FIT3152 - Notes

Data Science (Monash University)

FIT3152 – DATA ANALYTICS: NOTES

Olumide Yinka-Kehinde: 28791444

Table of Contents

-

L1: INTRODUCTION TO DATA SCIENCE

INTRODUCTION TO DATA SCIENCE:

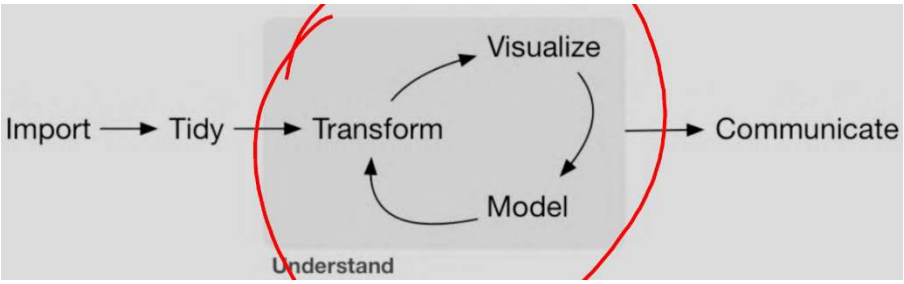
Data science:

- An inter-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data
- A "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyse actual phenomena" with data

Some necessary ***skills*** for a data scientist:

- Collect, cleanse, manage and combine data – may come from disparate sources
- Analyse and model the data using statistical and (AI) machine learning techniques

Data analysis process:



DATA STRUCTURES:

Data is stored in R using data structures (objects) to which functions (methods) are applied

Array:

- Contains data of the same type
- Vector: 1D, Matrix: 2D, Array: 3^{+} Dimensions

Data Frame:

- Row × Column data format – **each column is a vector**

List:

- An ordered collection of (possibly different) types

DESCRIPTIVE STATISTICS:

HYPOTHESIS TESTING:

A hypothesis about a certain population parameter is tested using sample data usually consisting of one or two groups

Hypothesis is evaluated against a null hypothesis that assumes equality/no difference between the group(s) and the value being tested

P-values: Evaluate how well the used sample data support the argument of the null hypothesis

- ***High p-values:*** Indicate that the sample data is **likely** with a true null hypothesis
- ***Low p-values:*** Indicate that the sample data is **unlikely** with a true null hypothesis

Low p-value indicates there is more evidence to conclude that the test hypothesis is true against the null hypothesis

REGRESSION:

Provides a model for the relationship between different variables with primary aim of prediction

Minimize sum of squared error (squared residuals) to come up with best fit

CORRELATION AND PREDICTION:

Correlation indicates the degree of association between 2 variables

Usually quantifies the strength of the linear relationship between variable (-1 to +1)

Higher association between 2 variables (positive or negative correlation) indicate a better predictive capability

HISTOGRAM:

A histogram graphically displays the shape and spread of data

Groups data into a number of class intervals and counts the frequency in each

L2: VISUALIZATION

VISUALIZATION OF DATA:

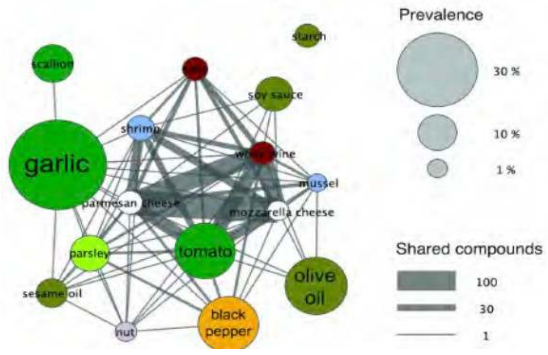
For data graphics, think about:

- What is being conveyed?
- How it is being conveyed, what is the main device: size, shape, colour, position
- Number of dimensions represented → How many variables associated with each data point?
- How is space used

How many dimensions does the figure show?

A. 1
B. 2
C. 3
D. 4
E. More than 4

Think about how many variables are associated with each data point...



Major graphic types include: Time series, statistical distributions, maps, hierarchies, networks

5 dimensions in the Iris data set

VISUALIZATION USING R:

GGPLOT2:

- One of the most commonly used packages for display quality graphics
- Made up of data points + scales + annotations + statistical + summaries

Graphs are constructed first with a Geom, which specifies the type of plot and the data

Following this, aesthetic element are added:

- Statistics (summaries, data transformation)
- Scales/coordinate systems
- Faceting (conditional grouping of subset of data)

- Positional adjustment (jitter, etc.)
- Annotation
- Aesthetics

L3: DATA MANIPULATION IN R

Visualising data:

- Number of dimensions in a data set
  - 5 dimensions → Graphic capable of displaying 5 dimensions
- Major families graph types: Time series, statistical distributions, maps, hierarchies, networks

Aggregate: Applies function to single columns, grouped by a factor

- Mean of "x" based on gender

By: Applies function across multiple columns, grouped by a factor

- Correlation between variables

DATA MANIPULATION:

SUMMARIZING DATA BY GROUPS:

Data grouped by factors:

- Applying a function to a single column
- Applying a function to a group of columns

Why do we need to do this?

- To simplify the data, making comparisons easier
- Reduce data complexity, enabling further analysis

SUMMARIZING DATA BY GROUPS:

aggregate:

Function creates a table by applying a function to data in individual columns grouped by a factor (or factors).

- Note: columns referred to by indices [ ] for compactness

```
> aggregate(iris[1:4], iris[5], mean)
     Species Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa         5.01        3.43         1.46       0.246
2 versicolor         5.94        2.77         4.26       1.326
3  virginica         6.59        2.97         5.55       2.026
```

Use cols 1–4, ivs

Grouping variable (factor) col 5, iris

function we apply

?aggregate

- Description

  `aggregate(x, ...) : Splits the data into subsets, computes summary statistics for each, and returns the result in a convenient form.`

- Usage

  `aggregate(x, by, FUN, ..., simplify = TRUE)`

Template

- Arguments

  `X : An R object.`
  `By : List of grouping elements`
  `FUN : Function to compute the summary statistics`
  `Simplify : Indicates whether results should be simplified to a vector or matrix if possible.`

by:

Enables a function to be applied across multiple columns of a data frame grouped by a factor or factors. First breaks data into groups, then applies a function.

- To calculate the correlation of sepal length and width

```
> by(iris, iris[5], function(df) cor(df$Sepal.Length,
  df$Sepal.Width))
Species: setosa
[1] 0.743
------------------------------------------------------------
Species: versicolor
[1] 0.526
------------------------------------------------------------
Species: virginica
[1] 0.457
```

Grouping Variable — data

?by: applying the cor function

Looking more closely at the way correlation is calculated:

Data frame — Column of factors — Declaring new function on the fly. Parameter is temporary data frame created for each factor

```
> by(iris, iris[5], function(df) cor(df$Sepal.Length,
  df$Sepal.Width))
```

Values in temp data frame passed to cor function

df = temp data frame

FIT3152 Data analytics – Lecture 3 Slide 14

as.table:

Function converts the output format of a function from a list to a table

```
> as.table(by(iris, iris[5], function(df) cor(df[1], df[2])))
Species
    setosa versicolor  virginica
     0.743      0.526      0.457
```

name of data frame

```
> Sepal.cor <- as.data.frame(as.table(by(iris, iris[5],
  function(df) cor(df[1], df[2]))))
> Sepal.cor
     Species  Freq
1     setosa 0.743
2 versicolor 0.526
3  virginica 0.457
```

col name given by default

as.data.frame:

Function converts "coerces" the output of a table into a data frame

colnames:

Function assigns new column names to a data frame

Merging data frames (and saving):

Using a common column – "Species" – and – rounding data

Note: We could've used ***"cbind"*** – since "Species" matches up in the data frames

merge
```
> iris.cor <- merge(Sepal.cor, Petal.cor, by = "Species")
```
round
```
> iris.cor[,2] = round(iris.cor[,2], digits = 3)
```
round
```
> iris.cor[,3] = round(iris.cor[,3], digits = 3)
```
Save file
```
> write.csv(iris.cor, file = "Iris.cor.csv",
  row.names=FALSE)
```

SepalPetalcor.csv

Species	Sepal.cor	Petal.cor
setosa	0.743	0.332
versicolor	0.526	0.787
virginica	0.457	0.322

Adding (and removing) columns:

R adds new column to data frame if output of a column operation is specified as new column

cbind: Can be used to append a vector or data frame by columns

This lets us store the results of row operations, including factor generation

Add 2 columns containing aspect ratio (length/width) for sepals/petals as they're being calculated

```
> niris <- iris # creating a new data frame
> niris$Sepal.ar <- niris[1] /niris[2] # add new column
> niris$Petal.ar <- niris[3] /niris[4] # add new column
> head(niris)
```

give new column a name

Deleting columns:

Cannot be undone

To remove the first column:

- niris$Sepal.Length <- NULL

Tedious for multiple columns. **A quicker but potentially dangerous way to remove first 4 columns**

- niris <- niris[,c(5:7)] # Keep columns 5, 6, 7

which.max:

Printing row with longest petal

To find the row containing the longest petal (ignoring species), use:

```
> which.max(iris[,3])
[1] 119
```

iris[119]

To print this row use:

```
> iris[which.max(iris[,3]),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
119          7.7         2.6          6.9         2.3 virginica
```

To find the maximum for each species use 'by' function and a temporary data frame.

Longest petal in each group

factor

function

data

Putting together gives:

```
> by(iris, iris[5], function(df) df[which.max(df[,3]),])
Species: setosa
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
25          4.8         3.4          1.9         0.2  setosa
------------------------------------------------------------
Species: versicolor
   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
84            6         2.7          5.1         1.6 versicolor
------------------------------------------------------------
Species: virginica
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
119          7.7         2.6          6.9         2.3 virginica
```

do.call:

Repeatedly executes function based on inputs

First, assign a variable name (for clarity):

```
> max.type <- by(iris, iris[5], function(df)
  df[which.max(df[,3]),])
```

Var

repeat row with longest petal

```
> do.call(rbind, max.type)
           Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
setosa              4.8         3.4          1.9         0.2     setosa
versicolor          6.0         2.7          5.1         1.6 versicolor
virginica           7.7         2.6          6.9         2.3  virginica
```

This can be converted to a data frame using:

```
> XX <- as.data.frame(do.call(rbind, max.type))
```

WORKING WITH DATES AND TIMES:

To work with dates in R, you first need to convert the character representation of date into a "date" using the *"as.Date"* function so that data is read and interpreted correctly

Dunnhumby : data

customer_id	visit_date	visit_delta	visit_spend
40	4/04/10	NA	44.83
40	6/04/10	2	69.68
40	19/04/10	13	44.61
40	1/05/10	12	30.39
40	2/05/10	1	60.73
40	12/05/10	10	50
40	15/05/10	3	3
40	18/05/10	3	36.89
40	19/05/10	1	9.07
40	23/05/10	4	14.01
40	26/05/10	3	16.97
40	31/05/10	5	8.69
...	...	...	...

