Supervised Learning Algorithms

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Model selection
- Summary

Objectives

- Learn basic techniques for data classification and prediction.
- Realize the difference between the following classifications of data:
  - supervised classification
  - prediction
  - unsupervised classification

What is Classification?

- The goal of data classification is to organize and categorize data in distinct classes.
  - A model is first created.
  - The model is then used to classify new data.
  - Given the model, a class can be predicted for new data.
- Classification = prediction for discrete and nominal values

What is Prediction?

- The goal of prediction is to forecast or deduce the value of an attribute based on values of other attributes.
  - A model is first created based on the data distribution.
  - The model is then used to predict future or unknown values

- **In Machine Learning**
  - If forecasting discrete value → **Classification**
  - If forecasting continuous value → **Prediction**

Classification Example

- Example training database
  - Two predictor attributes: Age and Car-type (**S**port, **M**inivan and **T**ruck)
  - Age is numeric, Car-type is categorical attribute
  - Class label indicates whether person bought product
  - Dependent attribute is *categorical*

Age	Car	Class
20	M	Yes
30	M	Yes
25	T	No
30	S	Yes
40	S	Yes
20	T	No
30	M	Yes
25	M	Yes
40	M	Yes
20	S	No

Regression (Prediction) Example

- Example training database
  - Two predictor attributes: Age and Car-type (**S**port, **M**inivan and **T**ruck)
  - Spent indicates how much person spent during a recent visit to the web site
  - Dependent attribute is *numerical*

Age	Car	Spent
20	M	$200
30	M	$150
25	T	$300
30	S	$220
40	S	$400
20	T	$80
30	M	$100
25	M	$125
40	M	$500
20	S	$420

Supervised and Unsupervised

- Supervised Classification = Classification
  - We know the class labels and the number of classes
- Unsupervised Classification = Clustering
  - We do not know the class labels and may not know the number of classes

Preparing Data Before Classification

- **Data transformation**:
  - Discretization of continuous data
  - Normalization to [-1..1] or [0..1]
- **Data Cleaning**:
  - Smoothing to reduce noise
- **Relevance Analysis**:
  - Feature selection to eliminate irrelevant attributes

Application

- Credit approval
- Target marketing
- Medical diagnosis
- Defective parts identification in manufacturing
- Crime zoning
- Treatment effectiveness analysis

Classification is a 3-step process

- **1. Model construction (Learning):**
  - Each tuple is assumed to belong to a predefined class, as determined by one of the attributes, called the **class label**.
  - The set of all tuples used for construction of the model is called **training set**.
  - The model is represented in the following forms:
    - Classification rules, (IF-THEN statements),
    - Decision tree
    - Mathematical formulae

1. Classification Process (Learning)

Name	Income	Age	Credit rating
Samir	Low	<30	bad
Ahmed	Medium	[30...40]	good
Salah	High	<30	good
Ali	Medium	>40	good
Sami	Low	[30..40]	good
Emad	Medium	<30	bad

Training Data

class

Classification Method

Classification Model

IF Income = 'High'
OR Age > 30
THEN Class = 'Good

OR

Decision Tree

OR

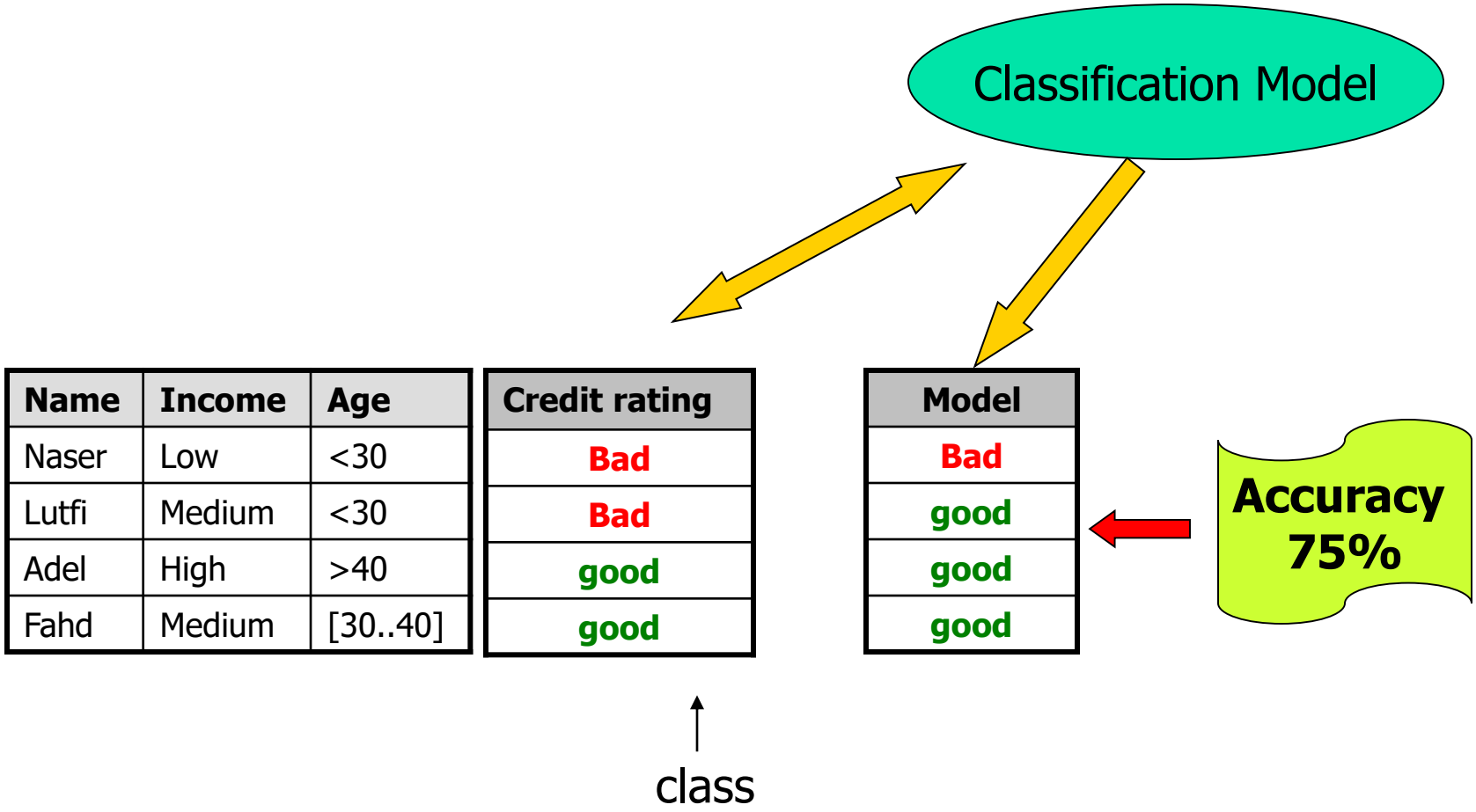
Mathematical For

Classification is a 3-step process

2. Model Evaluation (Accuracy):

- Estimate accuracy rate of the model based on a **test set**.
- The known label of test sample is compared with the classified result from the model.
- Accuracy rate is the percentage of test set samples that are correctly classified by the model.
- Test set is independent of training set otherwise over-fitting will occur

