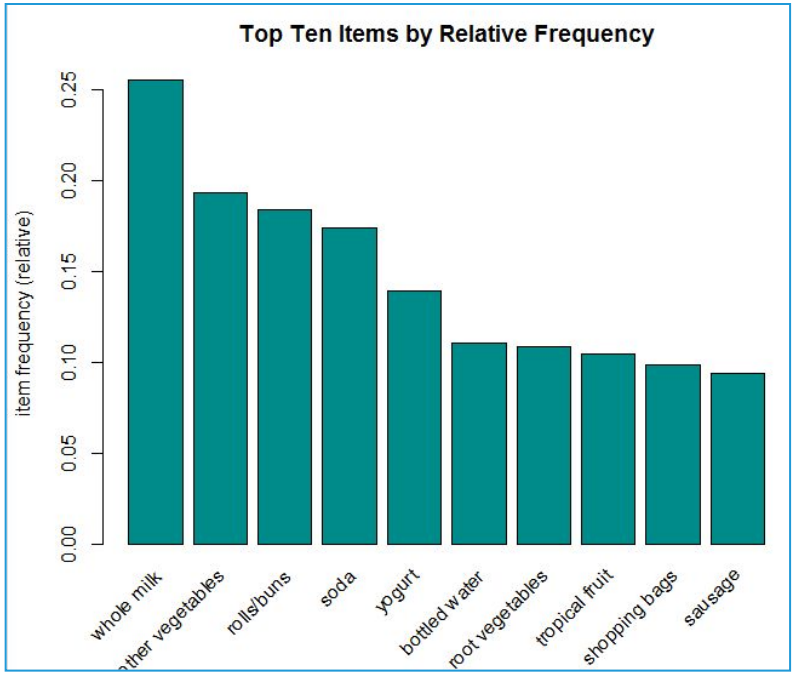


Interpretation:
The plot shows items by frequency in a descending order.

Item Frequency Plot

```
itemFrequencyPlot(Groceries,topN=10,type="relative",
col="darkcyan",main="Top Ten Items by Relative Frequency")
```

```
# Output
```



- **type= "relative"** displays barplot with the relative frequency
- **col=** specifies the colour of the bars

Interpretation:

- The plot shows items by relative frequency in a descending order.

Get and Display the Rules

```
#Get the Rules
rules<-apriori(Groceries,parameter=list(supp=0.001,conf=0.8))
```

- The Apriori algorithm employs level-wise search for frequent itemsets.
- **apriori()** is used to mine frequent itemsets, association rules or association hyperedges using this algorithm.
- The default is to mine rules with **support 0.1, confidence 0.8.**
- **Here, we have used threshold of 0.001 for support.**
- **apriori()** returns an object of class rules or itemsets.

```
#Show Top 5 Rules But Only 2 Digits
options(digits=2)
```

options in base R allows the user to set global options which affect the way in which R computes and displays results. We have set **digits=2** to display results with only 2 digits.

```
inspect(rules[1:5])
```

inspect in package **arules** displays association and plus additional information formatted for online inspection.

Get and Display the Rules

```
# Output of Rules
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5   0.001      1
 maxlen target   ext
     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.03s].
writing ... [410 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Interpretation:

- The output displays parameter specification, algorithmic control and absolute minimum support count.
- It also lists down tasks performed and time taken to complete them.
- We are interested in knowing how many rules were created; 410 in our case.

Get and Display the Rules

```
# Output of inspect
```

	lhs		rhs	support	confidence	lift	count
[1]	{liquor,red/blush wine}	=>	{bottled beer}	0.0019	0.90	11.2	19
[2]	{curd,cereals}	=>	{whole milk}	0.0010	0.91	3.6	10
[3]	{yogurt,cereals}	=>	{whole milk}	0.0017	0.81	3.2	17
[4]	{butter,jam}	=>	{whole milk}	0.0010	0.83	3.3	10
[5]	{soups,bottled beer}	=>	{whole milk}	0.0011	0.92	3.6	11

Interpretation:

- **inspect()** returns list of lhs and rhs items, their support, confidence and lift values.

Manage How the Rules are Displayed

```
#Sort the Rules
rules<-sort(rules,by="lift",decreasing=TRUE)
```

- sort() from package arules is used
- by="lift" indicates sort by values of Lift
- decreasing= logical, specifies the direction of sorting. Default is decreasing=TRUE.

```
#Show Top 5 Rules (Sorted)

options(digits=2)

inspect(rules[1:5])
```

Top Five Rules (Sorted)

```
# Output
```

	lhs		rhs	support	confidence	lift	count
[1]	{liquor, red/blush wine}	=>	{bottled beer}	0.0019	0.90	11.2	19
[2]	{citrus fruit, other vegetables, soda, fruit/vegetable juice}	=>	{root vegetables}	0.0010	0.91	8.3	10
[3]	{tropical fruit, other vegetables, whole milk, yogurt, oil}	=>	{root vegetables}	0.0010	0.91	8.3	10
[4]	{citrus fruit, grapes, fruit/vegetable juice}	=>	{tropical fruit}	0.0011	0.85	8.1	11
[5]	{other vegetables, whole milk, yogurt, rice}	=>	{root vegetables}	0.0013	0.87	8.0	13

Interpretation:

- The rules are now sorted based on lift. Sorting ensures that most relevant rules appear first.

Quick Recap

In this session, we learnt **Market Basket Analysis:**

Market Basket Analysis

- Mathematical modeling technique based upon the theory that if you buy a certain group of items, you are likely to buy another group of items
- Transactions, Support, Confidence and Lift are the key concepts used in this analysis
- The analysis is performed by creating and studying rules based on different itemsets

Market Basket Analysis in R

- Package **`arules`** and **`arulesViz`** are used for undertaking MBA
- **`itemFrequencyPlot()`** plots frequency
- **`apriori()`** function creates rules. **`inspect()`** displays association and additional information
- **`plot()`** in **`arulesViz`** can create static or interactive plots

Market Basket Analysis - II

Contents

Targeting Items

Association rules should be used to explain decision making and further utilised to form effective strategies.

Continuing with the example of consumer's buying preferences, the following two questions can be of interest. Reference item is Whole Milk.

What are customers likely to buy if they purchase whole milk?

Targeting Items

```
library(arules)
data("Groceries")

rules<-apriori(Groceries,parameter=list(supp=0.001,conf=
0.15,minlen=2),appearance=list(default="rhs",lhs="whole
milk"),control=list(verbose=FALSE))

rules<-sort(rules,decreasing=TRUE,by="confidence")

inspect(rules[1:5])
```

- We have already seen the basic arguments used in **apriori().**
- **minlen=** in **parameter=** is used to specify how many items to be considered

  Default is **minlen=1** which means that rules with only one item will be created.
- **appearance=** is used to restrict item appearance.
- **control=** controls the algorithmic performance of mining algorithm.
- **verbose=FALSE** ensures R does not show progress report.

* Here we continue with previous ppt & use the same inbuilt dataset “Groceries”

Targeting Items

```
# Output
```

	lhs		rhs	support	confidence	lift	count
[1]	{whole milk}	=>	{other vegetables}	0.075	0.29	1.5	736
[2]	{whole milk}	=>	{rolls/buns}	0.057	0.22	1.2	557
[3]	{whole milk}	=>	{yogurt}	0.056	0.22	1.6	551
[4]	{whole milk}	=>	{root vegetables}	0.049	0.19	1.8	481
[5]	{whole milk}	=>	{tropical fruit}	0.042	0.17	1.6	416

Interpretation:

- Based on confidence, customers are most likely to move to other vegetables immediately after buying whole milk.

Visualise Rules