Without date conversion

Calculating minimums without date conversion:

```
> min.type <- by(DH, DH[1], function(df)
  df[which.min(df[,2]),])
> do.call(rbind,min.type)
```

	customer_id	visit_date	visit_delta	visit_spend
40	40	01-05-10	12	30.39
79	79	01-01-11	9	81.70
119	119	01-03-11	3	10.69
123	123	01-02-11	4	35.20
134	134	01-02-11	1	54.77

With date conversion

Calculating minimums with date conversion:

```
> min.type <- by(DH, DH[1], function(df)
  df[which.min(as.Date(df[,2],"%d-%m-%y")),])
> do.call(rbind,min.type)
```

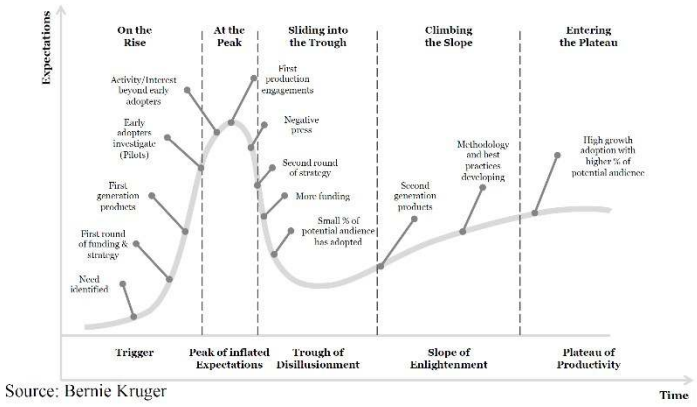
	customer_id	visit_date	visit_delta	visit_spend
40	40	04-04-10	NA	44.83
79	79	07-04-10	NA	150.87
119	119	01-04-10	NA	20.00
123	123	02-04-10	NA	66.94
134	134	01-04-10	NA	50.32

L4: DATA SCIENCE METHODS & TIDYING DATA

DATA SCIENCE INDUSTRY:

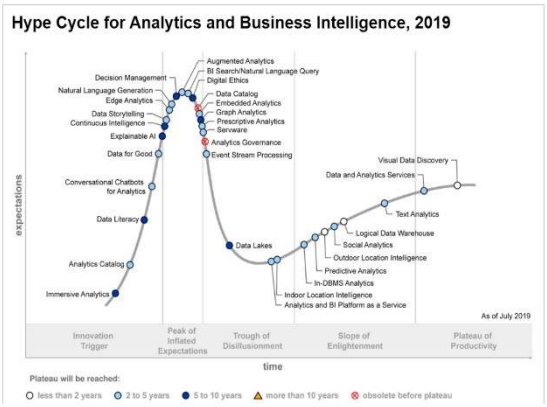
GARTNER HYPE CYCLE:

Gartner Hype Cycle



Source: Bernie Kruger

Gartner Hype Cycle: Analytics & BI



Key phases of technology's life cycle:

- Innovation trigger
- Peak of Inflated Expectations
- Trough of Disillusionment
- Slope of Enlightenment
- Plateau of Productivity

Gartner Hype Cycle: Analytics & BI:

On the rise:

- Explainable AI, data for good, data storytelling, natural language generation

At the peak:

- Augmented analytics, natural language query, digital ethics

Sliding into the trough

- Data lakes, analytics governance, analytics and BI as a service

They identify 5 key trends:

- *Augmented Analytics:* Using machine learning to automate data preparation and analytics
- *Digital Culture:* Data literacy, digital ethics, data-for-good
- *Relationship Analytics:* Growing use of graph, location and social network analysis
- *Decision Intelligence:* Capturing factors leading to a decision
- *Operationalising and Scaling:* Building up analytics service for any part of the business that requires it

Data Science: Key skills

- Statistics, programming, machine learning, data visualisation, data munging, texting mining

HOW DATA SCIENCE IS BEING USED:

Descriptive: What happened?

- Condense data into smaller, more useful pieces of information
- Dashboards, mostly static; Standard and ad-hoc reporting

Diagnostic: Why did it happen?

- Data analysis by employing predefined criteria
- Rules based data analysis (controls testing, suspicious, transaction activity)
- Essential to process to rectification and improvement

Explorative: What might be interesting?

- What is not apparent at first glance
- Automated discovery (machine learning such as clustering)

Predictive: What is likely to happen?

- Predictive modelling on historical data to produce future likelihood of events
- Prediction (Random forest, NN, Deep learning)

Prescriptive: What we can do about it?

- Use model to intervene
- Suggests best option for handling a future scenario

DATA SCIENCE METHODOLOGIES:

A data mining methodology to meet specific requirements of industrial procedures is needed

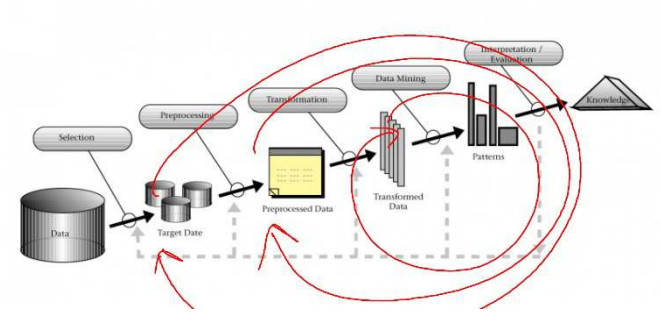
KDD – (Knowledge Discovery in Databases)

- Broad process of finding knowledge in data and emphasises the "high-level" application of particular data mining methods

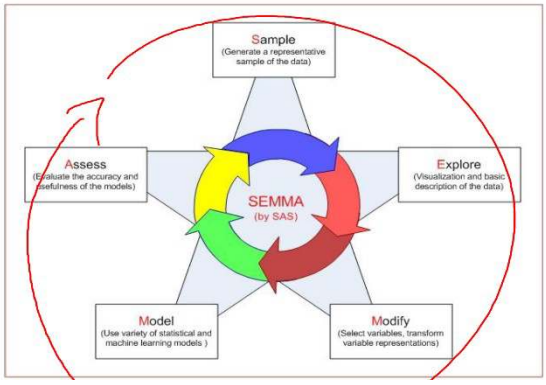
SEMMA – (Sample, Explore, Modify, Model and Assess)

- Methodology for data mining process proposed by the SAS Institute for software package Enterprise Miner

KDD



SEMMA

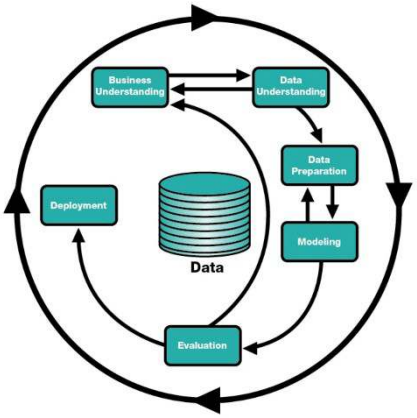


CRISP–DM

CRISP-DM – (Cross-Industry Standard Process for DM):

Developed by a consortium of data mining vendors and companies through an effort founded by the European commission

(CRISP-DM preferred due to inclusion of business aspects)

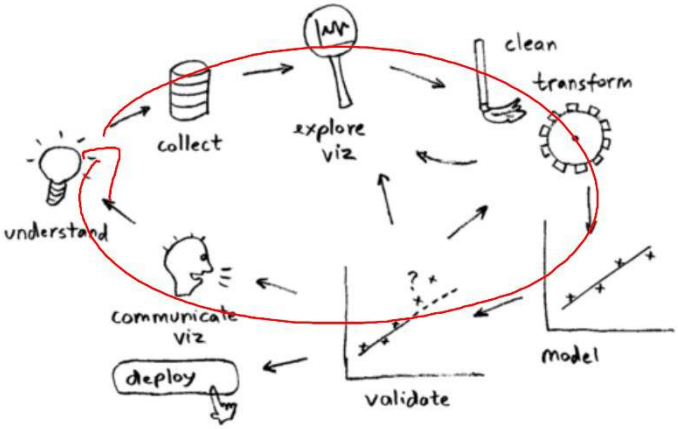


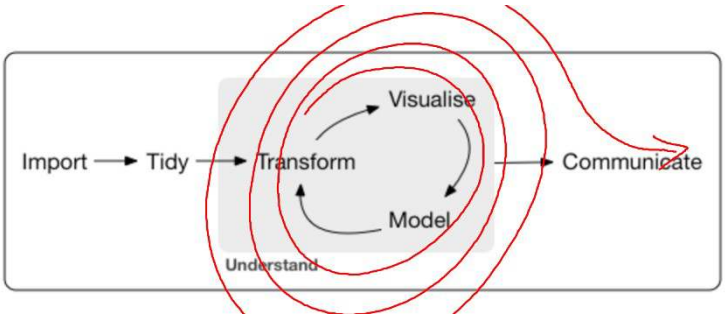
http://crisp-dm.eu/

CRISP-DM:

- Business understanding
- Data understanding
- Data prepartion
- Modelling
- Evaluation
- Deployment

DATA SCIENCE WORKFLOW:





DIRTY & TIDY DATA:

Dirty data: some concerns

Identifier	Edition Statement	Place of Publication	Date of Publication	Publisher	Title	Author	Contributors
206		London	1879 [1878]	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	FORBES, Walter.
216		London; Virtue & Yo	1868	Virtue & Co.	All for Greed. [A novel. The dedication s	A., A. A.	BLAZE DE BURY, Ma
218		London	1869	Bradbury, Evans & C	Love the Avenger. By the author of â€œA	A., A. A.	BLAZE DE BURY, Ma
472		London	1851	James Darling	Welsh Sketches, chiefly ecclesiastical, to	A., E. S.	Appleyard, Ernest Si
480	A new edition, revis	London	1857	Wertheim & Macint	[The World in which I live, and my place	A., E. S.	BROOME, John Henr
481	Fourth edition, revis	London	1875	William Macintosh	[The World in which I live, and my place	A., E. S.	BROOME, John Henr
519		London	1872	The Author	Lagonelli. By the author of Darmayne [	A., F. E.	ASHLEY, Florence En
667		pp. 40. G. Bryan & Co: Oxford, 1898			The Coming of Spring, and other poems	A., J. [A., J.	ANDREWS, J. - Writ
874		[London]	16		A Warning to the inhabitants of Englan	[Rema[illegible].	ADAMS, Mary.
1143		London	1679		A Satyr against Vertue. (A poem: supp	A., T.	OLDHAM, John.
1280		Coventry	1802	Printed by J. Turner	An Account of the many and great Loan		CARTE, Samuel. [JAC
1808		Christiania	1859		Erindringer som Bidrag til Norges Histor	AALL, Jacob.	AALL, J. C. [LANGE, C
1905		Firenze	1888		Gli Studi storici in terra d'Otranto ... Fra	AAR, Ermanno - pse	S., L. G. D. [SIMONE,
1929		Amsterdam	1839, 38-54		De Aardbol. Magazijn van hedendaagsch		WITKAMP, Pieter Ha
2836		Savona	1897		Cronache Savonesi dal 1500 al 1570 ... A	ABATE, Giovanni Ap	ASSERETO, Giovanni
2854		London	1865	E. Moxon & Co.	See-Saw, a novel ... Edited [or rather, w	ABATI, Francesco.	READE, William Win
2956		Paris	1860-63		Gel[illegible]ise d'une partie de la Haute E	ABBADIE, Antoine T	RADAU, Rodolphe.
2957		Paris	1873		[With eleven maps.]	ABBADIE, Antoine T	RADAU, Rodolphe.
3017	Nueva edicion, anot	Puerto-Rico	1866		[Historia geogr[illegible]fica, civil y politica de	ABBAD Y LASIERRA,	ACOSTA Y CALBO, Jo
3131		New York	1899	W. Abbatt	The Crisis of the Revolution, being the s	ABBATT, William.	ANDRE[illegible], John - Ma
4598		Hull	1814	The Author	Peace: a lyric poem. [With prefatory ad	ABBOTT, Thomas Ea	WRANGHAM, Franc
4884		London	1820	J. Hatchard & Son	Abdallah; or, The Arabian Martyr: a Chr		BARHAM, Thomas F
4976	[Another edition.] A	Oxonii	1800	J. Cooke, etc.	[Abdolletiphi HistoriÃ¦ Ã¦gypti compen		WHITE, Joseph - Car
5382		London	1847, 48 [1846-48]	Punch Office	The Comic History of England ... With ...	A'BECKETT, Gilbert	LEECH, John - Artist
5385	[Another edition.] Il	London	[1897?]	Bradbury, Agnew &	[The comic history of England ... With tv	A'BECKETT, Gilbert	LEECH, John - Artist
5389	[Another edition.]	London	[1897?]	Bradbury, Agnew &	[The Comic History of Rome ... Illustrate	A'BECKETT, Gilbert	LEECH, John - Artist
5432		Milano	1893		Signa: opera in tre atti [founded on the	A'BECKETT, Gilbert	MAZZUCATO, Giova
6036		London	1805	C. & R. Baldwin	The Venetian Outlaw, a drama in three		ELLISTON, Robert W
6821		Aberdeen	1837	J. Davidson & Co.	Description of the Coast between Aberd		DUNCAN, William -

Data in the real world is dirty, it can be:

Incorrect data:

- For data to be correct (valid), its values must adhere to its domain (valid values)
- Month must be in range of 1-12

Inaccurate data:

- Data value can be correct without being accurate
- “VIC” and “Sydney” are both correct but shouldn’t’ be used together

Business rule violations

- Start date should always precede an finish date

Inconsistent data:

- Uncontrolled data redundancy results in inconsistencies
- Customer name may be recorded on 2 different databases as Mary Smith, Mary L. Smith

Incomplete data

Non-integrated data

TIDY DATA:

Tidy data seeks a consistent format that has:

- Each variable in its own column
- Each observation in its own row
- Each value in its own cell

- 2 benefits: Consistency and exploits R's vector nature

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

Tidy data

This data format is preferred as each observation is in a separate row, indexed by level (treatment).

person	treatment	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Table 3: The same data as in Table 1 but with variables in columns and observations in rows.