2. Classification Process (Accuracy Evaluation)

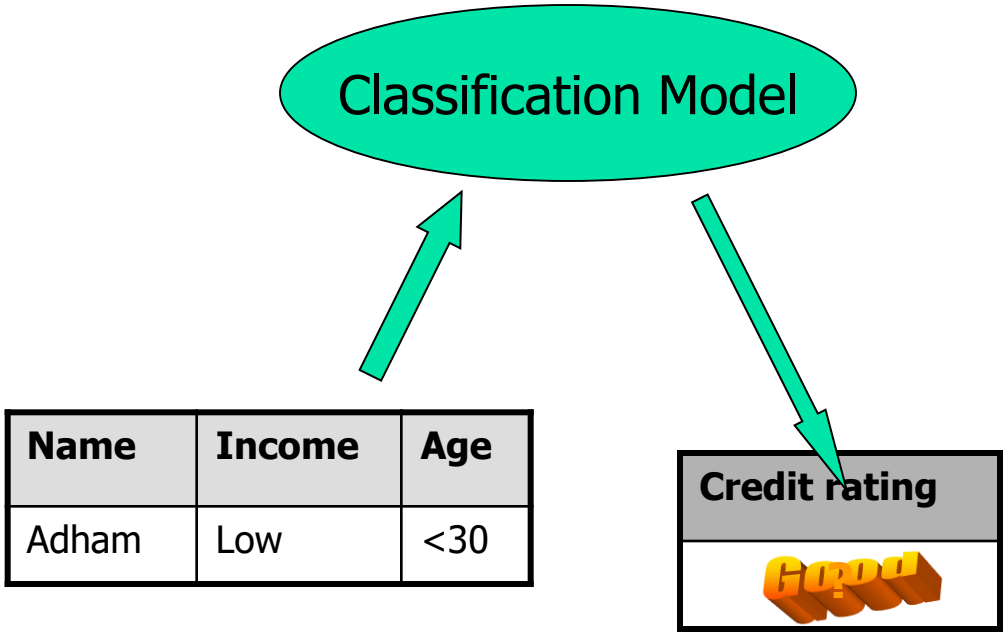


Classification is a three-step process

3. Model Use (Classification):

- The model is used to classify unseen objects.
  - Give a class label to a new tuple
  - Predict the value of an actual attribute <prediction>

3. Classification Process (Use)



Classification Methods

- Decision Tree Induction
- Neural Networks
- Bayesian Classification
- Association-Based Classification
- K-Nearest Neighbour
- Case-Based Reasoning
- Genetic Algorithms
- Rough Set Theory
- Fuzzy Sets
- Etc.

Comparing Classification and Prediction Methods

- Accuracy- *This is the ability of the model to correctly predict the class level of new or previously unseen data*.
  - classifier accuracy: predicting class label of new or previously unseen data.
  - predictor accuracy: guessing value of predicted attributes new or previously unseen data.
- Speed (computational cost)
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)

Comparing Classification and Prediction Methods

- Robustness: handling noise and missing values (ability of model to make correct predictions)
- Scalability: the ability to construct the model efficiently given large amounts of data.
- Interpretability:
  - This refers Level of understanding and insight provided by the model (classifier or predictor).
- Other measures, e.g., goodness of rules, such as decision tree size.

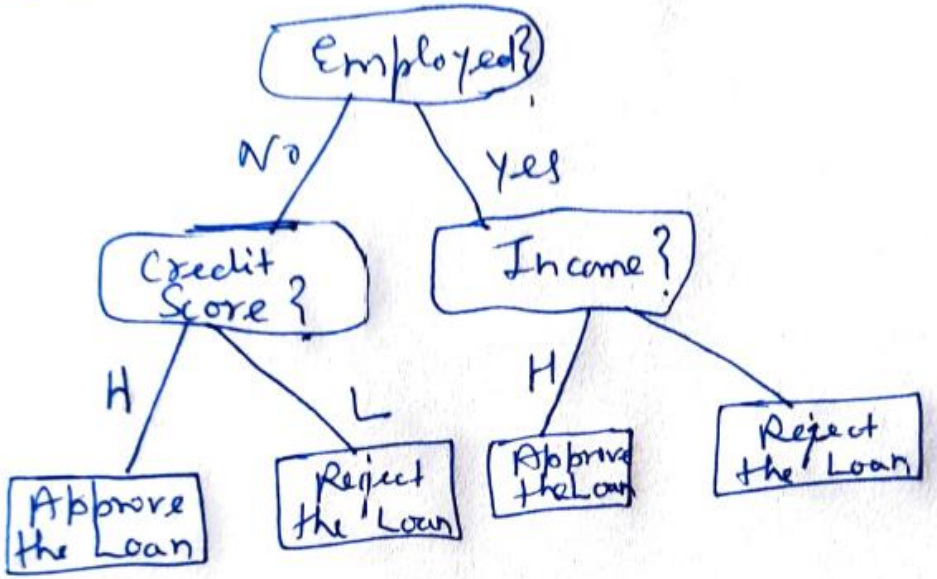
Decision Tree

Decision Tree

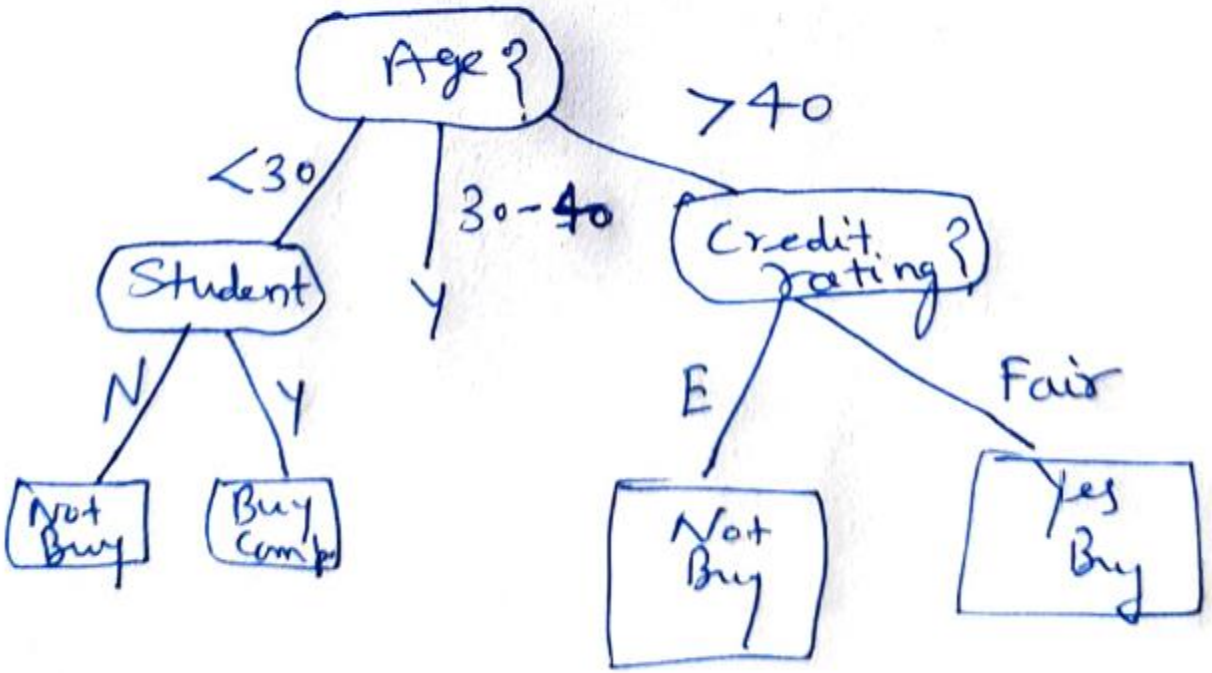
We have:

1) Decision Nodes / Test Nodes
2) Leaf Nodes
3) Edges outcome of test

Example: ① Approve the Loan OR Not Approve the Loan



② Person is likely to buy a computer ?