```
#Creating Interactive Graph

library(arulesViz)
rules<-apriori(Groceries,parameter=list(supp=0.001,conf=0.15,minlen=2)
,
appearance=list(default="rhs",lhs="whole
milk"),control=list(verbose=FALSE))

plot(rules,method="graph",interactive=TRUE,shading=NA)
```

Interactive Graphs

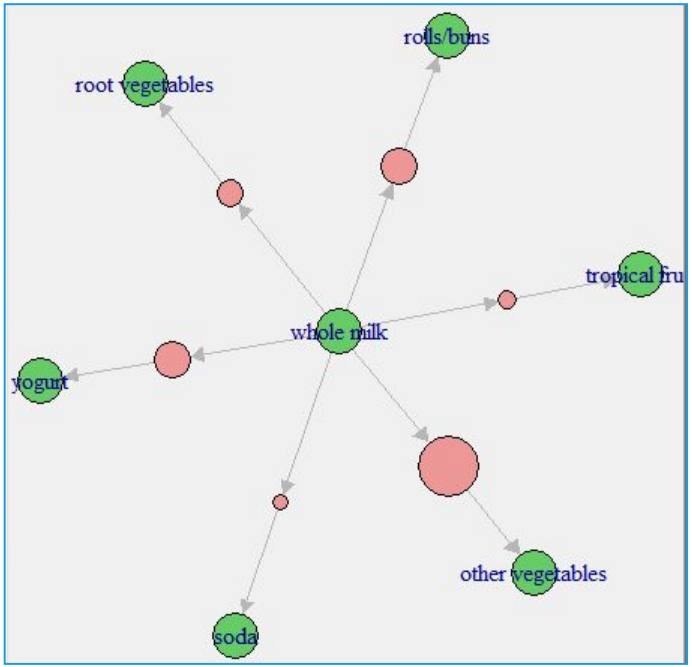
Upon running the command for interactive graph in **`arulesViz`**, a new window opens

Graph uses vertices and edges to visualise rules

- Vertices are itemsets or items
- Edges indicate relationship in rules
- Labels on the edges or colour or width of the arrows displaying edges represent interest measures

Note that graph-based visualisations are viable only for a small set of rules, as more number of rules would make the graph cluttered and difficult to interpret

Interpreting the Interactive Graph



Out target item is whole milk, which is at the centre of the graph

- The coloured vertex represent confidence. As seen before, confidence for {whole milk, other vegetables} is the highest, followed by yogurt and rolls/buns.
- Lowest confidence in the top five is for {whole milk, tropical fruit}

Case Study

Background

- Transactions data collected from point of sales is generally in long format. Arules package requires the data to be in wide format.

Objective

- To convert available data to a format suitable for association analysis and conduct analysis via arules package in R

Available Information

- Each transaction is given a unique ID
- Items basket contains five items, items purchased during each transaction are recorded

Data Snapshot

Transactions Data for MBA

id	item
1	B
1	C
1	D
1	E
2	A
2	B
2	C
2	D
2	E
3	A
3	B

Columns	Description	Measurement	Possible values
id	Transaction Id	-	Positive Integers
item	Items purchased	A,B,C,D,E	5

Data Conversion

```
#Convert the Data
trans<-read.transactions("Transactions Data for MBA.csv",format="single",sep=",",cols=c("id","item"),header=TRUE)
```

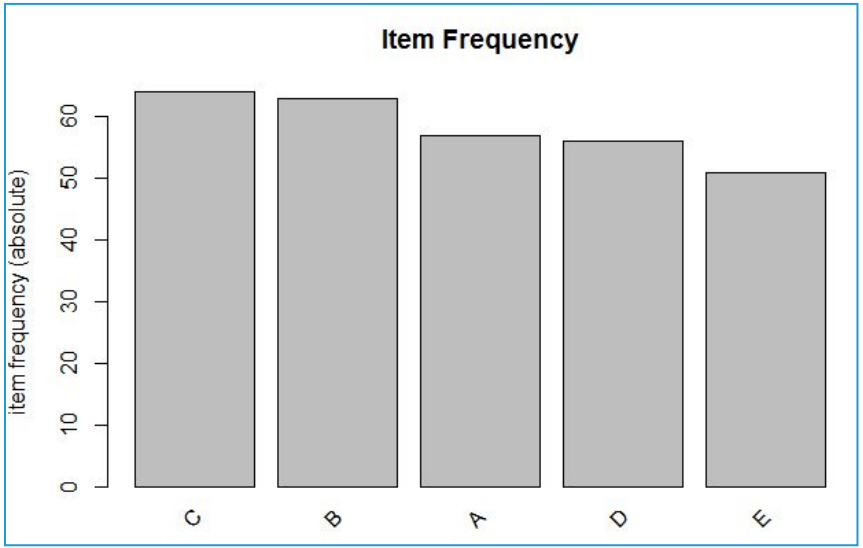
read.transactions() in package arules reads a transactions data file and creates a transaction object.
format= indicates the format of the dataset. **"single"** implies each line corresponds to a single item, containing at least ids for the transaction and the item. **"basket"** implies each line in the transaction data file represents a transaction where the items (item labels) are separated by the characters specified by **sep**.
For the **"single"** format, **cols=** is a numeric or character vector of length two giving the numbers or names of the columns (fields) with the transaction and item ids, respectively.
sep= is a character string specifying how fields are separated in the data file. The default ("") splits at whitespaces.

Association Analysis

```
#Visualise Frequency
itemFrequencyPlot(trans,topN=5,type="absolute")
```

itemFrequencyPlot() calculates item frequency and returns a barplot.
topN= instructs R to plot only top N highest item frequency or lift
type= is a character string indicating whether item frequencies should be displayed relative of absolute.

```
# Output
```



Interpretation:

- The plot shows items by frequency in a descending order.

Association Analysis

```
#Get the Rules

rules<-apriori(trans,parameter=list(supp=0.001,conf=0.8))

inspect(rules[1:5])
```

apriori() is used to mine frequent itemsets, association rules or association hyperedges using this algorithm with specified support and confidence
inspect() in package **arules** displays association and additional information formatted for online inspection

Association Analysis

```
# Output
```

```
Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support
        0.8    0.1    1 none FALSE            TRUE       5   0.001
 maxlen target   ext
     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 0

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[5 item(s), 100 transaction(s)] done [0.00s].
sorting and recoding items ... [5 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 done [0.00s].
writing ... [5 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Interpretation:

The output displays parameter specification, algorithmic control and absolute minimum support count.

It also lists down tasks performed and time taken to complete them.

We are interested in knowing how many rules are created; Here 5 rules are created.