There are many other actions that may be needed to clean and tidy data sets, including:

- Replacing missing values (Dealing with)
- Standardisation
- Normalisation

TRANSFORMING DATA:

3 challenges, we will:

- Recode data by creating a new index
- Extract a subset of data based on values in a range
- Extract a subset of data based on values in a second data frame
- Display effect of 2 variables on third using a heatmap

Heatmaps **are a useful graphic to observe the effect of 2 factors on a variable**

L5: NETWORK ANALYSIS

Studying network structure lets us:

- Determine relative importance of key players in a social (or other) network
- Understand fundamental structure of natural or man-made systems, reasons for fragility or robustness
    - Fragility → Train breaks at Caulfield? No work
    - Robustness → Internet server down? Internet don't stop because other pathways

Terminology: Edges and Vertices

Vertices (nodes) typically represent entities in the network

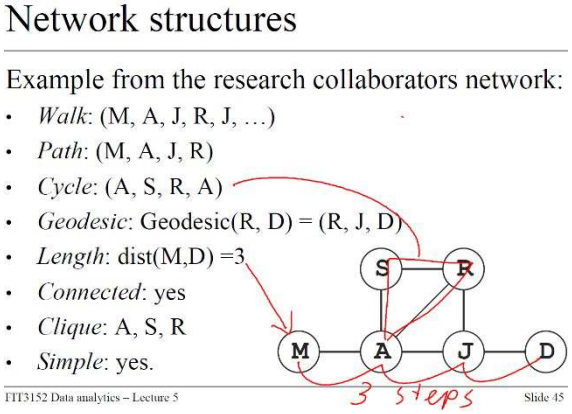
Edges (arcs) represent connections between these entities

- Edges may be ***undirected*** (Friend A ←→ Friend B) or directed (Parent → Child)
- Edges may be ***weighted***: To indicate he strength of a relationship or bond

NETWORK STRUCTURES:

Particular link sequences have formal descriptions:

- ***Walk:*** A sequence of links
- ***Path:*** A walk with no repeated vertices
- ***Cycle:*** A walk that begins and ends at the same vertex
- ***Geodesic:*** Shortest path between two vertices
- ***Length:*** Number of links in a walk or path
- ***Connected:*** There is a path between each pair of vertices
- ***Directed graph:*** Definitions above apply; Travel on each edge permitted in 1 direction only
- ***Loop:*** An edge from a vertex to itself
- ***Complete:*** A graph where every vertex is joined to every other vertex
- ***Subgraph:*** A subset of a graph
- ***Clique:*** A subgraph that is complete (every vertex joined to every other vertex)
- ***Simple:*** A graph with no loops or multi-edges (> 1 edge between same pair of vertices) can be connected or disconnected

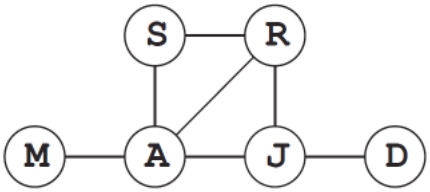


Adjacency matrix:

Summarizes the network by indicating the connections between individuals

	J	A	S	D	R	M
J	0	1	0	1	1	0
A	1	0	1	0	1	1
S	0	1	0	0	1	0
D	1	0	0	0	0	0
R	1	1	1	0	0	0
M	0	1	0	0	0	0

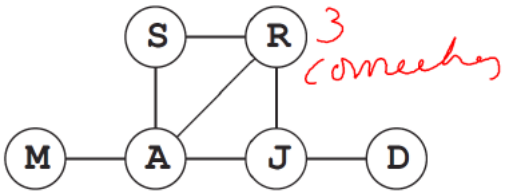
Symmetric because undirected



Degree of a vertex:

Arguably the single most important measure of a vertex's significance in a network

- ***Degree:*** Number of edges connected to a vertex; Size of the vertex's neighbourhood
- For directed graphs, this is adapted to in-degree and out-degree
- Example: $d_R = 3$



NETWORK STATISTICS:

Statistics of the network as a whole include:

- ***Diameter:*** Longest geodesic between any 2 vertices
  - "How far apart are the 2 most separated elements"
- ***Average path length:*** Average distance (geodesic) between any 2 vertices over the whole network
- ***Degree distribution:*** Probability distribution describing magnitude of vertices in the network

Analysis of these and other factors describe how *connected, robust or fragile* etc. a network is

Example from research collaborators network:

Degree distribution and distance matrices for the research collaborators network:

- *Diameter: = max(dist(u,v)) = 3*
- *Degree distribution:*
- *Average path length: 1.667*

Pull from distance matrix

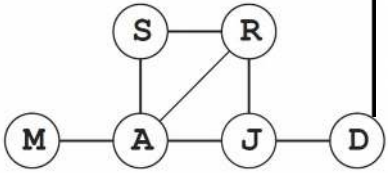
Degree	N
0	0
1	2
2	1
3	2
4	1

25 ÷ 15

Sum of distances matrix

Count number in distance matrix

	J	A	S	D	R	M
J		1	2	1	1	2
A			1	2	1	1
S				3	1	2
D					2	3
R						2
M	Distance Matrix					



Importance of a vertex is based on 2 factors:

Number of connections with other vertices

- Degree

Centrality of vertex within the network (strategic power to control information)

- Betweenness
- Closeness
- Eigenvector

CENTRALITY MEASURES:

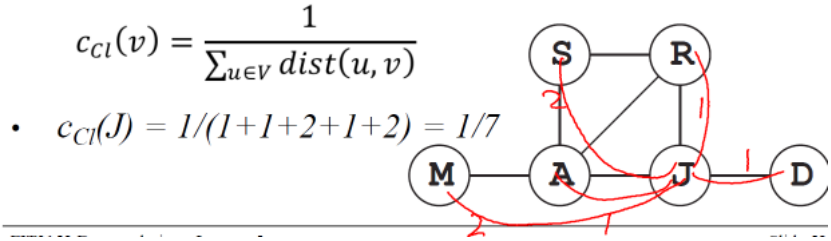
CLOSENESS CENTRALITY:

For this measure, a **vertex is "close" if there is a small total distance between it and all the other vertices in the network**

- **Closeness centrality:** Inverse of total distance between a vertex and the others

$$c_{Cl}(v) = \frac{1}{\sum_{u \in V} dist(u,v)}$$

- $c_{Cl}(J) = 1/(1+1+2+1+2) = 1/7$



CLOSENESS CENTRALITY:

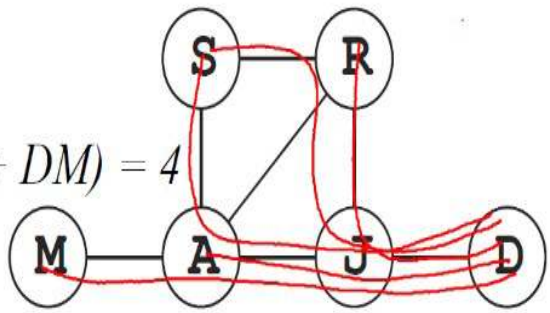
This measure indicates the **degree to which the vertex is "between" other vertices**

- **Betweenness centrality:** Sums number of shortest paths between ***s*** and ***t*** $\sigma(s,t)$ through vertex ***v*** (proportionally if more than one shortest path exists*)

$$c_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

$c_B(J) = (DR+DA+DS^*+DM) = 4$

$c_B(M) = 0$



EIGENVECTOR CENTRALITY:

Too difficult to calculate by hand but included for completeness

- Gives higher weight to vertices with neighbours that are more central in the graph
- Google PageRank

WHICH CENTRALITY MEASURE:

Betweenness Centrality:

- Measures the hub potential of a node
- High Between Centrality nodes act as hubs/relays/bridges

Closeness Centrality:

- Measures how well a node is connected locally
- High Closeness Centrality nodes are strong local influencers

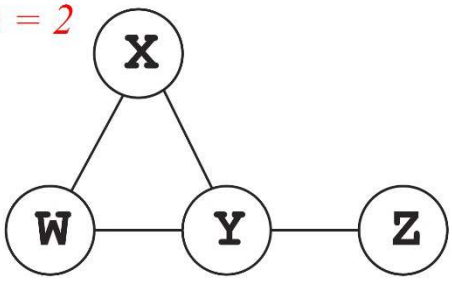
Eigencentrality:

- Weights a node according to the quality of its connections
- Nodes connected to important nodes are ranked higher

Class example – answers

For the graph below calculate the following:

- *Average path length ave(1, 1, 2, 1, 2, 1) = 1.333.*
- *Diameter for example d(W,Z) = 2*
- $d_Y = 3$
- *Degree distribution 3, 2, 2, 1*
- $c_B(Y)$ *W-Z, X-Z = 2*
- $c_{Cl}(Y)$ *1/(1+1+1) = 0.333.*
- *Cliques 2:W-X, W-Y, X-Y, Y-Z, 3:W-X-Y*



Look at notes for how to write R code for networks

R:

Graph summary

Clique 4 

Diameter, average path length, clique size:

> diameter(g)
[1] 3

Clique 3 

> average.path.length(g)
[1] 1.666667

Clique 2 

> table(sapply(cliques(g),length))
1 2 3
6 7 2

Clique 1 •

Summary: who is the most important person in the network?

	J	A	S	D	R	M
Degree	3.00	4.00	2.00	1.00	3.00	1.00
Betweenness	4.00	5.00	0.00	0.00	1.00	0.00
Closeness	0.14	0.17	0.11	0.09	0.14	0.10
Eigenvector	0.80	1.00	0.69	0.29	0.90	0.36

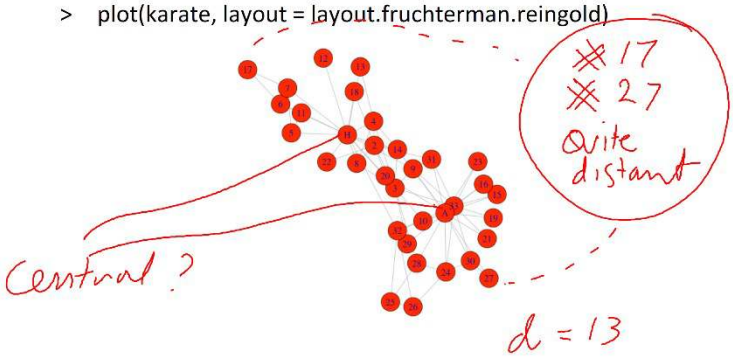
Biggest on all measures

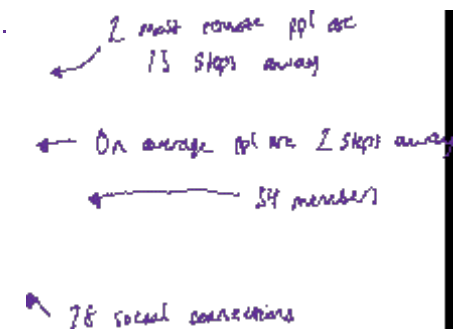
Karate club – looking at the data

> library(igraph)
> library(igraphdata)
> data(karate)
> diameter(karate)
[1] 13
> average.path.length(karate)
[1] 2.4082
> V(karate)
+ 34/34 vertices, named: ...
> E(karate)
+ 78/78 edges (vertex names): ...

Karate club – force-directed plot

> plot(karate, layout = layout.fruchterman.reingold)





Karate club – vertex statistics

Important!

Actor	Degree	Closeness	Betweenness
Mr Hi	16	0.0077	250.1
Actor 2	9	0.0061	33.8
Actor 3	10	0.0060	36.6
Actor 4	6	0.0053	1.3
Actor 5	3	0.0046	0.5
Actor 6	4	0.0046	15.5
Actor 7	4	0.0047	15.5

Actor	Degree	Closeness	Betweenness
Actor 18	2	0.0058	16.1
Actor 19	2	0.0057	3.0
Actor 20	3	0.0075	127.1
Actor 21	2	0.0062	0.0
Actor 22	2	0.0053	0.0
Actor 23	3	0.0048	10.8
Actor 24	5	0.0042	1.0

UKfaculty – sorted by centrality

```
> hdeg = summ[order(-deg),]
> head(hdeg)
   index deg  bet      clo   eig
C1    29  62  684 0.00452 1.000
K1    37  54 1223 0.00450 0.317
J2    62  43  966 0.00503 0.308
E      5  38  727 0.00450 0.126
K1    37  54 1223 0.00450 0.317
J2    62  43  966 0.00503 0.308
```

```
> hclo = summ[order(-clo),]
> head(hclo)
   index deg  bet      clo   eig
J2    62  43  966 0.00503 0.308
C1    29  62  684 0.00452 1.000
E      5  38  727 0.00450 0.126
K1    37  54 1223 0.00450 0.317
```

Actor 10	2	0.0058	7.3
Actor 11	3	0.0053	0.5
Actor 12	1	0.0044	0.0
Actor 13	2	0.0062	0.0
Actor 14	5	0.0058	1.2
Actor 15	2	0.0052	0.0
Actor 16	2	0.0042	0.0
Actor 17	2	0.0033	0.0

Actor 27	2	0.0051	0.0
Actor 28	4	0.0047	6.5
Actor 29	3	0.0061	10.1
Actor 30	4	0.0053	0.0
Actor 31	4	0.0053	3.0
Actor 32	6	0.0063	66.3
Actor 33	12	0.0061	38.1
John A	17	0.0076	209.5

What can we conclude about the structure of this network? Who are the most important people?