Example 3

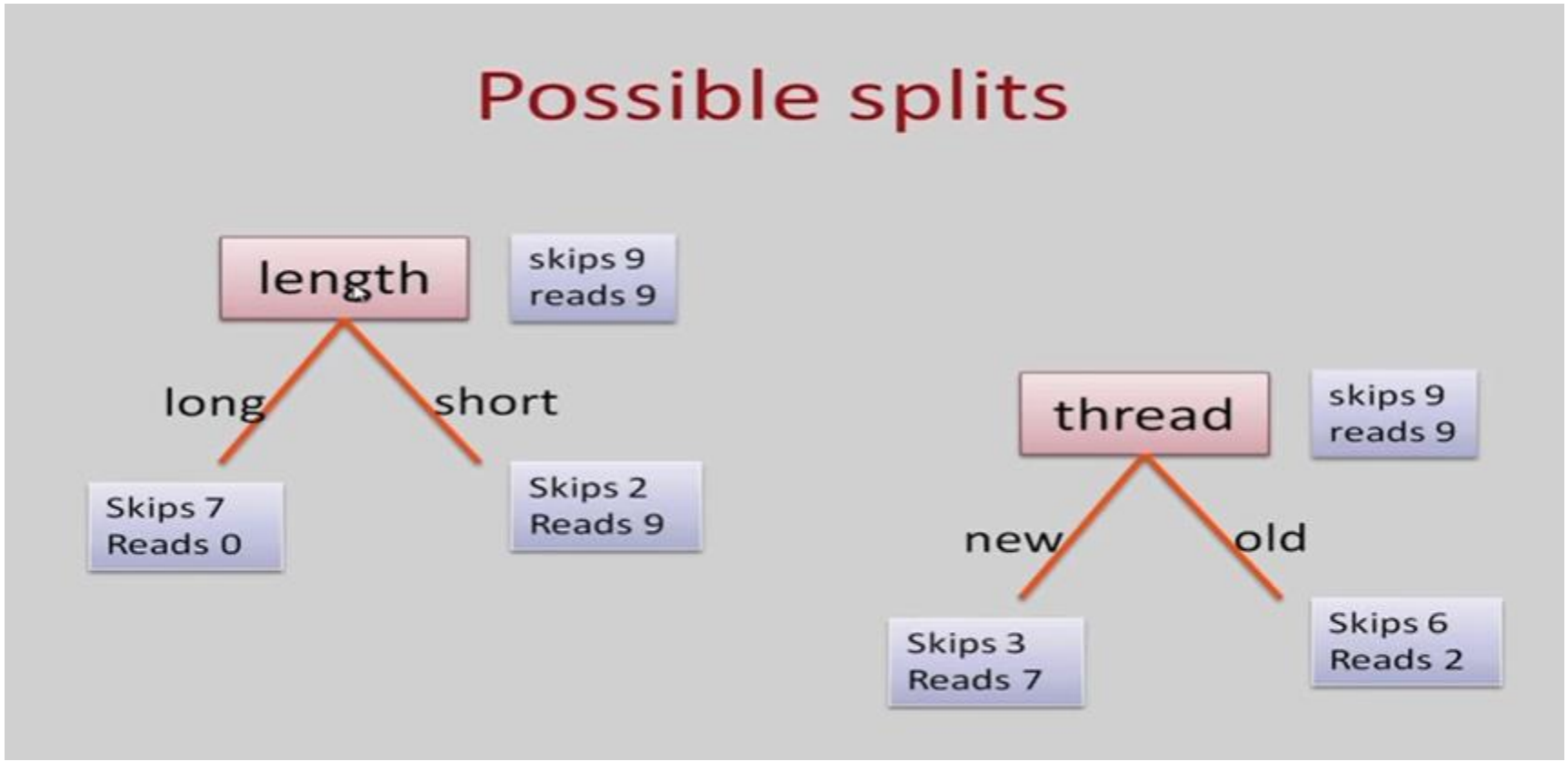
Example Data

Training Examples:

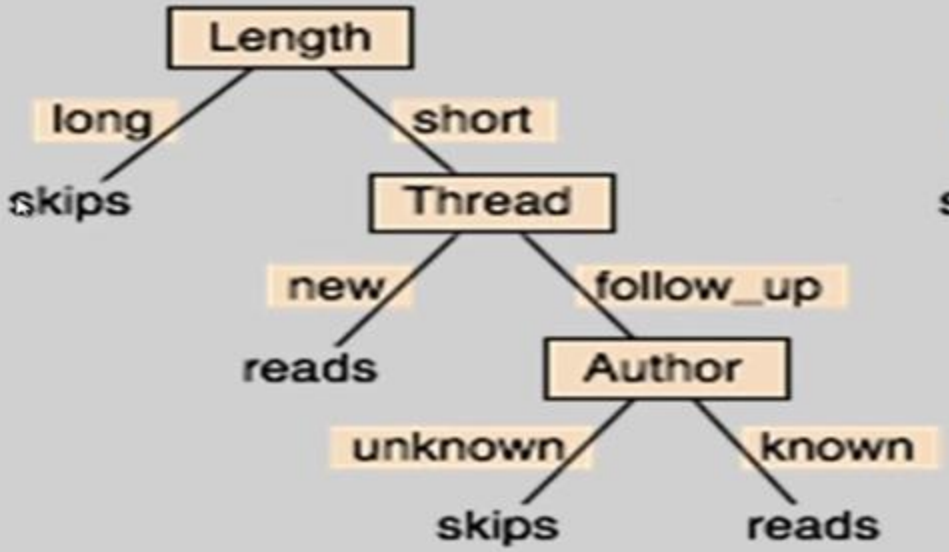
	Action	Author	Thread	Length	Where
e1	skips	known	new	long	Home
e2	reads	unknown	new	short	Work
e3	skips	unknown	old	long	Work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

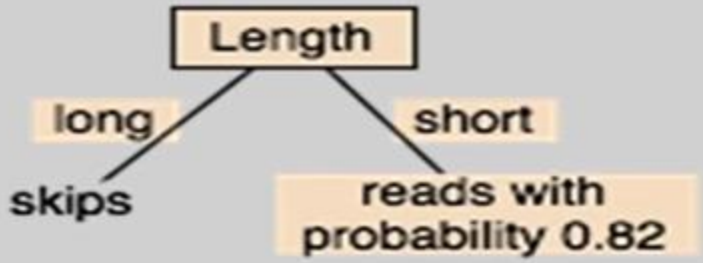
New Examples:

e7	???	known	new	short	work
e8	???	unknown	new	short	work


Possible splits
length
skips 9
reads 9
long
short
Skips 7
Reads 0
Skips 2
Reads 9
thread
skips 9
reads 9
new
old
Skips 3
Reads 7
Skips 6
Reads 2