Association Analysis

```
# Output
```

	lhs		rhs	support	confidence	lift	count
[1]	{A,D}	=>	{C}	0.25	0.81	1.3	25
[2]	{A,D,E}	=>	{C}	0.11	0.85	1.3	11
[3]	{A,B,E}	=>	{C}	0.10	0.83	1.3	10
[4]	{A,B,D}	=>	{C}	0.16	0.84	1.3	16
[5]	{A,B,D,E}	=>	{C}	0.06	1.00	1.6	6

Interpretation:

- **inspect()** returns list of lhs and rhs items, their support, confidence and lift values

Quick Recap

In this session, we learnt **Market Basket Analysis:**

Market Basket Analysis

- Mathematical modeling technique based upon the theory that if you buy a certain group of items, you are likely to buy another group of items
- Transactions, Support, Confidence and Lift are the key concepts used in this analysis
- The analysis is performed by creating and studying rules based on different itemsets

Market Basket Analysis in R

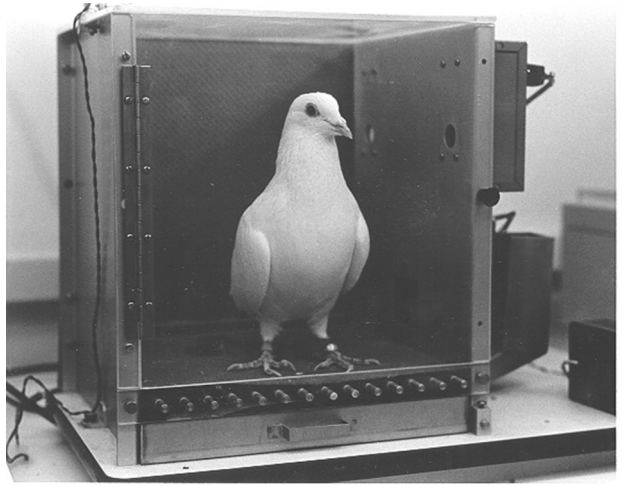
- Package **`arules`** and **`arulesViz`** are used for undertaking MBA
- **`itemFrequencyPlot()`** plots frequency
- **`apriori()`** function creates rules. **`inspect()`** displays association and additional information
- **`read.transactions()`** converts raw point of sales transactions data to a wide-format transactions object
- **`plot()`** in **`arulesViz`** can create static or interactive plots

Introduction to Artificial Neural Networks (R)

Contents

Pigeons as Art Experts

- Experiment:
- Pigeon in Skinner box
- Present paintings of two different artists (e.g. Chagall / Van Gogh)
- Reward for pecking when presented a particular artist (e.g. Van Gogh)



- Pigeons were able to discriminate between Van Gogh and Chagall with 95% accuracy (When presented with pictures they had been trained on)
- Discrimination still 85% successful for previously unseen paintings of the artists.

Pigeons as Art Experts

Pigeons do not simply memorize the pictures

They can extract and recognise patterns (the ‘style’)

They generalize from the already seen to make predictions

This is what neural networks (biological and artificial) are good at (unlike conventional computer).

Neural Networks Introduction

- Artificial neural network (ANN) is a machine learning approach that models human brain and consists of a number of artificial neurons.

- Each neuron in ANN receives a number of inputs.

- An activation function is applied to these inputs which results in activation level of neuron (output value of the neuron).

How do our brains work?

The Brain is a massively parallel information processing system. Our brains are a huge network of processing elements. A typical brain contains a network of 10 billion neurons.

Features of Neural Networks

- Massive connectivity
- Nonlinear, Parallel, Robust and Fault Tolerant
- Capability to adapt to surroundings
- Ability to learn and generalize from known examples
- Collective behaviour is different from individual behaviour

- Artificial Neural Networks mimics some of the properties of the biological neural networks

Why Neural Networks?

- There are two basic reasons why we are interested in building artificial neural networks (ANNs):

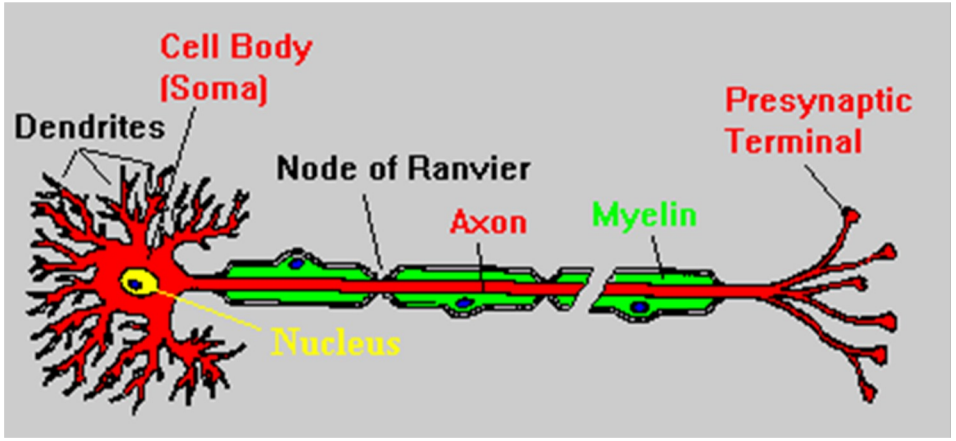
- Technical viewpoint:
- Some problems such as character recognition or the prediction of future states of a system require massively parallel and adaptive processing.

- Biological viewpoint:
- ANNs can be used to replicate and simulate components of the human (or animal) brain, thereby giving us insight into natural information processing.

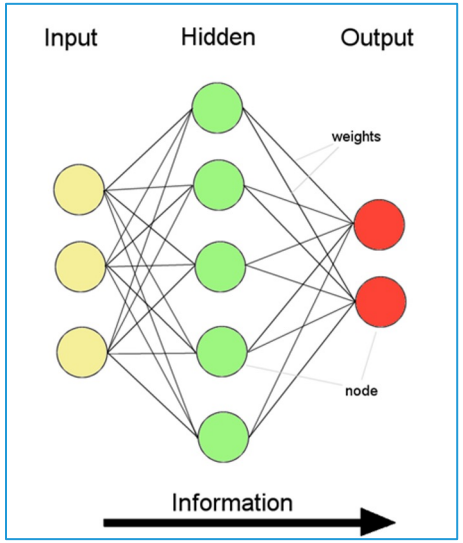
Biological Neuron vs Artificial Neuron (Perceptron)

- Information flow is unidirectional
- Data is presented to Input layer
- Passed on to Hidden Layer
- Passed on to Output layer
- Information is distributed
- Information processing is parallel

Biological Neuron



Network function f: R3 {0, 1}2



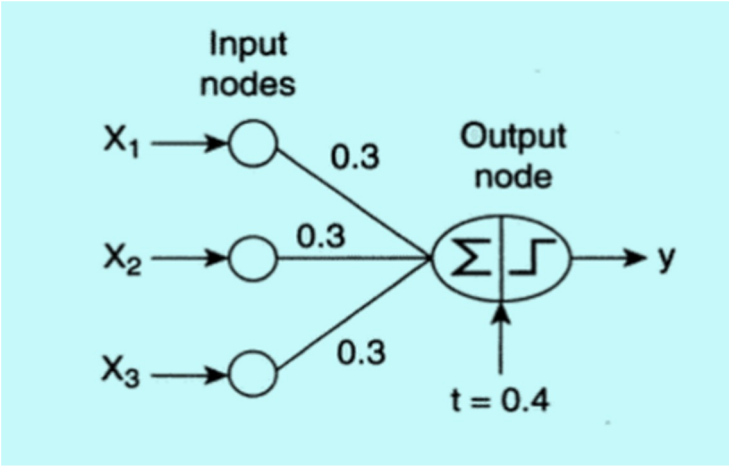
Perceptron Model

- Consider data with 3 Boolean variables X1, X2, X3 and an output variable y

Data

X1	X2	X3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	0	-1
0	0	1	-1
0	1	0	-1
0	1	1	1