important?!

```
E      5  38  727 0.00450 0.126
>
> hbet = summ[order(-bet),]
> head(hbet)
   index deg  bet     clo   eig
K1    37  54 1223 0.00450 0.317
J2    62  43  966 0.00503 0.308
E      5  38  727 0.00450 0.126
B      2  36  711 0.00368 0.452
C1    29  62  684 0.00452 1.000
L1    38  21  629 0.00398 0.057
```

```
L1    38  21  629 0.00398 0.057
>
> heig = summ[order(-eig),]
> head(heig)
   index deg  bet     clo   eig
C1    29  62 684.2 0.00452 1.000
E1    31  35 109.8 0.00313 0.822
U     21  38 143.3 0.00254 0.740
A3    79  22  70.4 0.00281 0.590
I1    35  28 263.6 0.00287 0.587
S     19  24  16.5 0.00257 0.515
```

L6: REGRESSION

Regression models the relationship between 2+ variables, from which we can:

- Observe the **effect of independent variables (inputs) on the dependent variable (output)**
- **Predict values for new data** (e.g. forecasting)
- Determine the **relative importance of variables** in the model
- Linear regression assumes a straight-line relationship

Fitting a regression model is a form of **supervised learning**

- **Model is "learned" from data consisting of known inputs and outputs**
- Learned model can be applied to unknown cases, this includes forecasting

LINEAR REGRESSION:

Linear regression tells the following:

- Linear relationship between *y* and *x*
- Strength of relationship (predictability)

Simple least squares regression assumes:

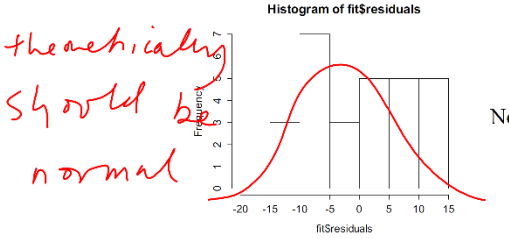
1. Relationship is **approximately linear**, which is form: $y \approx ax + b$
2. ***x*** and ***y*** are **numerical variables,** not categories for example
3. ***a*** and ***b*** are **calculated to minimise the squared error** between the observed values (the data) and the fitted values (those predicted by the model)
4. Errors are (approximately) **normally distributed**

REGRESSION DIAGNOSTICS:

Ideally, **residuals should be normally distributed**

Residuals should be **uncorrelated with input**

```
> hist(fit$residuals)
```



Not conclusive!

In "summary(fit)":

- Median residuals should = 0
- Max and min should be same distance from 0 in positive and negative direction

If a coefficient = 0 → Not important to the model

P-value close to 0 → Coefficient is important

P-value: Probability of obtaining the value of the test statistic (coefficient) if null hypothesis was true (that is, coefficient = 0)

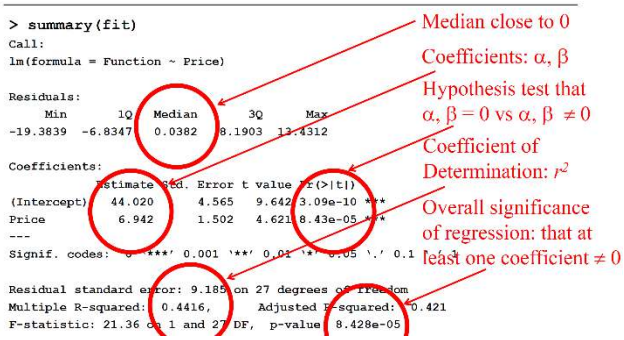
- If p-value close to 0 → Coefficient is significant in the regression model

Coefficient of Determination (R^2): Amount of variation in ***y*** in explained by ***x***

Overall p-value of model: Overall significance of regression that at least one coefficient ≠ 0

- Is there something in the regression having a significant effect

Diagnostics – summary



Prediction:

Linear model object can be used to calculate other fitted values such as forecasts as well as confidence and prediction intervals

MULTIPLE LINEAR REGRESSION:

OLS applied to multiple predictions, assumptions:

- Relationship is now of the **form**

$$y \approx a_1x_1 + a_2x_2 + a_3x_3 + \ldots + b, \text{ or}$$
$$y = a_1x_1 + a_2x_2 + a_3x_3 + \ldots + b + e, \text{ where } e \sim N(\mu, \sigma^2)$$

- ***x*** and ***y*** are **numerical variables**. We consider categories in ***x*** next
- a_i and ***b*** are calculated to **minimise the squared error** between the observed values (the data) and the fitted values (those predicted by the model)
- Errors are (approximately) **normally distributed**

Model: 2 predictors

Using only two input variables: cement and water:

```
> Concrete <- read.csv("Concrete_regression.csv")
> attach(Concrete)
> fit <- lm(Strength ~ Cement + Water)
> fit

Call:
lm(formula = Strength ~ Cement + Water)

Coefficients:
(Intercept)       Cement        Water
    49.9699       0.0763      -0.1961
```

$y = a x_{i1} + a_2 x_2 + \varepsilon$

-ve coefficient

more water = weaker cement

different units scale not important

Summary

large standard error test statistics (t values)

```
> summary(fit)
Call:
lm(formula = Strength ~ Cement + Water)

Residuals:
    Min      1Q  Median      3Q     Max
-36.60  -10.76    0.00    9.46   41.57

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 49.96990    3.98731   12.53  <2e-16 ***
Cement       0.07631    0.00416   18.36  <2e-16 ***
Water       -0.19612    0.02034   -9.64  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.9 on 1027 degrees of freedom
Multiple R-squared: 0.31,       Adjusted R-squared: 0.309
F-statistic:  231 on 2 and 1027 DF,  p-value: <2e-16
```

min, max, med all ok

model explains 31% of variance in Strength

all coeff sig

model sig overall

Smaller t-values → Can't demonstrate significance for these variables

REGRESSION WITH QUALITATIVE VARIABLES:

Qualitative (or categorical) predictors include gender, hair/eye colour, season, job type, etc.

- For these datasets there is **no sense in creating a linear predictor with numerical values**
- Simplest method is to **use an indicator (0, 1) matrix** to indicate a category

Person	Eye.colour
A	Blue
B	Brown
C	Green
D	Blue
E	Blue

--->

Person	Eye.Blue	Eye.Brown	Eye.Green
A	1	0	0
B	0	1	0
C	0	0	1
D	1	0	0
E	1	0	0

indicator columns

Other types of regression

There are many other regression models in addition to those covered today. Some examples from ATHR P65.

Model	Formula	
$y = \beta_0 + \beta_1 x + e$	`y ~ x`	Simple regression
$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + e$	`y ~ x1+x2`	Multiple regression
$y = \beta_0 + e$	`y ~ 1`	Intercept only (null) model
$y = \beta_1 x + e$	`y ~ 0+x`	Slope only
$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + e$	`y ~ x1*x2`	Main effects and products
	`y ~ x1+x2+x1:x2`	
$y = \beta_0 + \beta_1 x + \beta_2 x^2 + e$	`y ~ x+I(x^2)`	Quadratic term
$ln(y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + e$	`log(y) ~ x1+x2`	Log dependent

L7: DECISION TREES

Machine Learning:

Automated (statistical) learning of a concept form labelled sample data

- Spam filtering with an algorithm that takes some examples of spam and makes a rule to predict whether an email should go to the spam folder

How can a model learn a concept?

- **Descriptive:** Captures the training data;
- **Predictive:** Generalizes to unseen data;
- **Explanatory:** Describes the concept to be learned

INTRO TO CLASSIFICATION AND DECISION TREES:

CLASSIFICATION:

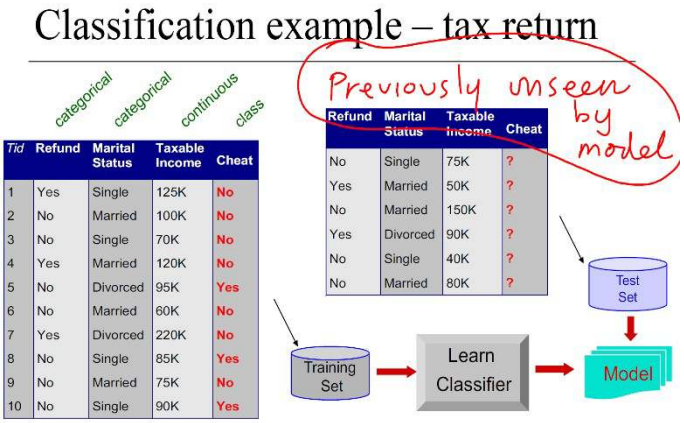
Using a collection of records containing a set of *attributes* where 1 of the attributes is the *class*, find a model to predict class a function other the other attributes

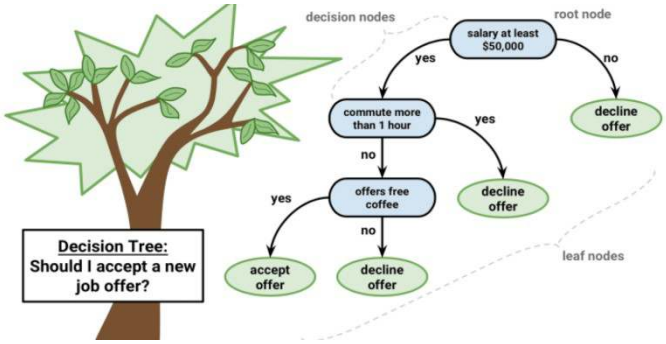
- **Goal:** Previously unseen records should be assigned a class as accurately as possible
- Data is usually divided into ***training set*** to build the model, and a ***test set*** used to validate (test the accuracy) of the model

Predict class based on other attributes

Classification techniques include:

Decision Tree based methods, Naïve Bayes/Bayesian Belief Networks, Ensemble methods, Artificial neural networks, Rule-based methods, Memory based reasoning, Support Vector Machines





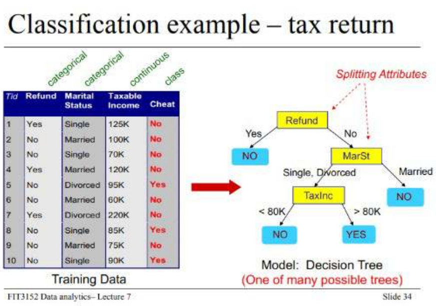
DECISION TREES:

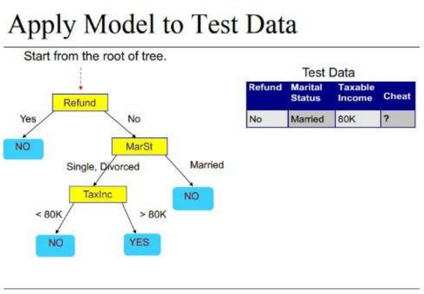
Decision trees are one of the most widely used and practical methods in machine learning:

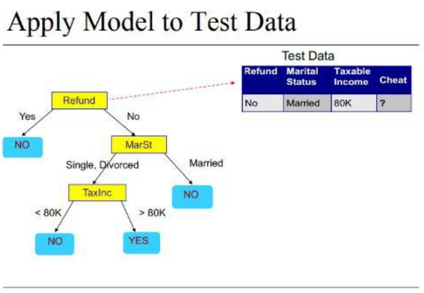
- Model uses **existing data** attributes and values
- Can be used to **classify new instances**
- Can be used to **profile existing data**
- **Robust to noises and missing values**
- Each tree can be viewed as sequence of **"if-then-else"** statements; This readability is highly desirable
- We can construct a simple decision trees by hand

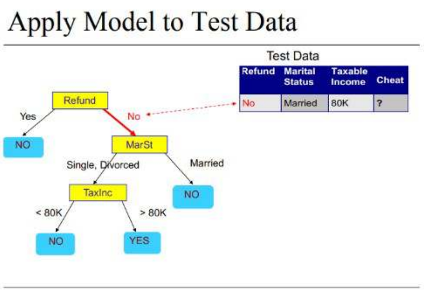
Tree constitutes:

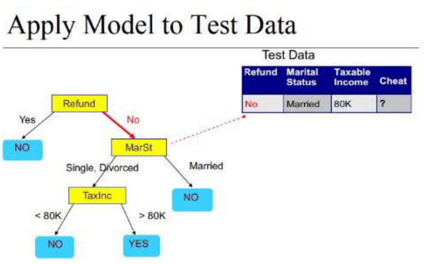
- **Leaf nodes** (class; final decision) and **non-leaf nodes** (decision attributes)
- **Branches** – Corresponding to the values of the decision attributes having either binary or multi-way splits
- **To classify an object:** Each decision node (starting from the root) compares an attributes of the object with a specific attribute value (or range) and takes the corresponding branch
- A **path** from the root to a leaf node gives the class of the object