Two Example DTs



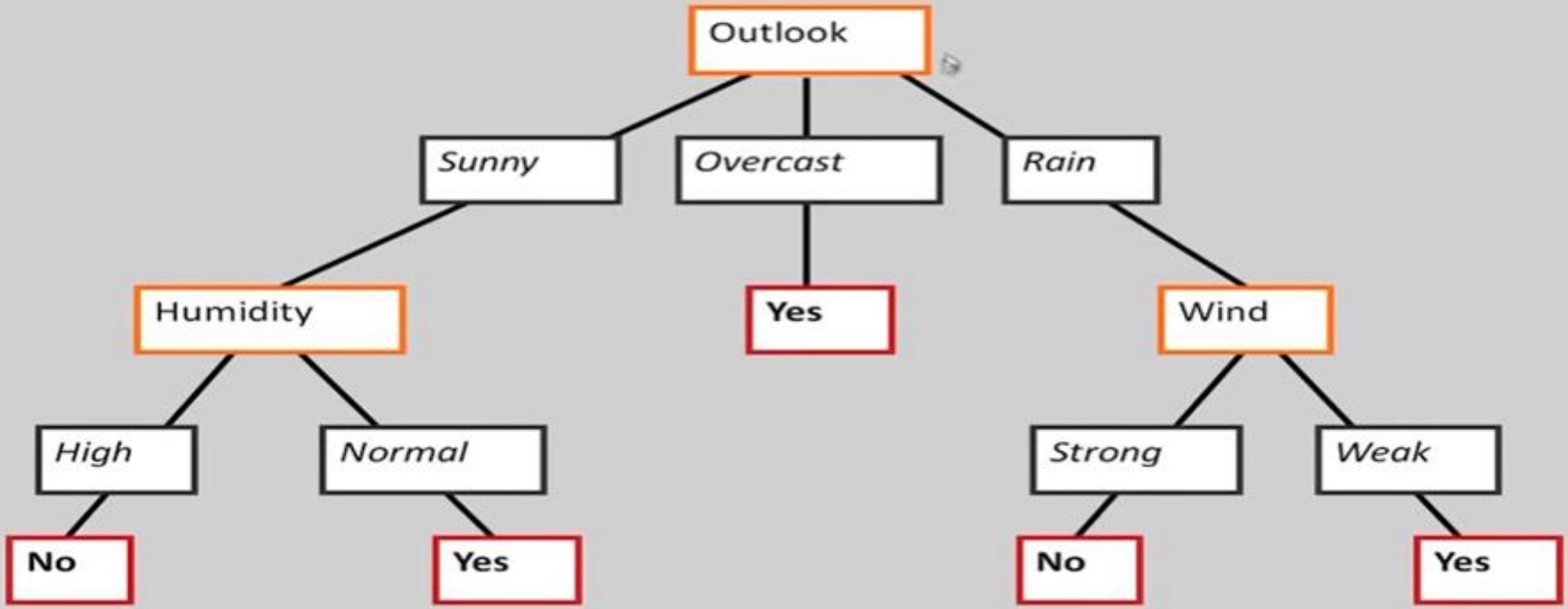


Example 4

Decision Tree for PlayTennis

- Attributes and their values:
  - Outlook: *Sunny, Overcast, Rain*
  - Humidity: *High, Normal*
  - Wind: *Strong, Weak*
  - Temperature: *Hot, Mild, Cool*
- Target concept - Play Tennis: *Yes, No*

Decision Tree for PlayTennis

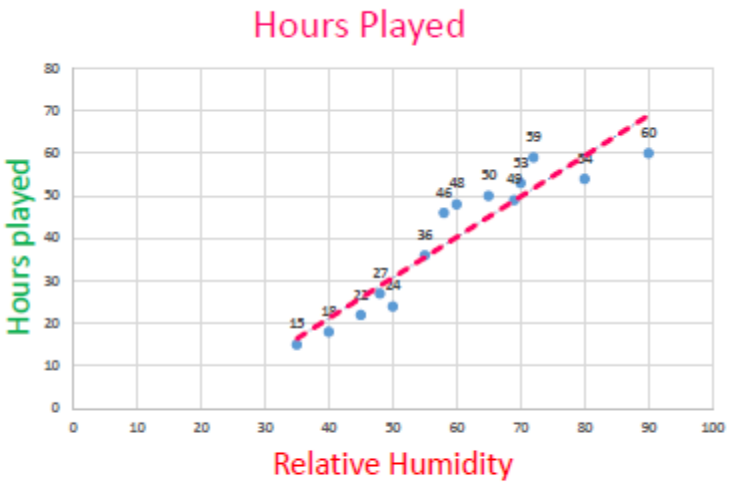


What is a Decision Tree?

- A decision tree is a flow-chart-like tree structure.
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf node represents class label

A quick recap of Linear Regression –Linear models

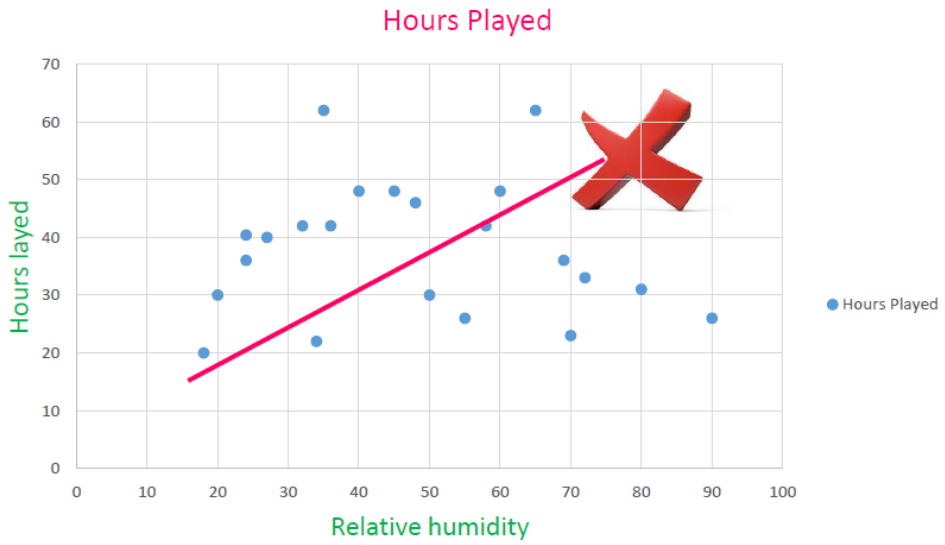
Relative Humidity	Hours Played
55	36
50	24
60	48
58	46
65	50
70	53
48	27
69	49
72	59
45	22
40	18
35	15
80	54
90	60



Linear Regression can handle this type of data quite well

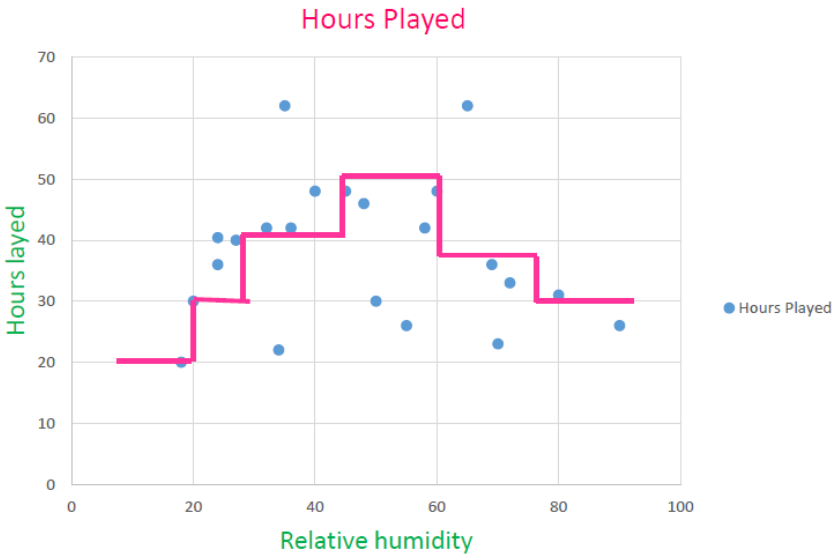
Can Linear Regression help us in this scenario?

Relative Humidity	Hours Played
55	26
50	30
60	48
58	42
65	62
70	23
48	46
69	36
72	33
45	48
40	48
35	62
80	31
90	26
27	40
36	42
24	40.4
32	42
24	36
34	22
18	20
20	30



Data are relatively scattered

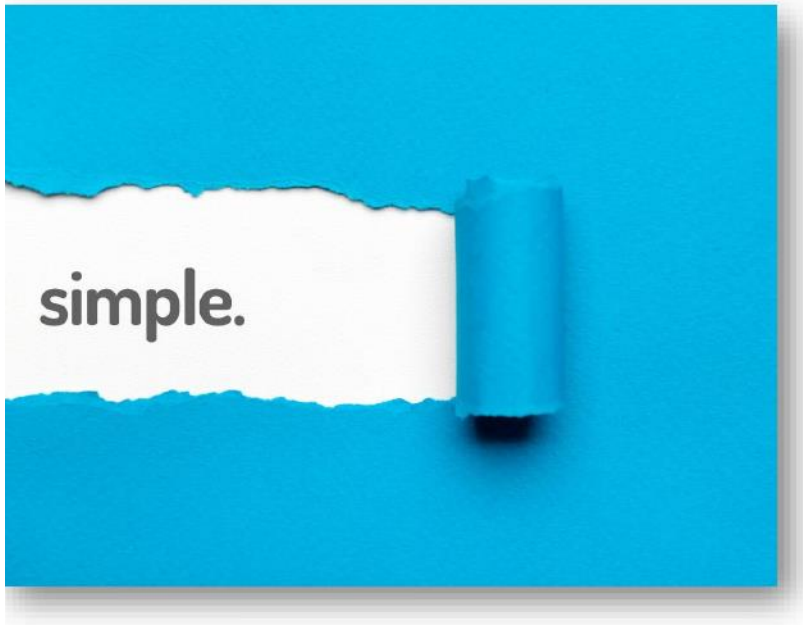
How does Decision Tree come to the rescue?