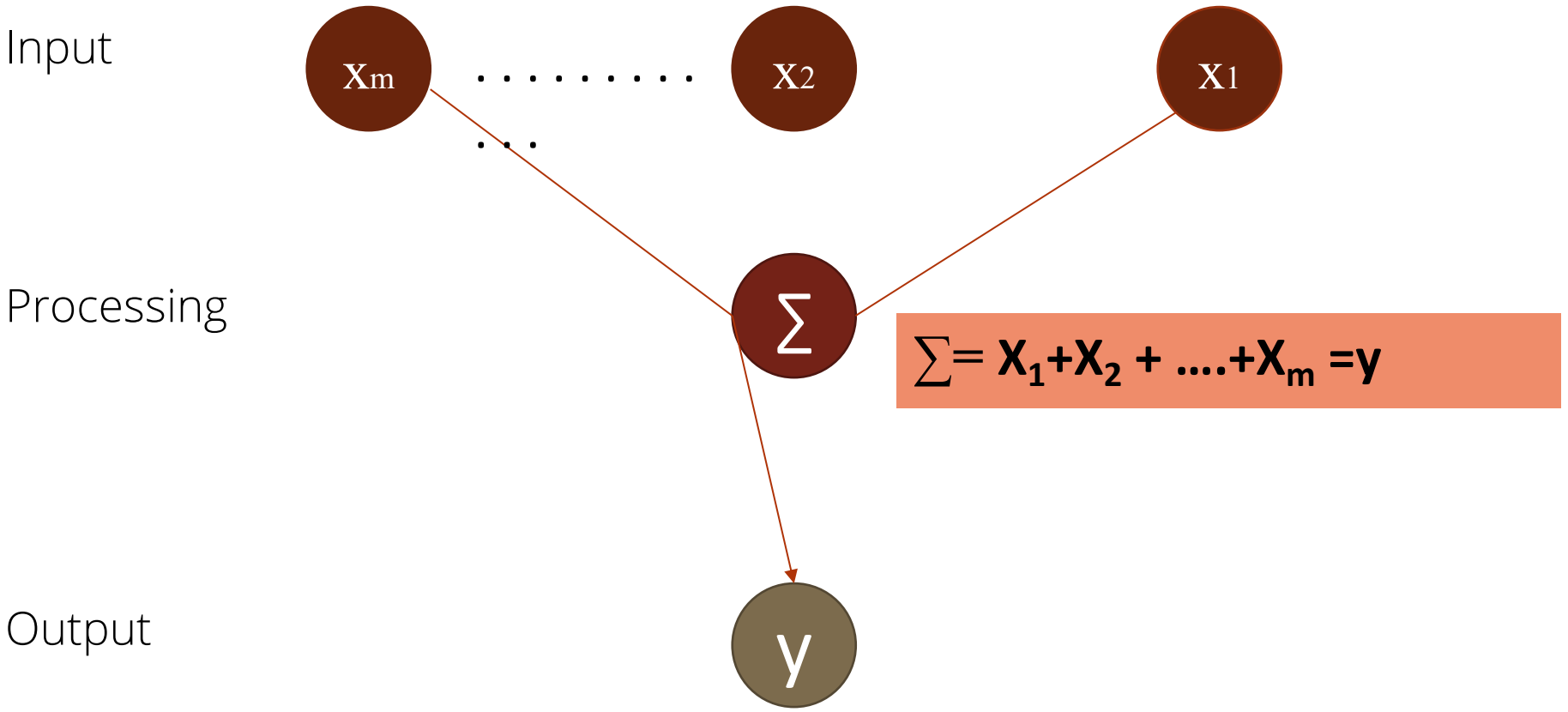
Perceptron



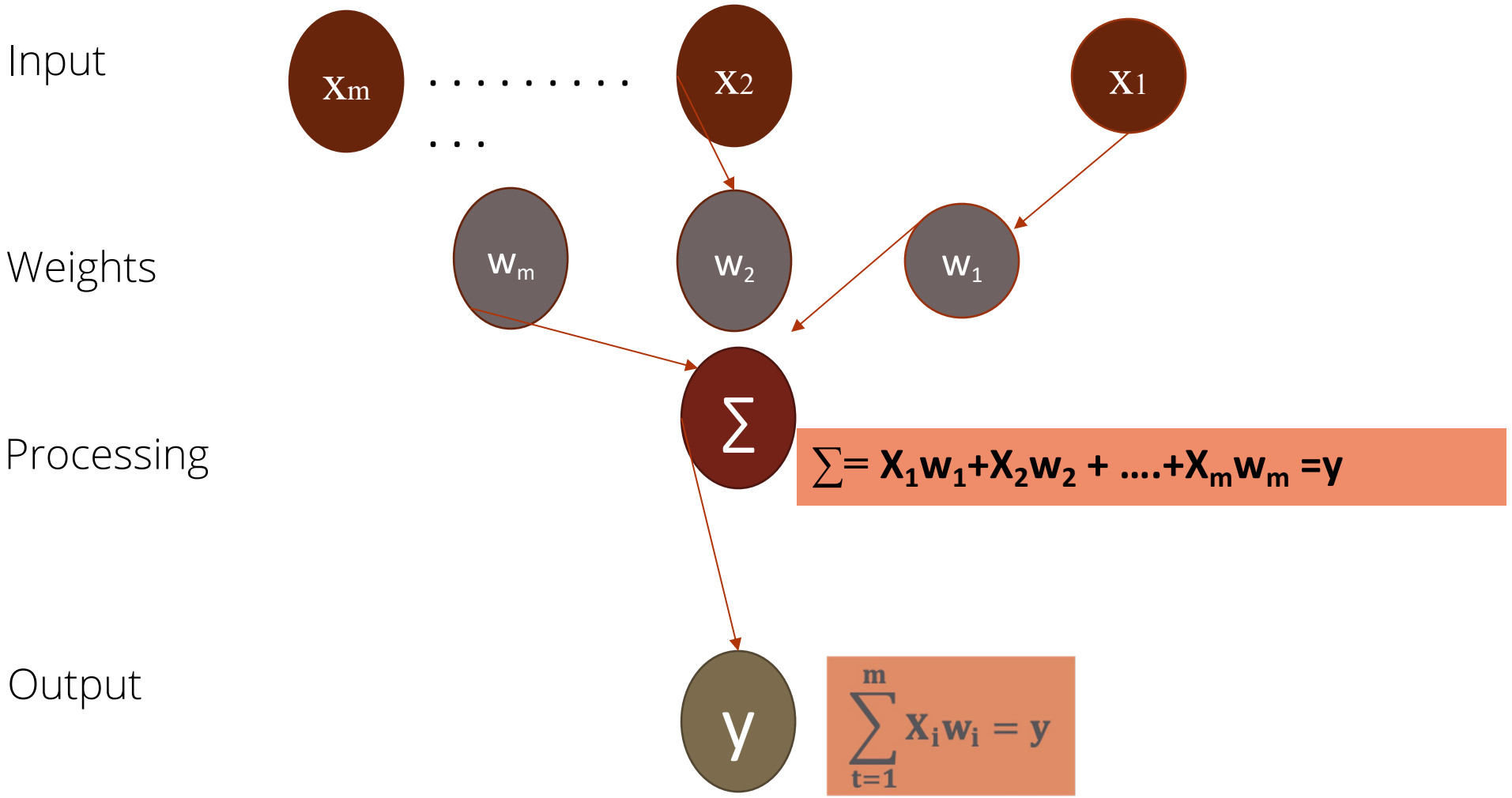
$$y \begin{cases} -1, & \text{if } 0.3\,X1 + 0.3\,X2 + 0.3\,X3 - 0.4 > 0; \\ 1, & \text{if } 0.3\,X1 + 0.3\,X2 + 0.3\,X3 - 0.4 < 0. \end{cases}$$

- Perceptron consists of two types of nodes: input nodes which are used to represent the attributes, and the output node which is used to represent the model output.
- Perceptron computes its output value, by performing a weighted sum on its inputs, subtracting a bias factor t from the sum, and then examining the sign of the result.