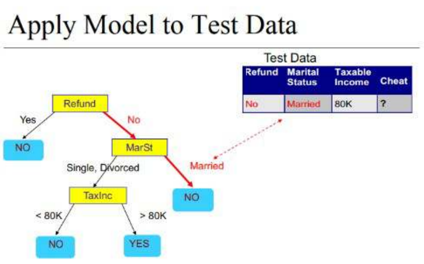


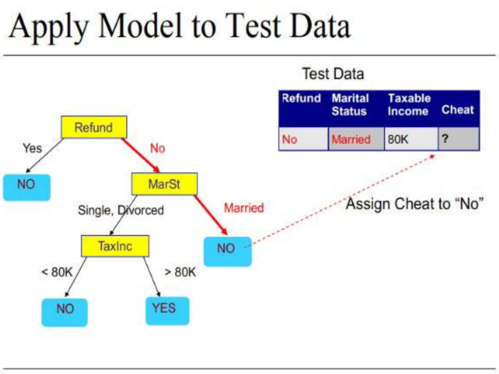












Many decision tree learning algorithms are variations on a core algorithm that *employs top-down, greedy search* through the space of possible decision trees

- Start with decision that simplifies the data the quickest

The algorithm aims to create **homogenous leaf nodes**

- Breaks data into pure groups in the end

SPECIFIC DECISION TREE ALGORITHM: ID3

The algorithm (Iterative Dichotomiser 3):

- At each step, **determine the "best" decision attribute**, ***A***, for next node
- Assign ***A*** as decision attribute for node
- For each value of ***A***, create new descendant
- **Sort training examples to that node** according to the attribute value of the branch
- If all **training examples are perfectly classified** (same value of target attribute) → Stop
    - Else, iterate over new leaf nodes

For this algorithm, assume **class attribute is categorical**

At each stage of the process we **try to find the "best" attribute and split to partition the data**

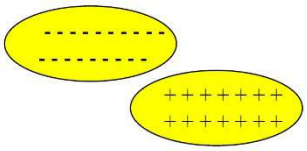
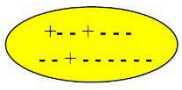
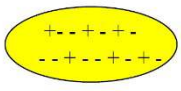
That decision **may not be the best overall** – but once it is made we stay with it for the rest of tree

- Generally called a ***greedy* approach** and may not result in best overall decision tree

At each split the **goal is to ↑ the homogeneity of the resulting datasets** with respect to the class or target variable (which we are trying to classify)

Homogeneity:

Suppose we have a binary targe attribute with values **"+"** and **"-"**:

- These two sets are homogeneous 
- This one is not  more -ves
- This one even less so  +, − about equal

ENTROPY AND INFORMATION GAIN:

Goal is to ↑ Information gain, ↓ Entropy

Information gain:

Which attribute to choose for splitting?

- **Information gain: Statistical property that measures how well a given attribute separates the training examples into homogenous groups according to target classification**
- **ID3** uses **information gain as the splitting criteria for building a tree** and chooses the attribute which provides the greatest information gain
- Information gain is determined using a measure from Information Theory called *Entropy*

ENTROPY:

Measure of disorder in a system; The aim is to reduce it

In information theory:

- **Entropy: Measures the uncertainty in a random variable – or message, or indicates how much information (or impurity) there is in an event**
- In general, the more uncertain or random the event is, the more information it will contain

Calculating entropy:

For a 2-class problem c_1 and c_2:

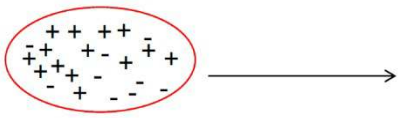
- ***P*** = Probability of belonging to each class
- Number in each class is $N_{c1} + N_{c2} = N$

$$\begin{aligned} \text{Entropy(S)} &= -\mathrm{P}_{c1}\log_2(\mathrm{P}_{c1}) - \mathrm{P}_{c2}\log_2(\mathrm{P}_{c2}) \\ &= -\frac{\mathrm{N}_{c1}}{\mathrm{N}}\log_2(\frac{\mathrm{N}_{c1}}{\mathrm{N}}) - \frac{\mathrm{N}_{c2}}{\mathrm{N}}\log_2(\frac{\mathrm{N}_{c2}}{\mathrm{N}}) \end{aligned}$$

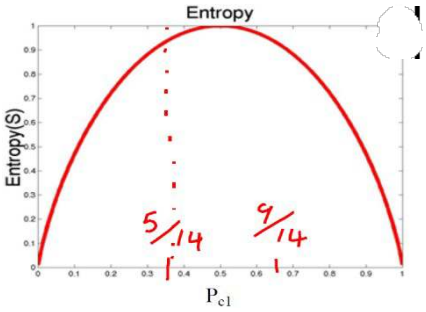
For a multi-class problem:

$$\begin{aligned} \text{Entropy(S)} &= -\sum_{i=1}^{C}\mathrm{P}_i\log_2(\mathrm{P}_i) \\ &= -\sum_{i=1}^{C}\frac{\mathrm{N}_i}{\mathrm{N}}\log_2(\frac{\mathrm{N}_i}{\mathrm{N}}) \end{aligned}$$

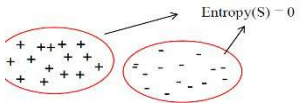
E.g. Suppose ***S*** is a collection of 14 examples, 9 positive and 5 negatives → [9+,5-]



$$\begin{aligned} \text{Entropy(S)} &= -P_{c1}\log_2(P_{c1}) - P_{c2}\log_2(P_{c2}) \\ &= -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) \\ &= 0.940 \end{aligned}$$



E.g. Suppose ***S*** has all positive or all negative examples



Entropy = 0 (minimum) if all members belong to the same class

Entropy = 1 (maximum) if the collection consists of equal number of positive/negatives examples

Note: $0\log_2 0 = 0$

Previous example as a spreadsheet:

- **If your calculator can't work out logs to base 2 then use the following:**

$$\log_2(x) = \frac{\log_{10}(x)}{\log_{10}(2)} \approx \frac{\log_{10}(x)}{0.3010}$$

Use my calc for this!

Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
9	5	0.6429	-0.6374	0.3571	-1.4854	0.9403

Note: $\log_2$

Information Gain:

Information gain: Expected ↓ in entropy caused by partitioning examples according to attribute ***A***

- ***Gain(S, A)*** of an attribute ***A***, relative to a collection of example ***S*** (with ***v*** groups having $|S_v|$ elements) is:

$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v)$$

Entropy before split — Expected entropy after split

How ID3 uses information gain:

The algorithm by 'splits' on the attribute that provides the most information gain – i.e. gives the purest class breakdown at each step in the decision tree

Recall: *Purer class = Entropy reduction!*

The algorithm:

- At each step, **determine the "best" decision attribute**, ***A***, for next node
- Assign ***A*** as decision attribute for node
- For each value of ***A***, create new descendant
- **Sort training examples to that node** according to the attribute value of the branch
- If all **training examples are perfectly classified** (same value of target attribute) → Stop
    - Else, iterate over new leaf nodes

Terminology:

- *Instance:* Single row in a data set. Also called an example or object
- *Attribute:* Aspect of an instance which can be categorical or numerical. Also called variable
- *Value:* Category that an attribute can take
- *Concept:* Thing to be learned. Also called *Class* or *Target*

Example: Playing Tennis

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

class

using class only

Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
9	5	0.6429	-0.6374	0.3571	-1.4854	0.9403

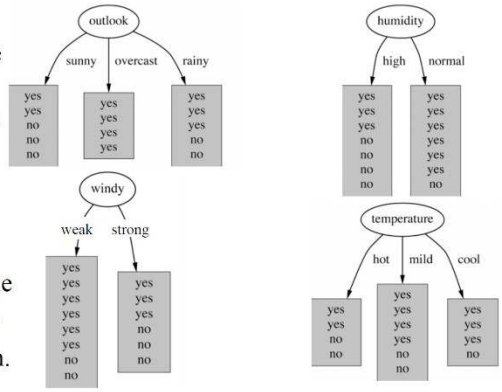
Without any knowledge of the weather there are 9 Yes and 5 No cases. Below is **initial entropy:**

$$E(S) = -\frac{9}{14} \cdot log_2\left(\frac{9}{14}\right) - \frac{5}{14} \cdot log_2\left(\frac{5}{14}\right)$$
$$E(S) = 0.9403$$

Which attribute to select?

Remember - ID3 chooses the attribute which gives the greatest information gain (reduction in Entropy), or the 'purest' result.

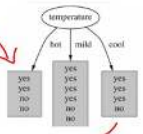
We next calculate the information gain for each attribute in turn.



Information gain: Temperature

Calculate entropy for each branch first:

- $E(S_{hot}) = -\frac{2}{4} \cdot log_2\left(\frac{2}{4}\right) - \frac{2}{4} \cdot log_2\left(\frac{2}{4}\right) = 1$
- $E(S_{mild}) = -\frac{4}{6} \cdot log_2\left(\frac{4}{6}\right) - \frac{2}{6} \cdot log_2\left(\frac{2}{6}\right) = 0.918$
- $E(S_{cool}) = -\frac{3}{4} \cdot log_2\left(\frac{3}{4}\right) - \frac{1}{4} \cdot log_2\left(\frac{1}{4}\right) = 0.811$



Now calculate expected entropy and information gain

- $Gain(S, Temp) = E(S) - E(S, Temp)$
- $Gain(S, Temp) = E(S) - \left(\frac{4}{14}1 + \frac{6}{14}0.918 + \frac{4}{14}0.811\right)$
- $= 0.9403 - 0.910$
- $= 0.0292$

4 cool
14 total
entropy after
entropy before
info gain

Information gain: Temperature

As a spreadsheet showing initial entropy and subsequent information gain:

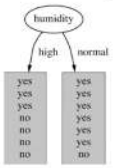
Initial State	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Entropy(S)	9	5	0.6429	-0.6374	0.3571	-1.4854	0.9403

Temperature	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Hot	2	2	0.5000	-1.0000	0.5000	-1.0000	1.0000
Mild	4	2	0.6667	-0.5850	0.3333	-1.5850	0.9183
Cool	3	1	0.7500	-0.4150	0.2500	-2.0000	0.8113
EEntropy(Temp)							0.9111
Gain(S, Temp)							**0.0292**

Information gain: Humidity

Calculate entropy for each branch first:

- $E(S_{high}) = -\frac{3}{7} \cdot log_2\left(\frac{3}{7}\right) - \frac{4}{7} \cdot log_2\left(\frac{4}{7}\right) = 0.9852$
- $E(S_{normal}) = -\frac{6}{7} \cdot log_2\left(\frac{6}{7}\right) - \frac{1}{7} \cdot log_2\left(\frac{1}{7}\right) = 0.5917$



Now calculate expected entropy and information gain

- $Gain(S, Humidity) = E(S) - E(S, Humidity)$
- $Gain(S, Humidity) = E(S) - \left(\frac{7}{14}0.9852 + \frac{7}{14}0.5917\right)$
- $= 0.9403 - 0.7885$
- $= 0.1518$

info gain

Calcs: Humidity, Wind, Outlook

Humidity	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
High	3	4	0.4286	-1.2224	0.5714	-0.8074	0.9852
Normal	6	1	0.8571	-0.2224	0.1429	-2.8074	0.5917
EEntropy(Humidity)							0.7885
Gain(S, Humidity)							**0.1518**

Wind	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Weak	6	2	0.7500	-0.4150	0.2500	-2.0000	0.8113
Strong	3	3	0.5000	-1.0000	0.5000	-1.0000	1.0000
EEntropyWind)							0.8922
Gain(S, Wind)							**0.0481**

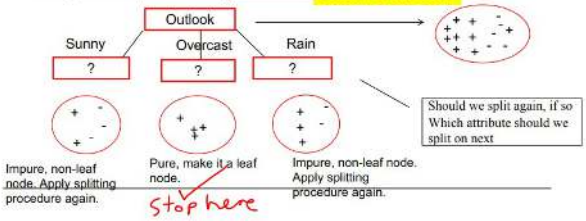
Outlook	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Sunny	2	3	0.4000	-1.3219	0.6000	-0.7370	0.9710
Overcast	4	0	1.0000	0.0000	0.0000	0.0000	0.0000
Rain	3	2	0.6000	-0.7370	0.4000	-1.3219	0.9710
EEntropy(Outlook)							0.6935
Gain(S, Outlook)							**0.2467**

Greatest info gain

Attribute giving greatest information gain

Which attribute to choose? Outlook, Temperature, Humidity or Wind?

Gain(S, Temperature) = 0.029
Gain(S, Humidity) = 0.151
Gain(S, Wind) = 0.048
Gain(S, Outlook) = 0.247 ← Choose this one!



stop here

Which attribute to split on next?

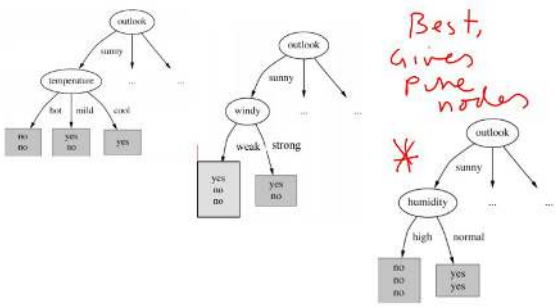
Now, starting with Sunny, which attribute should be split on next? Temperature, Humidity or Wind?