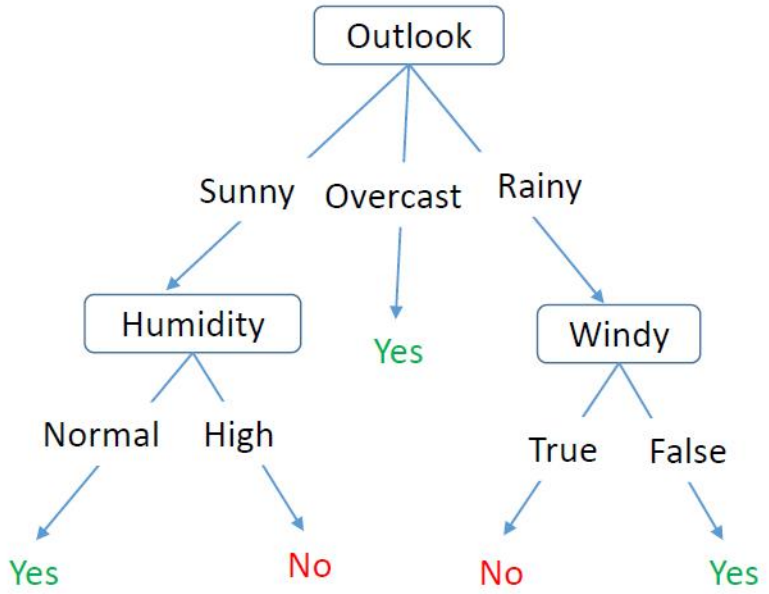


- Non linear data can well be handled by Decision Tree.
- It doesn't affect the performance of Decision Tree

Simplicity

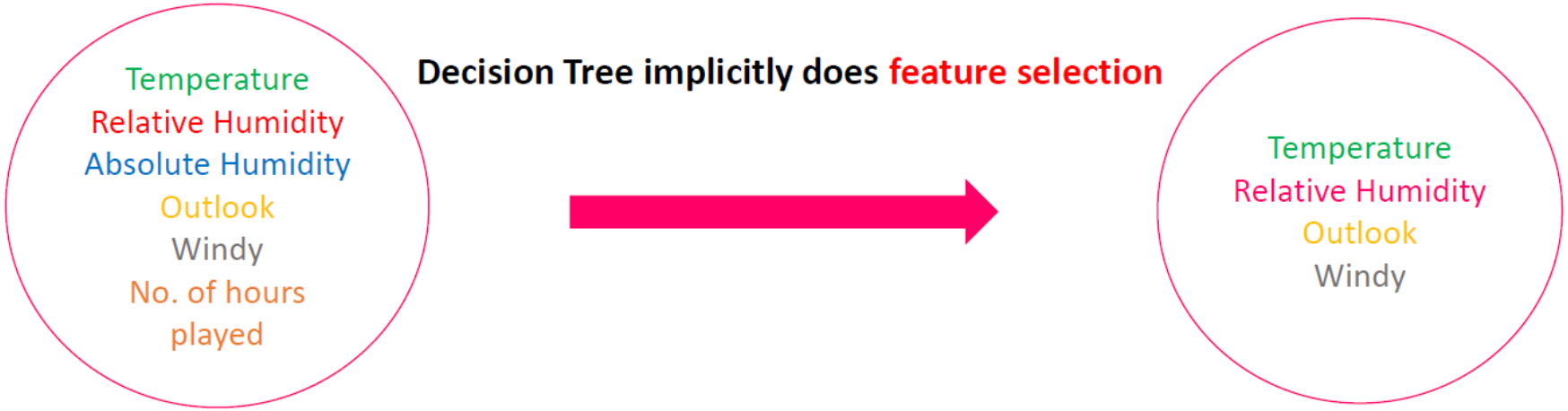
It’s simple to understand , interpret & visualize





Feature selection

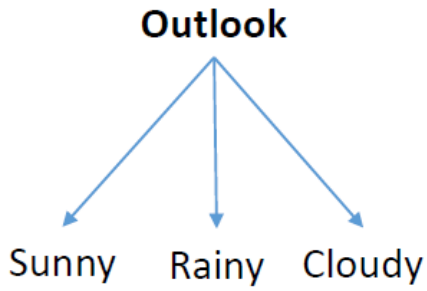
Decision Tree identifies and removes unnecessary, irrelevant & redundant attributes from data that do not contribute to accuracy of a predictive model



Handling different types of data

- Can handle both **categorical & numerical** data.
- So can be used both for **Regression & Classification**