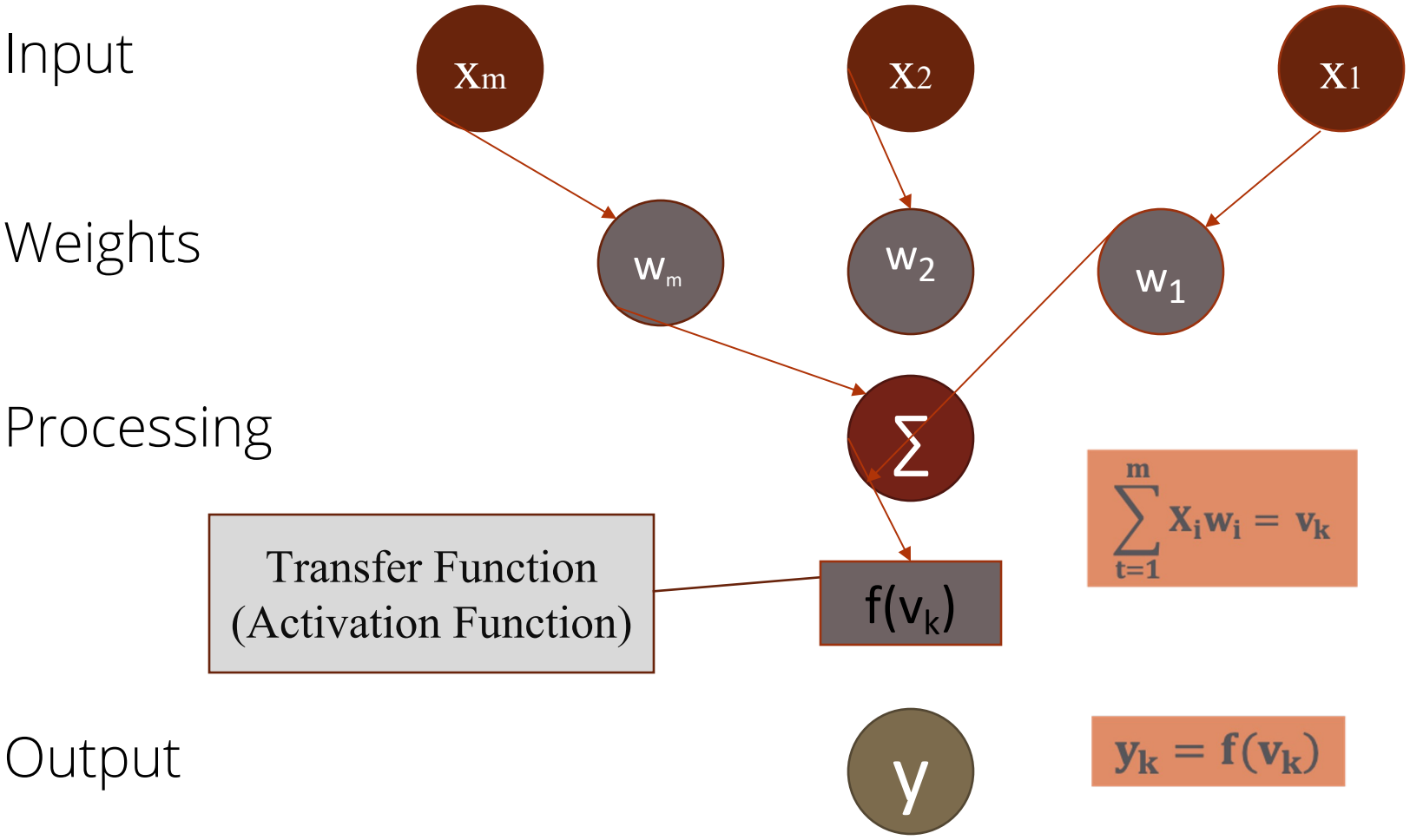
A Single Artificial Neuron…



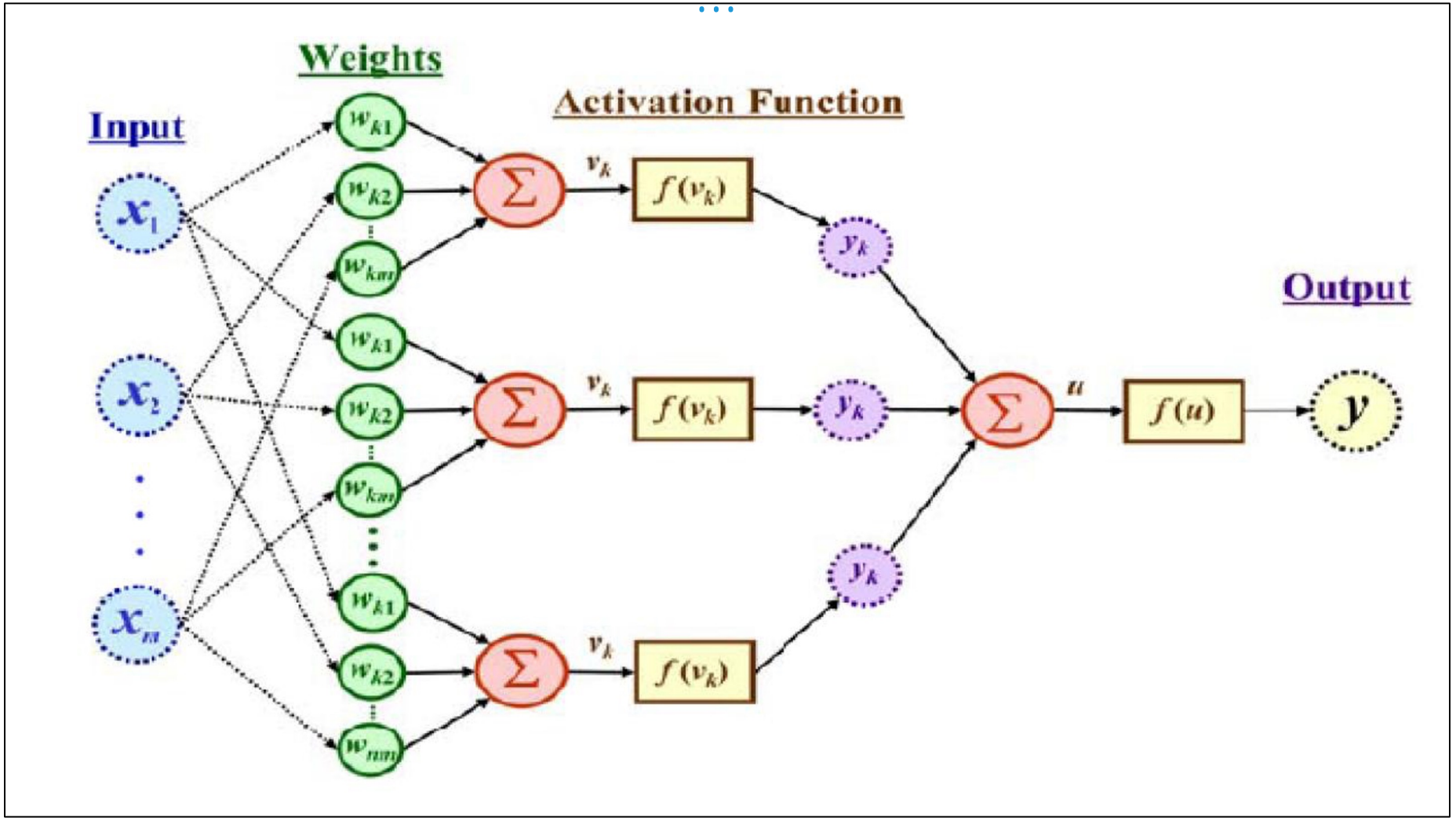
A Single Artificial Neuron…



A Single Artificial Neuron…

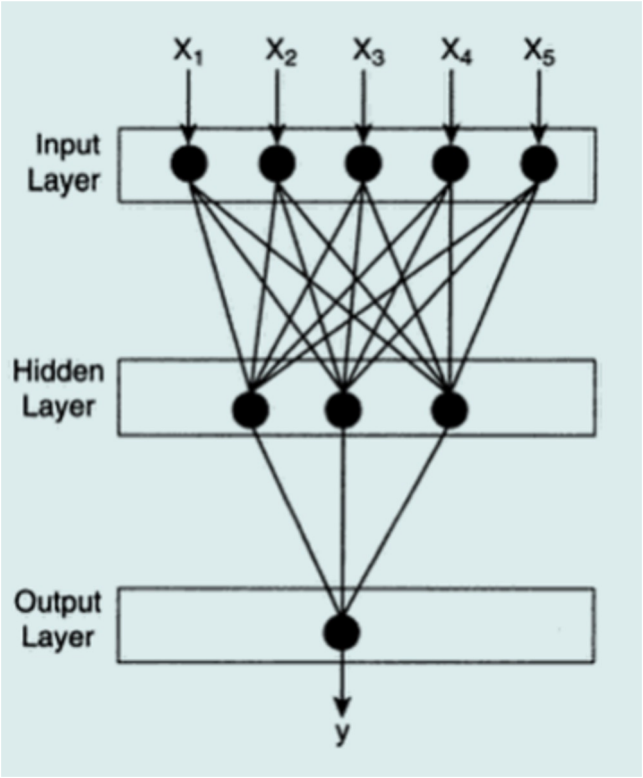


The output is a function of the input, that is affected by the weights, and the transfer functions



Artificial Neural Network (ANN)

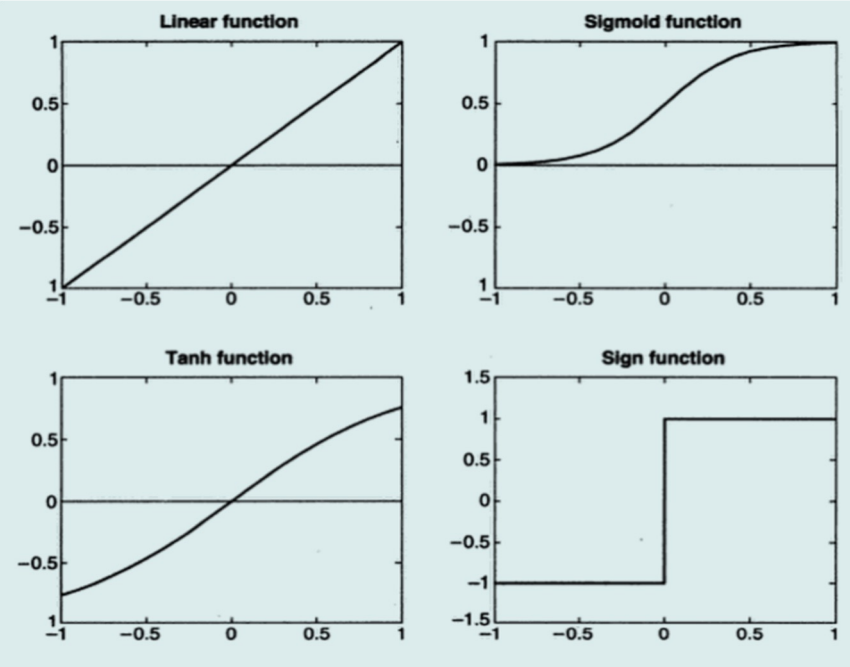
- A neural network is more complex than a perceptron model in multiple ways.
- Network may contain several intermediary layers between its input and output layers which are called hidden layers and the nodes embedded in these layers are called hidden nodes. This restructuring is known as a multilayer neural network.
- In a feed forward network, nodes in one layer are connected to nodes in the next layer.
- In a recurrent network, the links may connect nodes within the same layer or from one layer to the previous layers.