ID3 Step 2: gain(S_sunny, ???)

Now consider subset corresponding to "Sunny":

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
~~D3~~	~~Overcast~~	~~Hot~~	~~High~~	~~Weak~~	~~Yes~~
~~D4~~	~~Rain~~	~~Mild~~	~~High~~	~~Weak~~	~~Yes~~
~~D5~~	~~Rain~~	~~Cool~~	~~Normal~~	~~Weak~~	~~Yes~~
~~D6~~	~~Rain~~	~~Cool~~	~~Normal~~	~~Strong~~	~~No~~
~~D7~~	~~Overcast~~	~~Cool~~	~~Normal~~	~~Strong~~	~~Yes~~
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
~~D10~~	~~Rain~~	~~Mild~~	~~Normal~~	~~Weak~~	~~Yes~~
D11	Sunny	Mild	Normal	Strong	Yes
~~D12~~	~~Overcast~~	~~Mild~~	~~High~~	~~Strong~~	~~Yes~~
~~D13~~	~~Overcast~~	~~Hot~~	~~Normal~~	~~Weak~~	~~Yes~~
~~D14~~	~~Rain~~	~~Mild~~	~~High~~	~~Strong~~	~~No~~

Which attribute to split on next?



Entropy after "Outlook"

The entropy of each branch of the decision tree after split on Outlook is shown below.

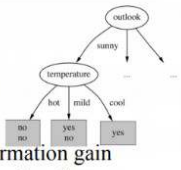
- Information gain in descendent trees is now measured as change in the entropy of each branch
- For example Entropy(Sunny) = 0.971

Outlook	Yes	No	P(Yes)	log2(Yes)	P(No)	log2(No)	Entropy
Sunny	2	3	0.400	-1.322	0.600	-0.737	0.971
Overcast	4	0	1.000	0.000	0.000	0.000	0.000
Rain	3	2	0.600	-0.737	0.400	-1.322	0.971
EEntropy(Outlook)							0.694

ID3 Step 2: Gain Sunny, Temperature

Calculate entropy for each branch first:

- $E(S_{sunny,hot}) = -\frac{0}{2} \cdot log_2\left(\frac{0}{2}\right) - \frac{2}{2} \cdot log_2\left(\frac{2}{2}\right) = 0$
- $E(S_{sunny,mild}) = -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = 1$
- $E(S_{sunny,cool}) = -\frac{1}{1} \cdot log_2\left(\frac{1}{1}\right) - \frac{0}{1} \cdot log_2\left(\frac{0}{1}\right) = 0$



Now calculate expected entropy and information gain

- $Gain(S, Sunny, Temp) = E(S, Sunny) - E(S, Sunny, Temp)$
- $Gain(S, Sunny, Temp) = E(S, Sunny) - \left(\frac{2}{5}0 + \frac{2}{5}1 + \frac{1}{5}0\right)$
- $= 0.971 - 0.4 = 0.571$

Calculations for all attributes shown on the next slide…

ID3 Step 2: Gain for Sunny Outlook

Sunny, Temp	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Hot	0	2	0.0000	0.0000	1.0000	0.0000	0.0000
Mild	1	1	0.5000	-1.0000	0.5000	-1.0000	1.0000
Cool	1	0	1.0000	0.0000	0.0000	0.0000	0.0000
EEntropy(Temp)							0.4000
Gain(Sunny, Temp)							**0.5710**

Sunny, Humid	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
High	0	3	0.0000	0.0000	1.0000	0.0000	0.0000
Normal	2	0	1.0000	0.0000	0.0000	0.0000	0.0000
EEntropy(Temp)							0.0000
Gain(Sunny, Humid)							**0.9710**

Greatest Info gain

Sunny, Wind	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Weak	1	2	0.3333	-1.5850	0.6667	-0.5850	0.9183
Strong	1	1	0.5000	-1.0000	0.5000	-1.0000	1.0000
EEntropyWind)							0.9510
Gain(Sunny, Wind)							**0.0200**

ID3 Step 2: Gain for Sunny Outlook

Which attribute to choose? Temperature, Humidity or Wind?

- $Gain(S_{sunny}, Wind) = 0.020$
- $Gain(S_{sunny}, Humidity) = 0.971$ ← Choose this one!
- $Gain(S_{sunny}, Temperature) = 0.570$

Repeat the process to find which attribute is the best to split on at this step

Not all leaves need to be 'pure'.
Splitting stops when the data can't be split any further.

Class activity

Class counts and expected entropy after rain outlook.

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Rain, Temp	Yes	No
Hot	0	0
Mild	2	1
Cool	1	1

Rain, Humid	Yes	No
High		
Normal		

Rain, Wind	Yes	No
Weak	3	0
Strong	0	2

Pure Node

ID3 Step 3: Gain for Rain Outlook

Rain, Temp	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Hot	0	0	0.0000	0.0000	0.0000	0.0000	0.0000
Mild	2	1	0.6667	-0.5850	0.3333	-1.5850	0.9183
Cool	1	1	0.5000	-1.0000	0.5000	-1.0000	1.0000
EEntropy(Temp)							0.9510
Gain(Rain, Temp)							**0.0200**

Rain, Humid	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
High	1	1	0.5000	-1.0000	0.5000	-1.0000	1.0000
Normal	2	1	0.6667	-0.5850	0.3333	-1.5850	0.9183
EEntropy(Temp)							0.9510
Gain(Rain, Humid)							**0.0200**

Rain, Wind	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Weak	3	0	1.0000	0.0000	0.0000	0.0000	0.0000
Strong	0	2	0.0000	0.0000	1.0000	0.0000	0.0000
EEntropyWind)							0.0000
Gain(Rain, Wind)							**0.9710**

The Final tree

The process of selecting a new attribute and partitioning the training examples is repeated for each non-leaf node, this time only using the examples associated with that node.
Attributes that have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any path through the tree.
The process continues for each leaf node until:

- every attribute has been included along that path through the tree or
- the training examples associated with this leaf node all have the same class.

The Decision Tree Rules

In addition to generating a tree structure, explicit rules for classifying 'play/don't play' are also generated:

If outlook = Overcast Then Play= Yes {No=0, Yes=4}
If outlook = Rain And wind = Strong Then Play= No {No=2, Yes=0}
If outlook = Rain And wind = Weak Then Play = Yes {No=0, Yes=3}
If outlook = Sunny And humidity = High Then Play = No {No=3, Yes=0}
If outlook = Sunny And humidity = Normal Then Play = Yes {No=0, Yes=2}

Further considerations:

Types of decision trees?

- Classification Trees (categorical – nominal attributes)
- Regression Trees (numerical – continuous attributes)
- Depends on target variable type

How to evaluate the performance of a model?

- Training and testing
- Confusion matrix

MODEL ACCURACY; TRAINING AND TESTING:

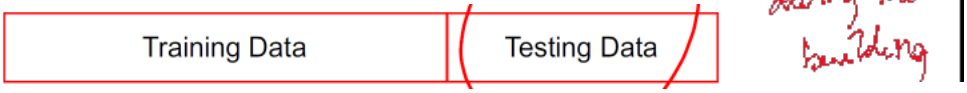
You can measure a classifier's performance in terms of the **error rate** (proportion of errors made over a whole set of instances)

Due to desirability of generalisation, **lower error on the training data is not a good measure**

To [illegible]ct the performance of a classifier of new data → Assess its error on a data set that played no part in its formulation

In gener[illegible] data set is divided into **2 subsets: Training and testing**

- Training for the learning model
- Testing for determining how well it will do on unseen data



PERFORMANCE EVALUATION:

How to determine accuracy of decision tree in classifying/predicting?

- Usual to have 2 data sets:
  - A *training set* and a *test set*
  - This can be created by dividing the data set into 2 sets (70%-30%)
- We create the decision tree model using the training set
- Then run the test set through the model to find out what the predicted class is
- Then compare the predicted class with the actual class to see how accurate the model is

Another way of assessing the performance is to calculate accuracy based on a confusion matrix (for the test data classification)

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

Most widely used metric:

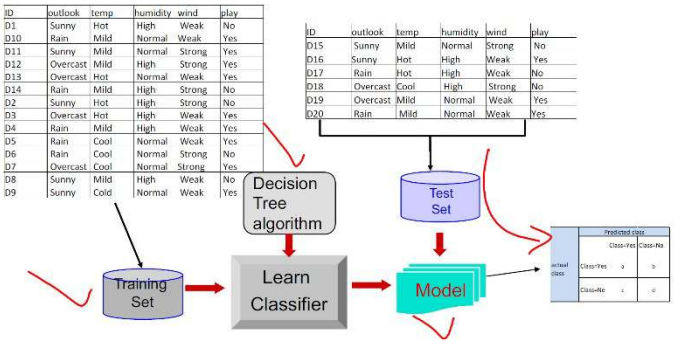
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Also, important:

$$Precision = \frac{TP}{TP + FP}$$

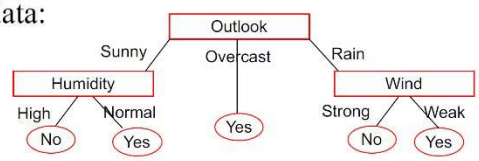
$$Sensitivity = \frac{TP}{TP + FN}$$

The Play Tennis example



The play tennis example

Let's see what our model would predict using the test data:



Day	Outlook	Temperature	Humidity	Wind	Play	Predict
D15	Sunny	Mild	Normal	Strong	No	yes
D16	Sunny	Hot	High	Weak	Yes	no
D17	Rain	Hot	High	Weak	No	yes
D18	Overcast	Cool	High	Strong	No	yes
D19	Overcast	Mild	Normal	Weak	Yes	yes
D20	Rain	Mild	Normal	Weak	Yes	yes

Metrics for Performance Evaluation…

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	2 (TP)	1 (FN)
	Class=No	3 (FP)	0 (TN)

Most widely-used metric:

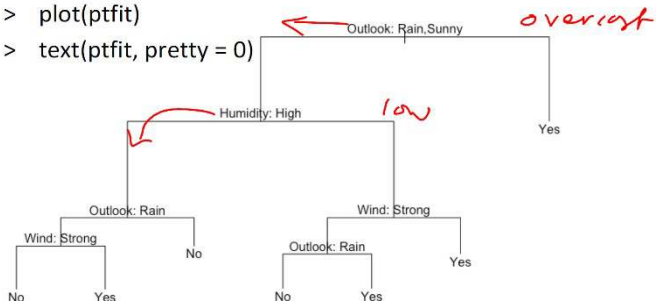
$$Accuracy = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{2 + 0}{6} = 33.3\%$$

DECISION TREES IN R:

Classification tree: plot

Headers give rule for left branching.

```
> plot(ptfit)
> text(ptfit, pretty = 0)
```



Classification tree: testing the model

To test the model, make a prediction for each and draw the confusion matrix.

```
> table(observed = iris.test$Species, predicted = ipredict)
            predicted
observed     setosa versicolor virginica
  setosa         13          0         0
  versicolor      0         14         3
  virginica       0          1        14
```

L8: NAÏVE BAYES

IMPROVING BASIC DECISION TREE:

Pros:

- **Easy to interpret** – explicit rules + graphical representation
- Can handle **mixed input** (discrete and continuous)
- **Robust** – to outliers (noise) and missing values
- Able to find the **most discriminating attributes**
- **Accuracy good** – in general

Cons:

- **Unstable:** Small changes may lead to a completely different tree
- Can become **overly complex**

PERFORMANCE METRICS:

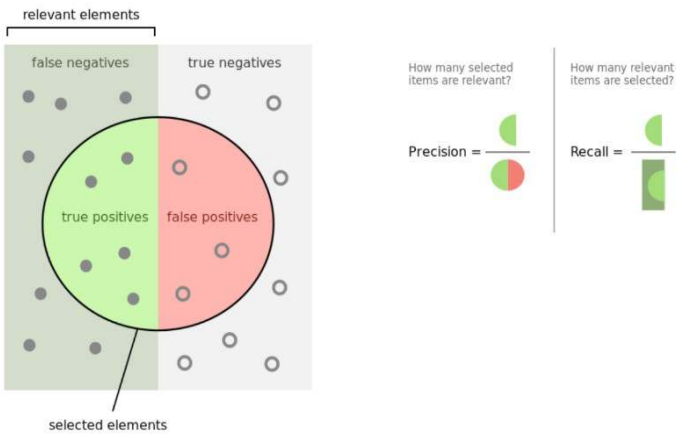
Precision: How good is the model at picking true positives from everything it classifies as positives

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Recall: Proportion of true positives from all that is actually a positive