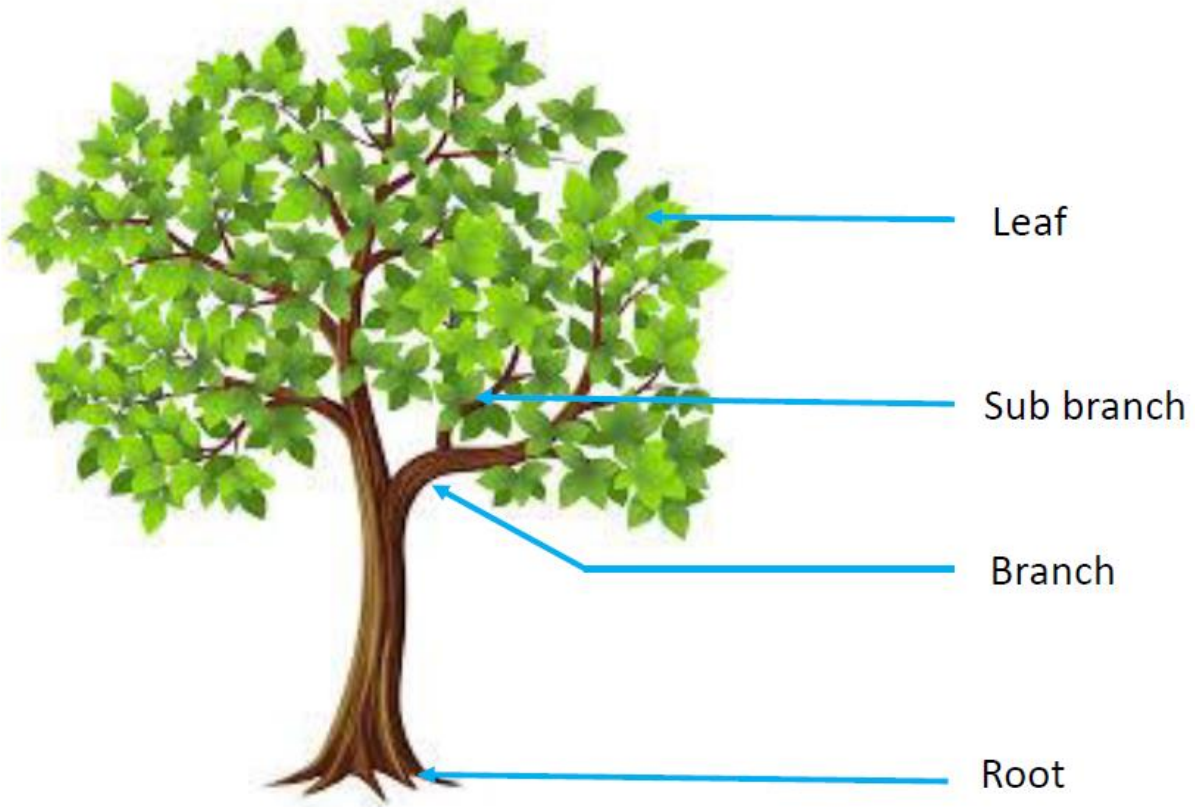
Categorical



Numeric

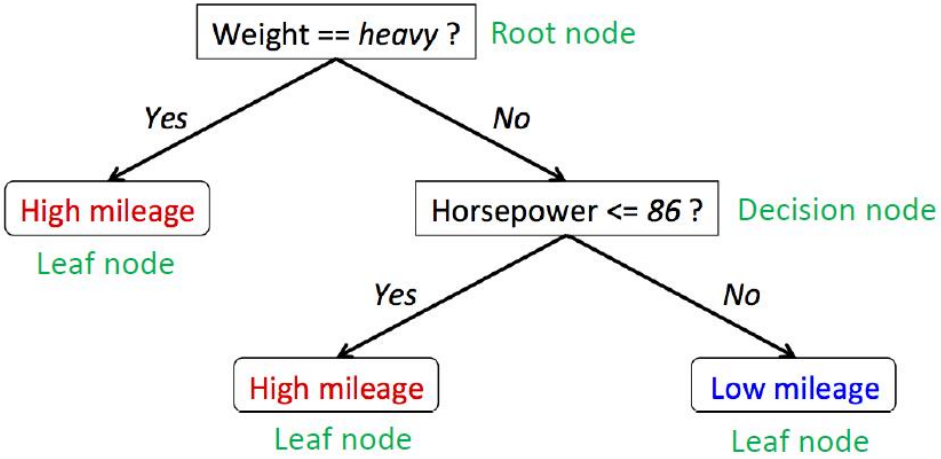
Hours Played
26
30
48
46
62
23
43
36

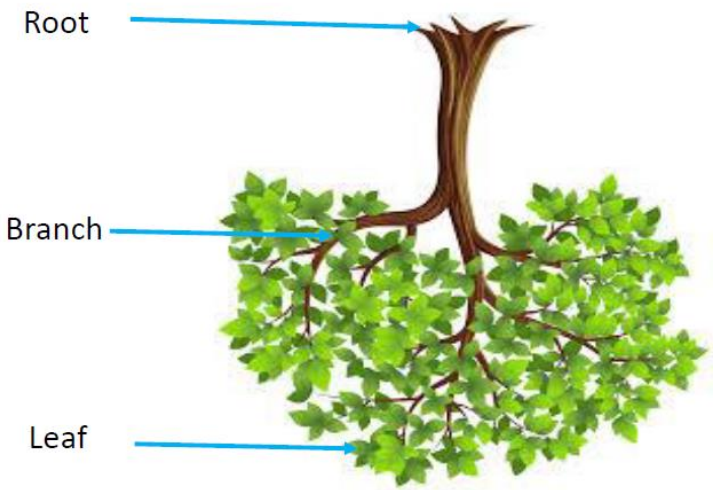
What is tree ?



What is Decision Tree(DT) ?

- A form of **supervised** learning used in ***classification*** & ***regression***,
- In which predicted values are obtained through a *inverted* **tree like structure**.





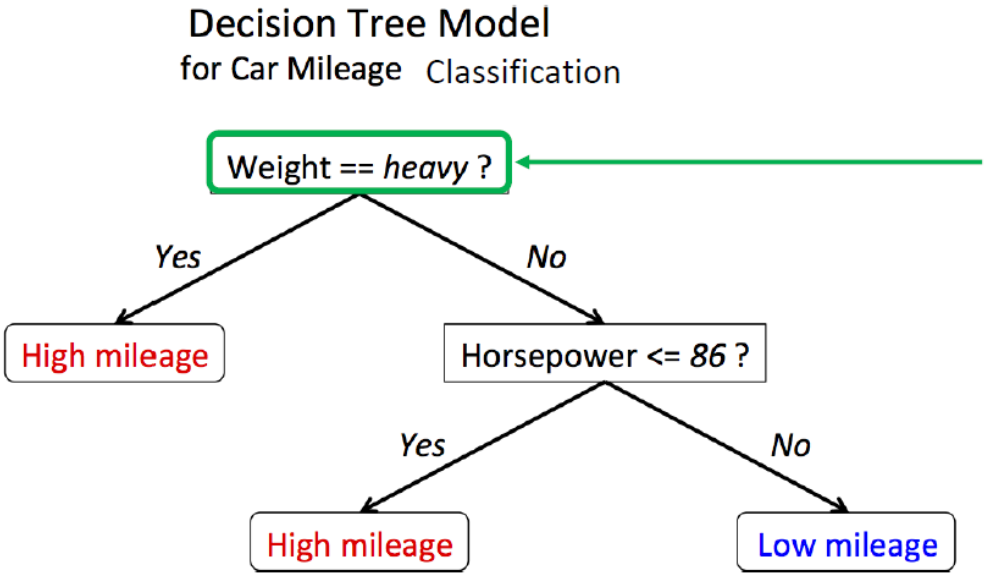
Use cases of Decision Tree

- Decision Tree is often called CART(*Classification & Regression Tree*).
- It is used in both *classification & regression* for predictive modelling.

Will London remain sunny tomorrow?

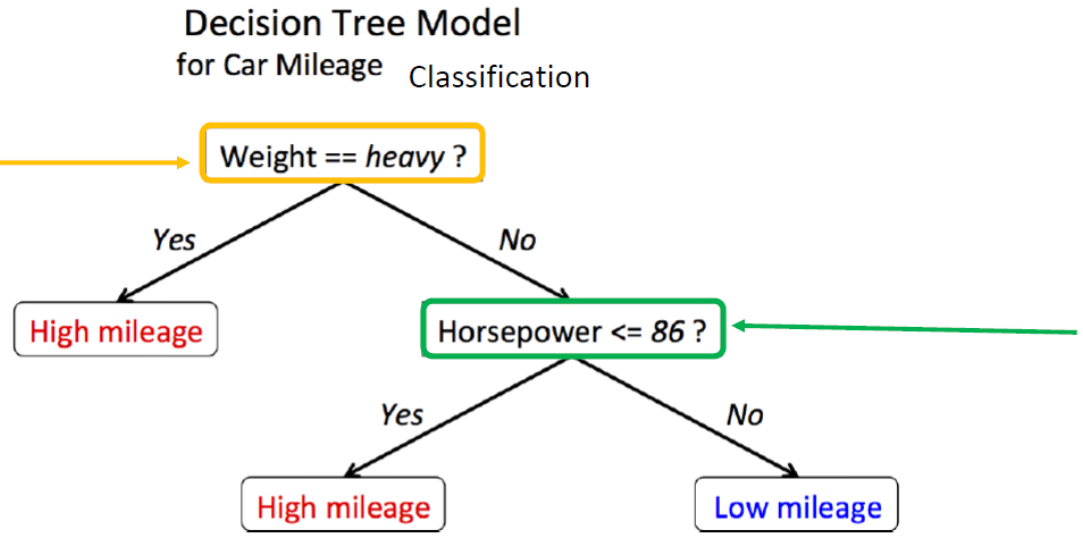
Predict Snowfall(in inch) in Berlin in January.

What is Root node of Decision Tree?



- Root node is first node of building a DT,
- In which only a single attribute is split.