Types of ANN

- The artificial neural network may use types of activation functions other than sign function. Examples of other activation function include linear, sigmoid, and hyperbolic tangent functions.

- These additional complexities allow multilayer neural networks to model more complex relationships between the input and output variables.

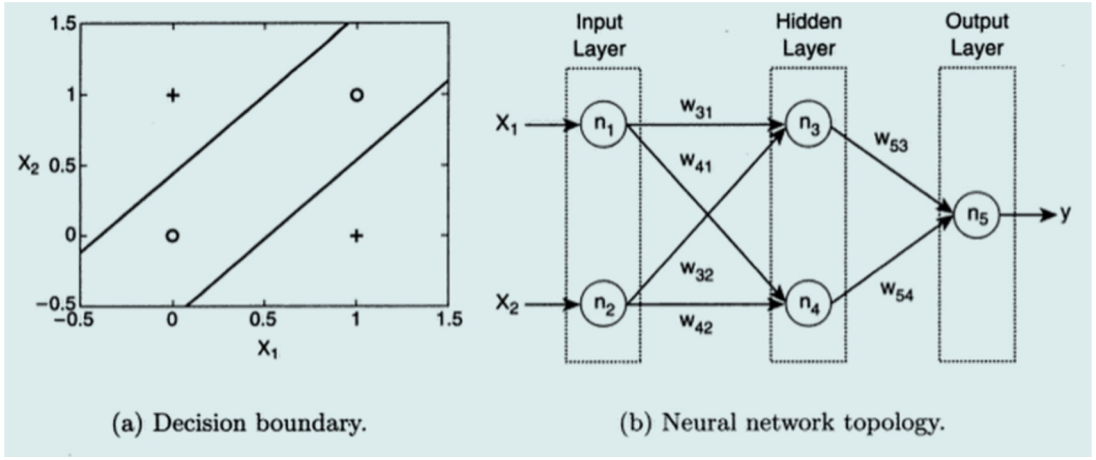


Learning ANN

- In ANN, we determine set of weights w that minimize total sum of squared errors.

$$E(w) = \frac{1}{2}\sum_{i-1}^{N}(y_i - \hat{y}_i)^2$$

- Minimum solution can be obtained by replacing $\hat{y}=w.x$ in the above equation.
- In most cases, output of ANN is non linear function of its parameters, because of the choice of its activation.
- To arrive at a global optimal solution in non linear models, we use gradient descent method to efficiently solve optimization problem.