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Metrics - Precision & Recall



Review: Performance Evaluation…

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	2 (TP)	1 (FN)
	Class=No	3 (FP)	0 (TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{2 + 0}{6} = 33.3\%$$

$$Precision = \frac{TP}{TP + FP} = \frac{2}{5} = 40\ \%$$

$$Recall = \frac{TP}{TP + FN} = \frac{2}{3} = 66.7\ \%$$

TREE SIZE AND ACCURACY:

Overfitting – Model that does not generate well

- Model is excessively **complex**
- Performs well on training data but **not on unseen data**
- **Low recall**

Underfitting – Model that is too simple

- Model is **too simple** to give accurate labels
- **Performs poorly** on both training and unseen data
- **Low precision**
    - Low ability to discriminate true positive from all positives

If outlook = Overcast Then Play= Yes
If outlook = Rain And wind = Strong Then Play= No
If outlook = Rain And wind = Weak Then Play = Yes
If outlook = Sunny And humidity = High Then Play = No
If outlook = Sunny And humidity = Normal Then Play = Yes

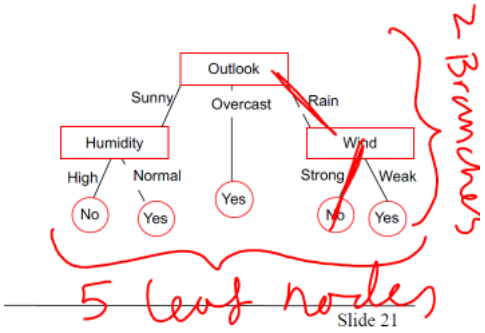
Tree size:

Leaf node count

- Corresponds to the number of rules that are encoded in the decision tree (5 in this example)

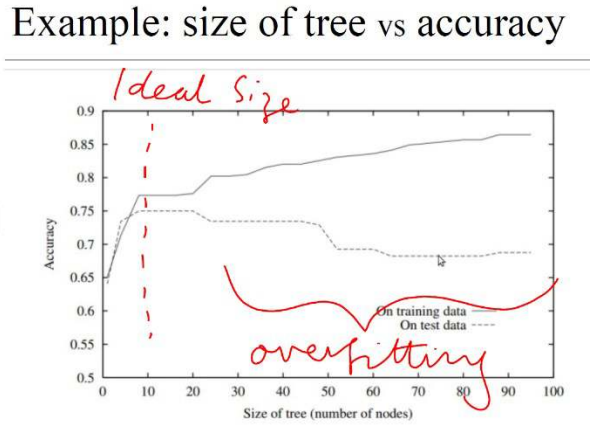
Tree height

- Corresponds to the maximum rule length and number of premises to be evaluated to reach class decision (2 in this example)



Is a tree with only pure leaves always the best classifier you have?

- No. This tree is the best classifier on training set, but possibly not on new and unseen data
- Because of overfitting, tree may not generalize very well



You can grow each branch of the tree deeply enough to perfectly classify the training examples but what about noise in the data or simply irrelevant features?

2 approaches to avoid over-fitting:

1. **Pre-pruning:** Stop growing the tree earlier, before it reaches the point where it over-fits on the training data
2. **Post-pruning:** Allow the tree to overfit the data and then post-prune the tree

PRUNING:

DT Post-Pruning:

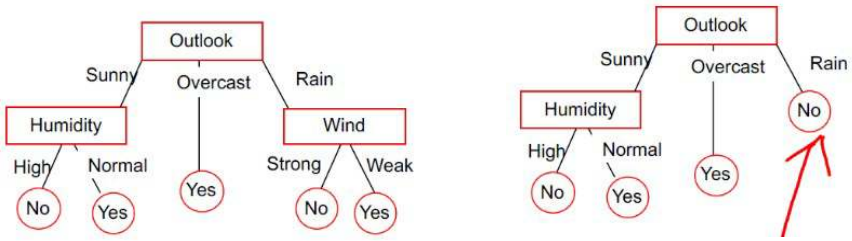
Post-pruning: Decision tree is first grown out as usual

Depth of tree is reduced by replacing sub-trees with leaf nodes

General concept is to remove the rules if they have little or negligible effect on the error rate

DT Post-Pruning Example:

1. Keep a portion of data as **validation set**
2. **Grow tree** out to usual size as per no-pruning
3. **Prune by replacing sub-trees with leaf nodes**, by measuring error against validation set
4. Class label of replacement leaf node determined by majority of labels in the removed sub-tree



CROSS VALIDATION:

Remember: You cannot test on cases that you have already used as part of the training!

- Therefore, you always need distinct and testing data sets

K-fold Cross Validation steps:

- **Partition data into *k*** disjoints subsets
- **Train on *k-1* partitions**, test on the remaining one
- **Repeat** until all partitions have been used to train/test

Popular approach is 10-fold cross validation

Make sure that all folds have same distribution of classes

"Stratification" (stratified sampling) ensures that classes are properly represented across partitions

Leave-one-out CV: Each case is left out, and the model is trained on all the remaining instances. The results across all *n* samples are later averaged to get the final error estimate

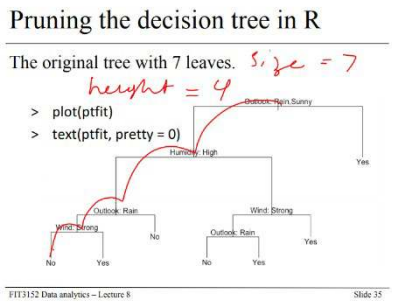
Example: 3-Fold CV

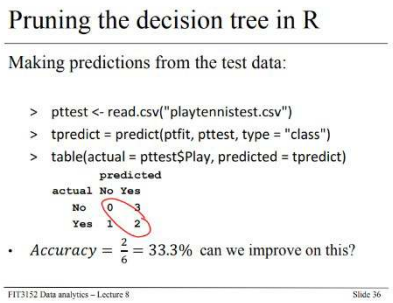
Partitioning:

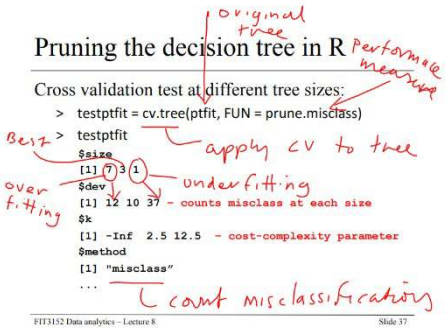
- 2/3 training (making the model)
- 1/3 testing (measuring performance of model)

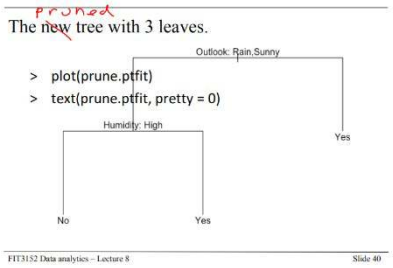
Repeat the procedure 3 times so that every case has been used exactly once for testing

EXAMPLES IN R:





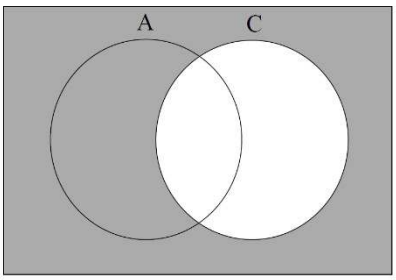




NAÏVE BAYES CLASSIFICATION:

BAYES' THEOREM:

Calculate the conditional probability that ***x*** is a member of ***A*** given that is a member of ***C***:



Calculate the conditional probability that ***x*** has attribute ***A*** given it is of class ***C***:

Bayes Theorem:

$$P(A \vee C) = \frac{P(A \cap C)}{P(C)} \rightarrow P(C) \cdot P(A \vee C) = P(A \cap C)$$

Prior probability of belonging to Class C is ***P(C)***

Obtain extra information: **Probability of having attribute A if a member of C → P(A|C)**

But what is the posterior probability of belonging to class C when having attribute, ***A***, that is, ***P(C|A)***?

Using **Bayes' Theorem**, this is:

$$P(C \vee A) = \frac{P(A \cap C)}{P(A)} = \frac{P(C) \cdot P(A \vee C)}{P(A)}$$

BAYESIAN CLASSIFIERS:

s

INDEPENDENT EVENTS:

s

NAÏVE BAYE'S CLASSIFIER:

s

HAND-WORKED EXAMPLE:

s

NAÏVE BAYES CLASSIFICATION IN R:

s

CLASSIFIER MODEL EVALUATION: ROC CURVES

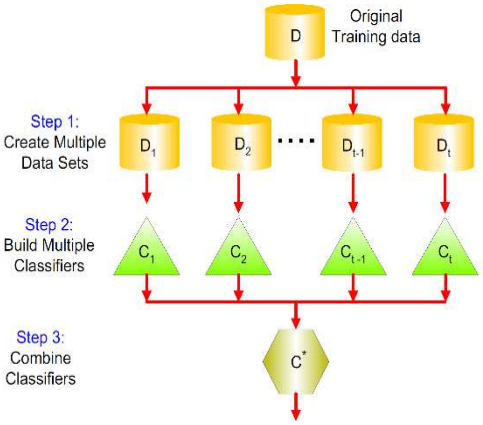
s

LIFT:

s

L9: ENSEMBLE METHODS & ANN

ENSEMBLE METHODS:



To improve classification accuracy:

- Build a collection of "experts" by **constructing a set of classifiers** from the training data
- They could be the same or different types of classifiers
- **Combine the results** of each classifier

This enables the **creation of a better classifier** from a collection of weak classifiers

Ensemble methods work best when:

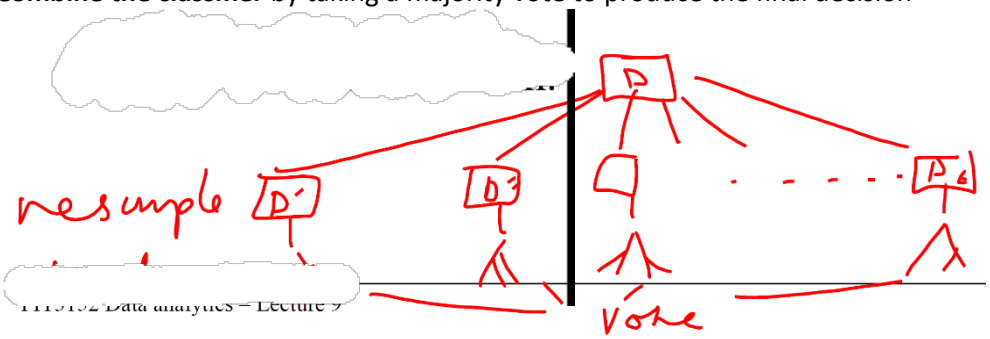
- The individual classifiers are **moderately (> 50%) accurate**
- Individuals classifiers are **not correlated**
- Pooling results of each classifier **reduces the variance** of the overall classification
- Decision trees work well as the individual classifier

Disadvantage: Model produced not as easy to interpret as single tree

BAGGING (BOOSTRAP AGGREGRATION):

Bagging algorithm:

- Make **multiple replicates of the original data by sampling**, with replacement, from the training set. Replicates are the same size as original
- Construct a **single classifier for each replicate**
- **Combine the classifier** by taking a majority vote to produce the final decision



Bootstrapping: Resampling the original data set to produce multiple synthetic data sets

- Ex: Re-sample 100 times play tennis

Sampling is uniform → Each bootstrap replicate may have multiple instances of the original data points and contains approximately 63% of the original data set (each sample is slightly different)

Build a classifier on each of the replicates and combine results by voting

- Multiple classifiers ↓ the (high) variance of individual decision trees (Sample variance is $\frac{\sigma}{\sqrt{n}}$)
- Larger sample = More stable estimates

When to use bagging:

- When there is ***noise in data***
- Useful for ***unstable classifiers*** → Small changes (high variance) in training data cause large changes in the classifier
    - Decision trees, neural networks, linear regression
- Not recommended for stable classifiers
    - K-Nearest Neighbours, Naïve Bayes

BOOSTING:

Multiples trees are grown slowly, using incremental improvement

Data set is not bootstrap sampled, but training examples are weighted, with a higher weight given to hard to classify examples at each step

Classification is by a weighted sum from each classifier, with **more accurate classifiers having a greater weight**

Learning as you go and give higher weight to data points where difficult to classify

Boosting algorithm (Adaptive Boosting):

For each tree:

- Assign **equal weights to each point in training set**; Fit basic tree
- **Repeat *n* iterations: Update weights of misclassified items and normalise;** Update tree, building on current tree
- Output the **final classifier as weighted sum of votes** from each tree

Advantage:

- Enables improved classification of imbalanced data sets (where most instances are of one class)
- Tends to achieve better accuracy than bagging

Disadvantage:

- Can lead to over fitting if the number of tree is too great

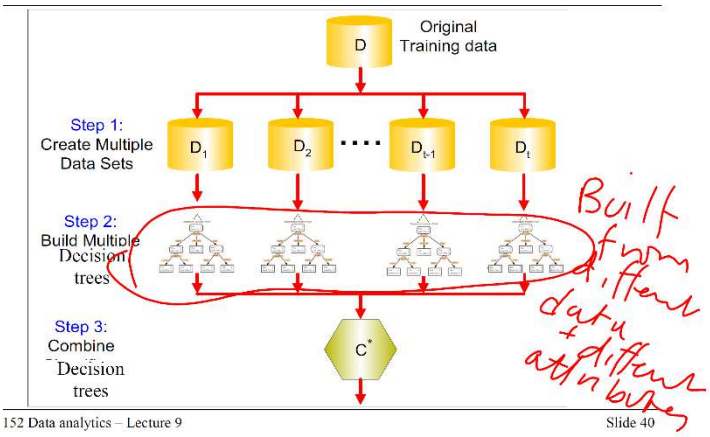
RANDOM FOREST:

Refinement of bagged decision trees

Specifically designed for decision trees

Random Forest Algorithm:

- Create m**ultiple data sets from the original training set** using subsets of data points and subsets of attributes
- **Build a decision tree classifier** for each new data
- Combine the classifiers by taking a **majority vote to produce the final decision**



Advantages:

More accurate than individual trees on large data sets

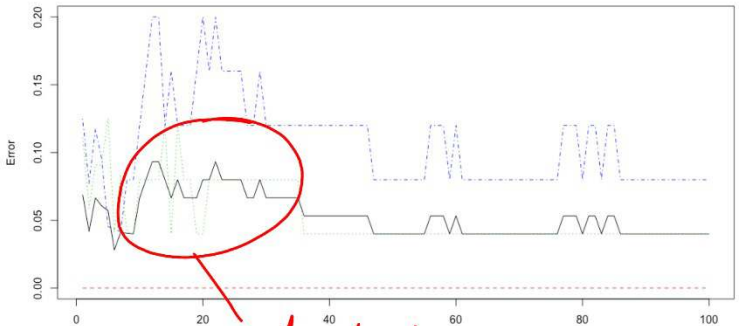
- No need to prune
- Not sensitive to outliers
- Overfitting not a problem

Disadvantages:

- Bagging varies sample data and number of trees
- Random forests varies the attribute used to build tree as well as the sample data and number of trees
- Some trees built without dominant attributes, so effect of minor attributes become evident

How does number of trees affect error?

```
> plot(randomForest(Species ~ ., data=iris[sub,],
keep.forest=FALSE, ntree=100))
```



All of the above models can be further improved by cross validation

Artificial Neural Network:

Have the ability to 'learn' by weighting the contribution of each neuron to a decision (output)

They are accurate, can handle redundant attributes and noisy data

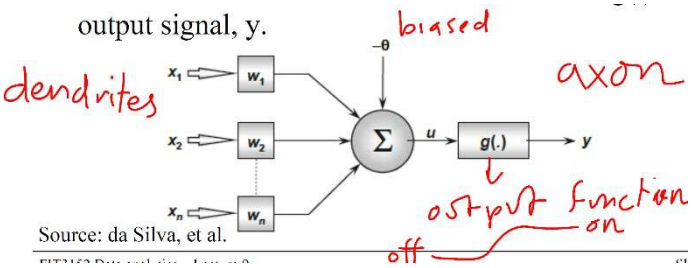
Large ANNs give rise to 'deep learning'

ARTIFICIAL NEURONS:

Artificial neurons aggregate:

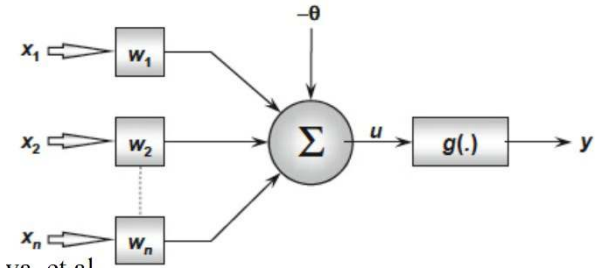
- Weighted (***W_1***, etc.) input signals (***X_1***, etc.)
- Activation threshold bias $(-\theta)$
- To create an activation voltage

This is transmitted via an ***activation function g(.)*** as an ***output signal, y***



Under the McCullogh and Pitts model, the **output by the artificial neurons** can be modelled as:

$$u = \sum_{i=1}^{n} w_i \cdot x_i - \theta \qquad y = g(u)$$



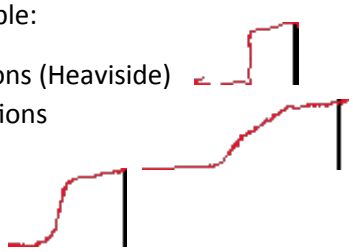
Operation of artificial neuron summarised by following steps:

- **Input** is via a set of variables X_1, … etc.
- These are **multiplied by a synaptic weight** W_1, … etc.
- **Activation potential** is the weighted sum of inputs, with bias subtracted
- An **activation function** interprets the activation potential and limits the output of the neuron

Activation functions:

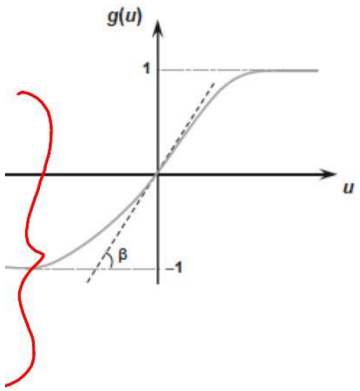
Partially differentiable:

- Step functions (Heaviside)
- Ramp functions



Fully differentiable:

- Logistic
- Hyperbolic Tangent (Tanh)
- Gaussian



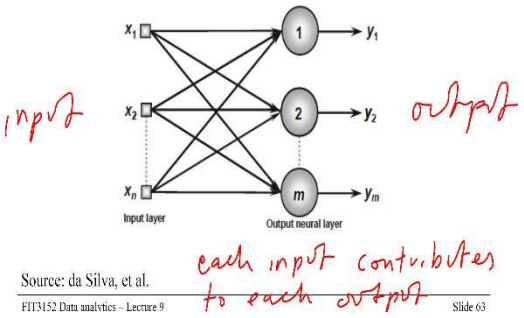
Network architectures:

Structure of the ANN determines the:

- Number of inputs the model can accept
- Number of outputs produced
- Hidden layer determine the complexity of interaction that can be modelled
- Feedback enables "learning"

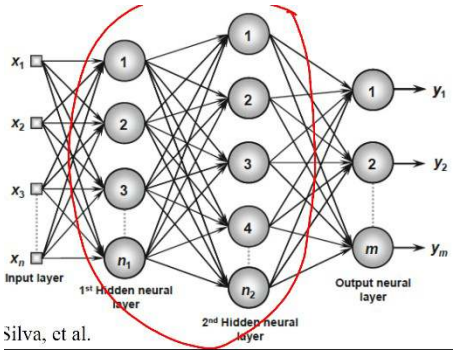
SINGLE LAYER FEEDFORWARD ANN:

n inputs, ***m*** outputs, information flow only in one direction



MULTIPLE LAYER FEEDFORWARD ANN:

n inputs, ***m*** outputs, 2 hidden layers



Hidden layers enable mixing (interaction) between neurons to occur

- Enables the neural network to decode complex, non-linear problems (optimization, pattern recognition, classification)

Number of **hidden layers** determines the complexity of problems the

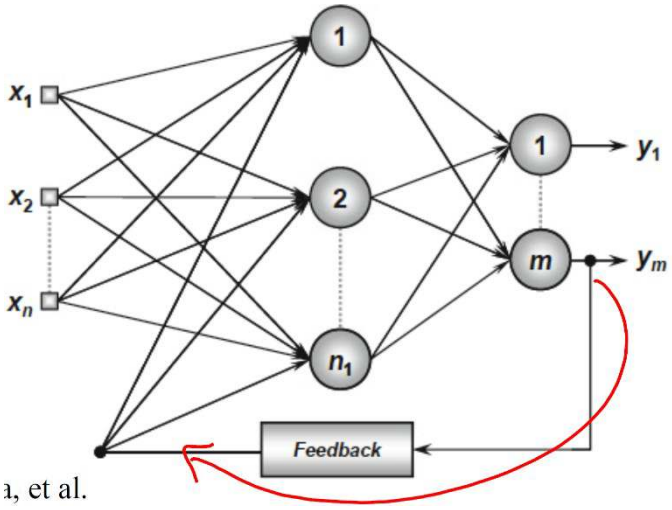
More hidden layers model more complex interactions

RECURRENT OR FEEDBACK ARCHITECTURE:

Outputs of the neurons become inputs for earlier layers

Feedback architectures enable dynamic information processing for time-varying systems.

- This includes time series prediction, optimisation, process control, etc.



TRAINING ANNS:

Requires tuning (adjusting) the weights of each synapse and thresholds of each neuron to **produce output results close to those of the training sets**

- Uses optimisation "to learn" the model

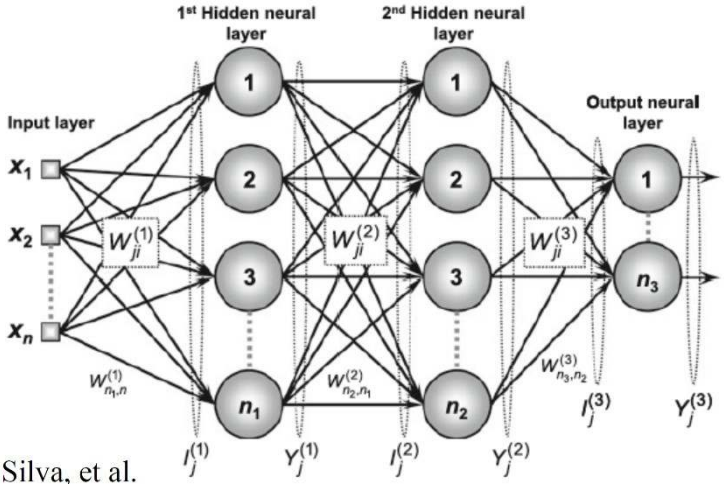
Supervised learning → Procedure is an iterative optimisation to reduce the error between know and predicted outputs

- Answer learned from labelled inputs

Unsupervised learning → The optimisation is more to produce clusters of similar subsets of the data

- Data not labelled, model "finds" answer

Multi-layer ANN showing weights at each synapse



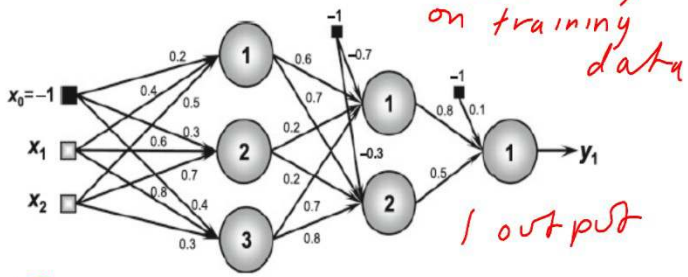
da Silva, et al.

Each weight calculated to

Example of a "trained" ANN showing weights at each synapse and evaluation of input (X_0 = -1)

synapse and evaluation of input ($X_0 = -1$)

max accuracy on training data



Inputs = 3

1 output

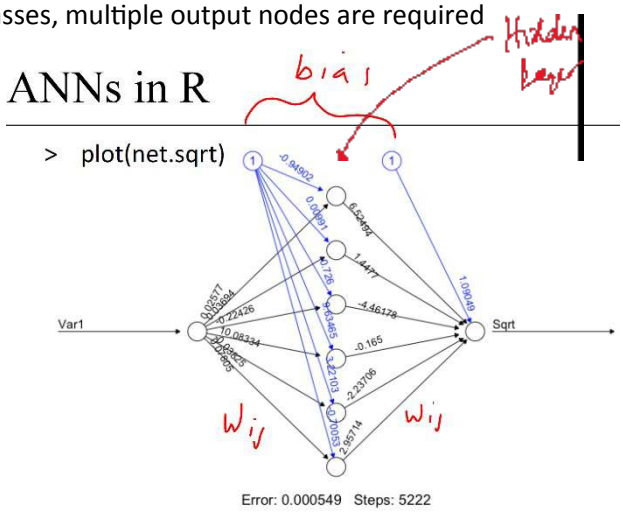
Setting up ANNs:

Pre-processing

- ***1 input*** neuron for each input variable
- ***1 output*** neuron for each output class
- ***Inputs can only be numerical*** (R will accept binary TRUE FALSE)
- Data should be ***normalised***
- ***Categorical data needs to be converted to binary*** columns as indicator variables
- ***No missing values***

ANNs in R:

To predict multiple (n>2) classes, multiple output nodes are required



L10: CLUSTERS

ENSEMBLE METHODS:

To improve classification accuracy:

Build