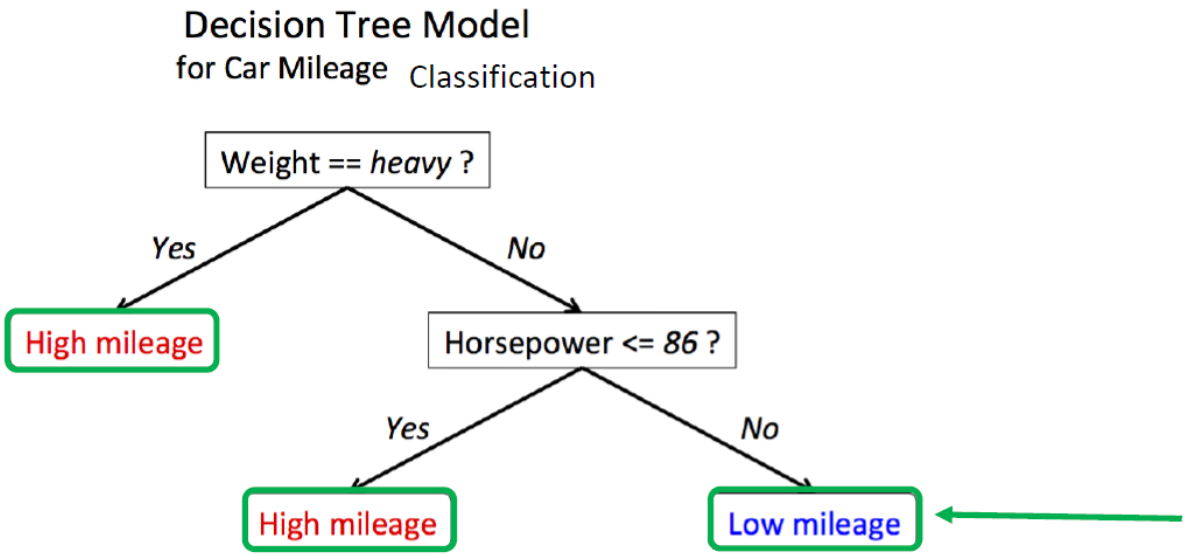
What is decision node of decision tree ?



Decision node is that node of DT that is split into decisions.

N.B: Root node can be decision node if it's split into decisions .

What are leaf nodes of Decision Tree?



- Also called **terminal nodes**.
- Nodes have **predicted decisions** about *target* variable.

CART(Classification and Regression Tree)

- **Clear skies**, bright and shining sun are ideal conditions for playing golf.
- Rain, harsh winds, and a low temperature can all have negative effect on playing golf.

CART(Classification and Regression Tree)

Outlook	Temperature	Humidity	Windy	Hours Played	Plat Golf
Rainy	Hot	High	FALSE	26	No
Rainy	Hot	High	TRUE	30	No
Overcast	Hot	High	FALSE	48	Yes
Sunny	Mild	High	FALSE	46	Yes
Sunny	Cool	Normal	FALSE	62	Yes
Sunny	Cool	Normal	TRUE	23	No
Overcast	Cool	Normal	TRUE	43	Yes
Rainy	Mild	High	FALSE	36	No
Rainy	Cool	Normal	FALSE	38	Yes
Sunny	Mild	Normal	FALSE	48	Yes
Rainy	Mild	Normal	TRUE	48	Yes
Overcast	Mild	High	TRUE	62	Yes
Overcast	Hot	Normal	FALSE	44	Yes
Sunny	Mild	High	TRUE	30	No

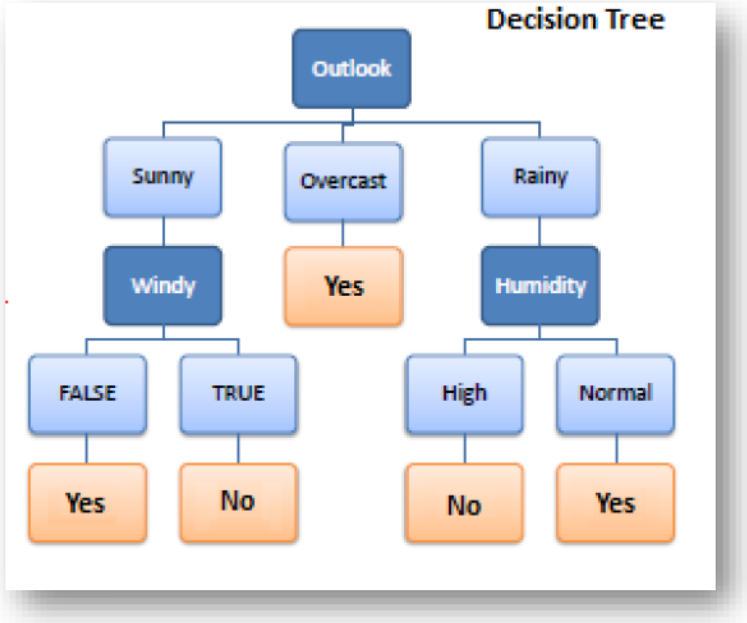
- From dataset we can do predictive analysis regarding whether Golf will be **played** or **not**(Classification problem).
- How many hours (Regression) will golf be played given weather conditions using Decision Tree algorithm.

CART: Classification Tree

Predictors

Target (**Categorical**)

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

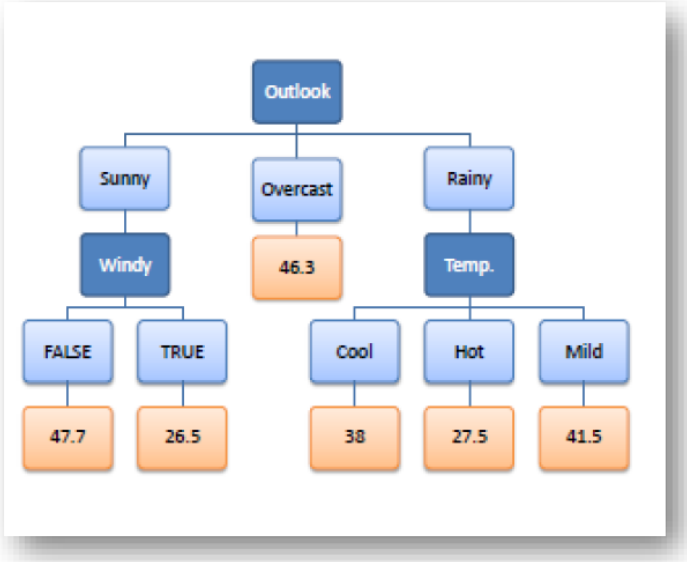


CART: Regression Tree

Predictors

Target (**Continuous**)

Outlook	Temperature	Humidity	Windy	Hours Played
Rainy	Hot	High	FALSE	26
Rainy	Hot	High	TRUE	30
Overcast	Hot	High	FALSE	48
Sunny	Mild	High	FALSE	46
Sunny	Cool	Normal	FALSE	62
Sunny	Cool	Normal	TRUE	23
Overcast	Cool	Normal	TRUE	43
Rainy	Mild	High	FALSE	36
Rainy	Cool	Normal	FALSE	38
Sunny	Mild	Normal	FALSE	48
Rainy	Mild	Normal	TRUE	48
Overcast	Mild	High	TRUE	62
Overcast	Hot	Normal	FALSE	44
Sunny	Mild	High	TRUE	30



How to build Decision Tree?

Let us build a Decision Tree using a very common algorithm namely:

- **ID3(Iterative Dichotomiser 3)**– uses entropy & information *gain* as metrics

ID3(Iterative Dichotomiser 3)

ID3 algorithm was invented by **Ross Quinlan**.

- The basic concept is to build a decision tree by employing **top-down , greedy search** through given data set to **test each attribute(variable) at each tree node.**

Which attribute should we start with?
And
How would we proceed once we get started?

What is Entropy?

Entropy: It is used to measure disorder in the system. If in a particular node, all examples are positive OR all examples are negative (i.e. all examples belong to the same class), then it is homogeneous set of examples and entropy is low.