(a) Decision boundary. (b) Neural network topology.

Gradient Descent Method

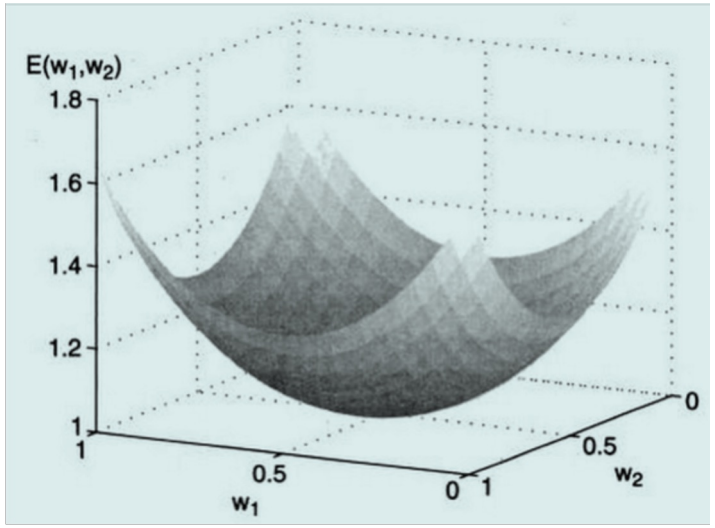
- The gradient descent algorithm can be written as:

$$w_j \leftarrow w_j - \lambda \frac{\partial E(w)}{\partial w_j}$$

- λ is the learning rate, while the second term states that the weight should be increased in a direction that reduces the overall error term

- Each w is initialized to random small value and the iterations are performed until local minima is obtained

- Converging to a local minimum can be very slow or fast depending on the size of learning rate. The weights of the output and hidden



Backpropagation

- In case of hidden nodes, computation becomes complex and error term cannot be assessed without prior information of weights. Back propagation method can be used which works in 2 phases – forward and backward.
- During forward phase, we compute output value of each neuron in the network using previous iteration weight. The computation progresses in the forward direction, outputs at the neuron k are computed prior to computing the outputs at level k+1.
- In backward phase, weight update formula is applied in the reverse direction, i.e. weights at level k+1 are updated before weights at level k.
- This backward propagation approach allows us to use errors for neurons at layer k+1 to estimate the errors for neurons at layer k.

Resilient Backpropagation

- Compared to the traditional back propagation algorithm, the Resilient propagation (rprop) algorithm offers faster convergence and is usually more capable of escaping from local minima.
- Rprop is a first-order algorithm and its time and memory requirement scales linearly with the number of parameters.
- Resilient propagation does not take into account the value of the partial derivative (error gradient), but rather considers only the sign of the error gradient to indicate the direction of the weight update.
- In practice, Rprop is easier to implement than BPNN.

Quick Recap

Neural Networks

- Artificial neural network (ANN) is a machine learning approach that models human brain and consists of a number of artificial neurons.
- In a feed forward network, nodes in one layer are connected to nodes in the next layer.
- In a recurrent network, the links may connect nodes within the same layer or from one layer to the previous layers.
- In ANN, we determine set of weights w that minimize total sum of squared errors.
- To arrive at a global optimal solution in non linear models, we use gradient descent method to efficiently solve optimization problem.
- In case of hidden nodes, we use backpropagation. The Resilient propagation (rprop) algorithm offers faster convergence than traditional backpropagation.

Building a Neural Network Model using R

Contents

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

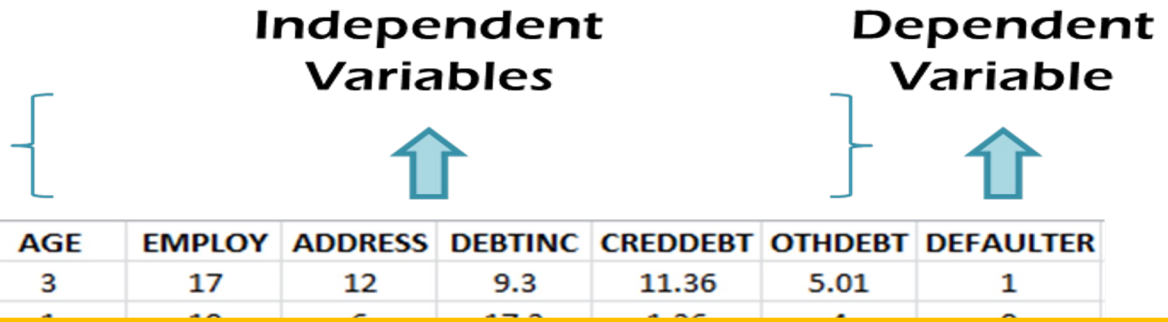
- To predict whether the customer applying for the loan will be a defaulter

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

Neural Network in R...Data Snapshot

BANK LOAN



SN	AGE	EMPLOY	ADDRESS	DEBTINC	CREDDEBT	OTHDEBT	DEFAULTER
1	3	17	12	9.3	11.36	5.01	1

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

Data Pre-Processing

- Since AGE is categorical variable, we create dummy variables before proceeding to neural network model.
- To set up a neural network to a dataset it is very important that we ensure a proper scaling of data. The scaling of data is essential because otherwise, a variable may have a large impact on the prediction variable only because of its scale.
- The common techniques to scale data are min-max normalization and Z-score normalization
- The min-max normalization transforms the data into a common range, thus removing the scaling effect from all the variables. Here we are using min-max normalization for scaling data.

Neural Network in R - Classifying Loan Defaulters...

```
bankloan<-read.csv("BANK LOAN.csv")

library(fastDummies)
bankloan <- dummy_cols(bankloan,select_columns =
"AGE",remove_first_dummy = T)

normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

bankloan$EMPLOY<-normalize(bankloan$EMPLOY)
bankloan$ADDRESS<-normalize(bankloan$ADDRESS)
bankloan$DEBTINC<-normalize(bankloan$DEBTINC)
bankloan$CREDDEBT<-normalize(bankloan$CREDDEBT)
bankloan$OTHDEBT<-normalize(bankloan$OTHDEBT)
```

Neural Network in R - Classifying Loan Defaulters…

```
library(neuralnet)
set.seed(09092022)
nn_bank <- neuralnet(DEFAULTER~AGE_2+AGE_3+EMPLOY+ADDRESS+
                       DEBTINC+CREDDEBT+OTHDEBT, data=bankloan,
                     hidden=3,err.fct="ce",linear.output=FALSE)

out_bank <- cbind(nn_bank$covariate,nn_bank$net.result[[1]])
dimnames(out_bank) <- list(NULL, c("AGE_2","AGE_3",
"EMPLOY","ADDRESS","DEBTINC","CREDDEBT","OTHDEBT","nn-output"))
head(out_bank)
```

- **neuralnet()** takes formula interface. Here, case (status) is the variable of interest.
- **hidden=** Number of hidden neurons in each layer.
- **err.fct=** a differentiable function that is used for the calculation of the error. ‘ce’ stands for cross entropy.
- **linear.output=** FALSE ensures that the output is mapped by the activation function to the interval [0, 1].
- By default the command uses Resilient Backpropagation with Weight Backtracking

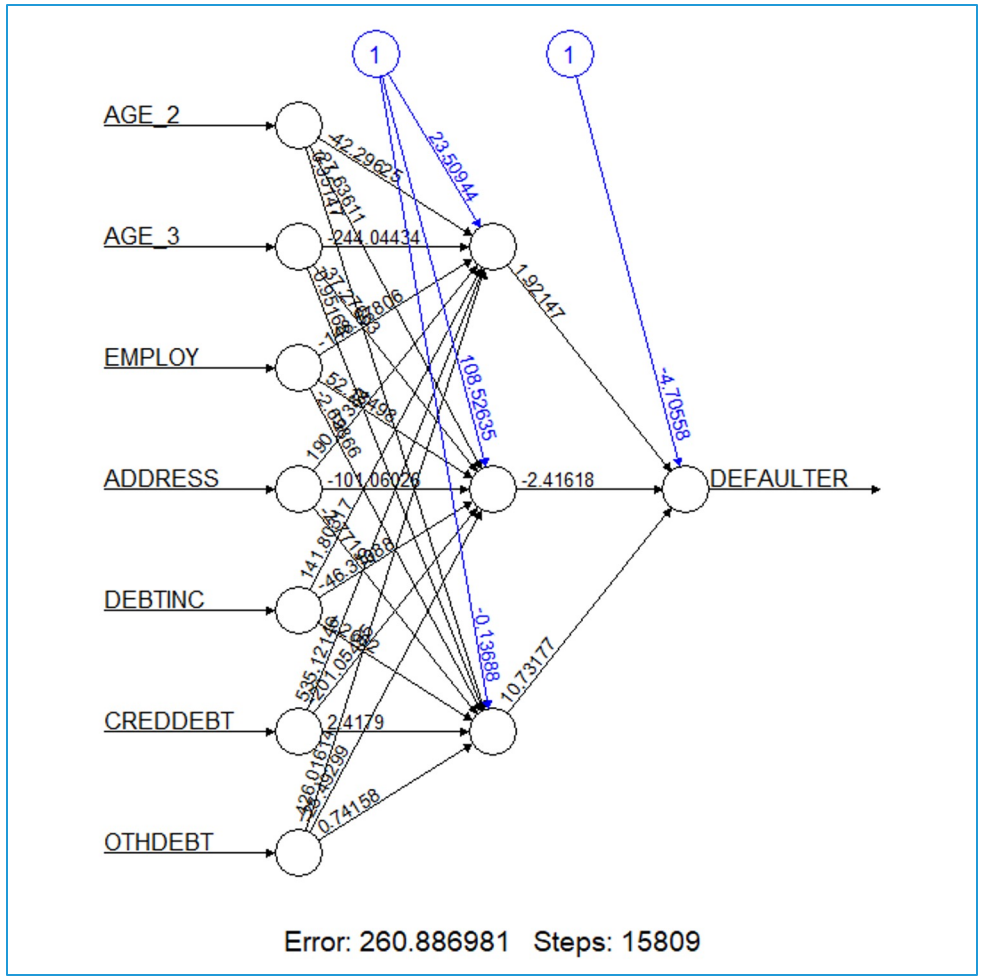
Neural Network in R - Classifying Loan Defaulters...

```
# Output
```

	AGE_2	AGE_3	EMPLOY	ADDRESS	DEBTINC	CREDDEBT	OTHDEBT	nn-output
1,]	0	1	0.54838710	0.3529412	0.21760391	0.55231144	0.18383988	0.943837210
2,]	0	0	0.32258065	0.1764706	0.41320293	0.06569343	0.14640474	0.152127464
3,]	1	0	0.48387097	0.4117647	0.12469438	0.04136253	0.07857672	0.003172850
4,]	0	1	0.48387097	0.4117647	0.06112469	0.12895377	0.02853966	0.009129689
5,]	0	0	0.06451613	0.0000000	0.41320293	0.08661800	0.11156412	0.805638180
6,]	0	1	0.16129032	0.1470588	0.23960880	0.01849148	0.07820608	0.311169470

Neural Network in R - Classifying Loan Defaulters...

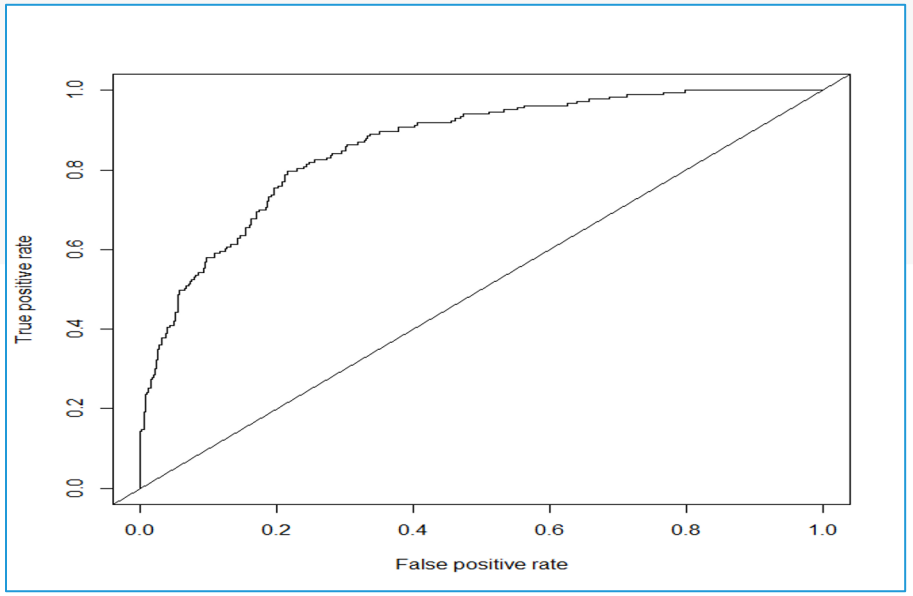
```
plot(nn_bank)

# Output
```



Neural Network in R - Classifying Loan Defaulters...

```
# ROC Curve for NN output
```

```
library(ROCR)
# Convert 'out' to dataframe
outdf<-as.data.frame(out_bank)
pred<-prediction(outdf$`nn-output`,bankloan$DEFAULTER)
perf<-performance(pred,"tpr","fpr")
plot(perf)
abline(0,1)

auc<-performance(pred,"auc")
auc@y.values
[1] 0.8606505
```



More About ANN Work

- Applications in Artificial Intelligence - Handwriting or Face Recognition, Voice Analysis
- The “building blocks” of neural networks are the neurons. In technical systems, we also refer to them as units or nodes.
- Basically, each neuron receives input from many other neurons, changes its internal state (activation) based on the current input, sends one output signal to many other neurons, possibly including its input neurons (recurrent network)
- Information is transmitted as a series of electric impulses, so-called spikes.
- The frequency and phase of these spikes encodes the information.
- In biological systems, one neuron can be connected to as many as 10,000 other neurons.
- Neurons of similar functionality are usually organized in separate areas (or layers).
- Often, there is a hierarchy of interconnected layers with the lowest layer receiving sensory input and neurons in higher layers computing more complex functions.

Quick Recap

Neural Networks in R	• Package **“neuralnet"** has **neuralnet()** that trains a neural network model