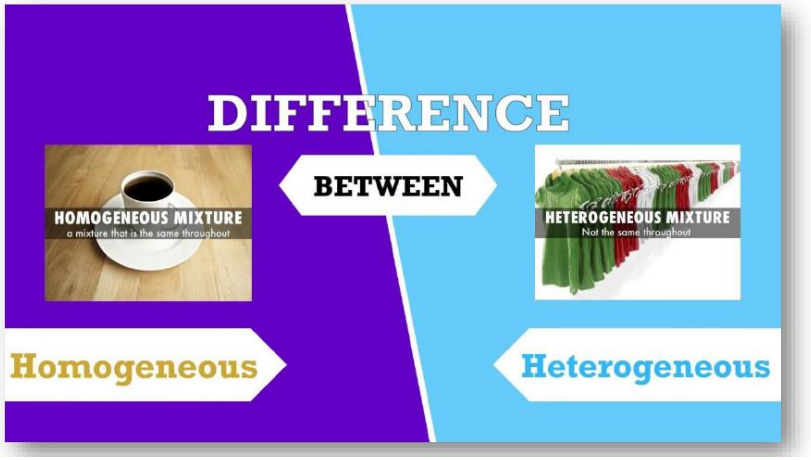
However if we have two classes and half of the examples belong to one class and half belong to another class, then entropy is high

$$Entropy(S) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- **Entropy(S)** is measure of **homogeneity** in data.

Important:

Entropy(S) = - p $\log_2$ p – q $\log_2$ q



Entropy of heterogeneous data

Homogeneous data

- Entropy is **0**.

$$Entropy(S) = - p \log_2 p - q \log_2 q$$

p=1, q= 0(it will be only heads)

$Entropy(S)= - 1 \log_2 (1) - 0 \log_2 0= 0$

Heterogeneous data

- Entropy is **1**.

$$Entropy(S) = - p \log_2 p - q \log_2 q$$

p=0.5, q= 0.5(it must be either heads or tails)

$Entropy(S)= - 0.5 \log_2 (0.5) - 0.5 \log_2(0.5)=1$

Information Gain(IG)

- Information Gain also called Kullback-Leibler divergence(KL divergence) indicates **relative change** in Entropy.
- IG is derived by **subtracting Entropy of a particular attribute (A) from Entropy of total samples(S)**.

Information Gain(S,A)= Entropy(S) – Entropy(S,A)

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

Entropy(S)= $-\left(\frac{9}{14}\right) log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right) log_2\left(\frac{5}{14}\right) = 0.94$

Entropy(S ,Outlook)= $\frac{5}{14} * 0.971 + \frac{5}{14} * 0.971 + \frac{4}{14} * 0 = 0.693$

Don't worry we'll break up the equation in subsequent slides.

Information Gain = 0.94-0.693

= 0.247

$$Entropy(S, A) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

Calculate Entropy & Information Gain to build a Decision Tree

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

To build a Decision Tree:

- Step 1: Calculate Entropy of set of samples,
- Step 2: Calculate Entropy of each attribute,
- Step 3: Finally calculate information gain.

N.B: Attribute with highest information gain will be root node of Decision Tree

Step 1: Let's calculate Entropy for entire sample

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

The initial step is to calculate Entropy of total set(S).

There are 5 **No** & 9 **Yes** of playing golf

Yes	No	Total
9	5	14
p	q	

Entropy(S) = - p $\log_2$ p – q $\log_2$ q

$$\text{Entropy(S)} = -\left(\frac{9}{14}\right) log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right) log_2\left(\frac{5}{14}\right)$$

$$= \mathbf{0.94}$$

Step 2: Calculate Entropy for each column

$$Entropy(S,A)=\sum_{j=1}^{v}\frac{|D_j|}{|D|}\times I(D_j)$$

For demonstration we'll calculate for **Outlook** column

Outlook	Play Golf
Rainy	No
Rainy	No
Overcast	Yes
Sunny	Yes
Sunny	Yes
Sunny	No
Overcast	Yes
Rainy	No
Rainy	Yes
Sunny	Yes
Rainy	Yes
Overcast	Yes

Outlook={ Sunny, Rainy ,Overcast }

$Entropy(S_{Sunny})=-(\frac{3}{5})\log_2(\frac{3}{5})-(\frac{2}{5})\log_2(\frac{2}{5})=0.971$

$Entropy(S_{Rainy})=-(\frac{2}{5})\log_2(\frac{2}{5})-(\frac{3}{5})\log_2(\frac{3}{5})=0.971$

$Entropy(S_{Overcast})=-1\log_2 1-0\log_2 0=0$

$Entropy(S,Outlook)=\frac{5}{14}*0.971+\frac{5}{14}*0.971+\frac{4}{14}*0=0.693$

Step 3: Calculate Information Gain

Outlook	Play Golf
Rainy	No
Rainy	No
Overcast	Yes
Sunny	Yes
Sunny	Yes
Sunny	No
Overcast	Yes
Rainy	No
Rainy	Yes
Sunny	Yes
Rainy	Yes
Overcast	Yes

IG(S ,Outlook) = Entropy(s) - Entropy(S ,Outlook)

IG(S ,Outlook) = 0.94-0.693 = **0.247**

The same way can calculate information gain for other attributes.

Information Gain from all attributes

		Yes	No
Outlook	Sunny	3	2
	Rainy	2	3
	Overcast	4	0

Information Gain= **0.247**

		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1

Information Gain= 0.029

		Yes	No
Humidity	High	3	4
	Normal	6	1

Information Gain= 0.152

		Yes	No
windy	FALSE	6	2
	TRUE	3	3

Information Gain= 0.048

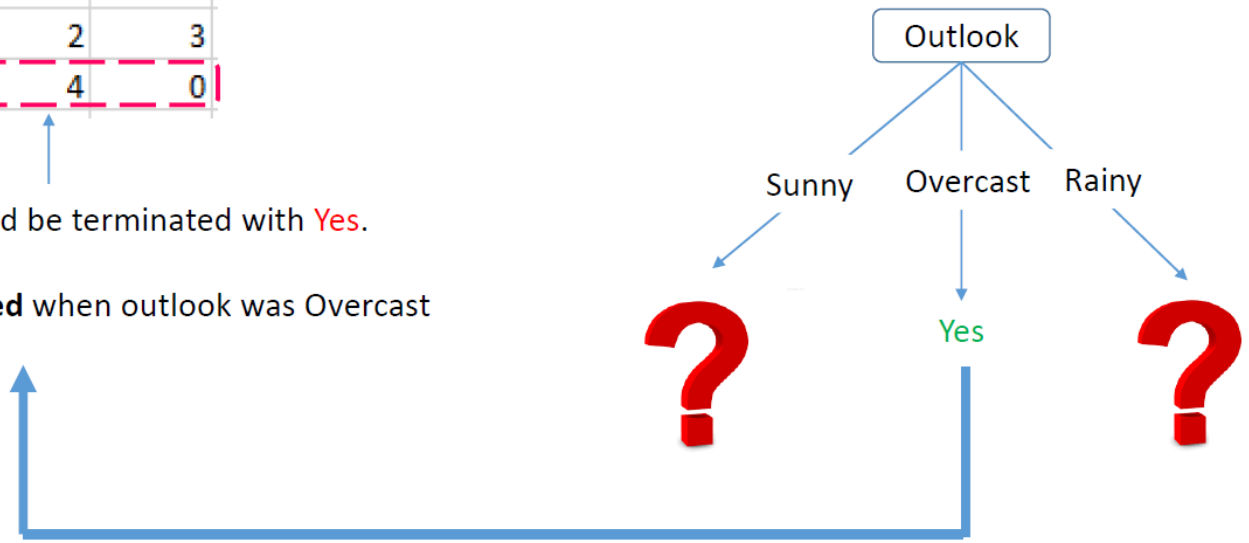
We'll use information gain to decide attribute node of tree

How does the tree look initially

As per information gain, Outlook will be root node because we've highest information gain from it.

		Yes	No
Outlook	Sunny	3	2
	Rainy	2	3
	Overcast	4	0

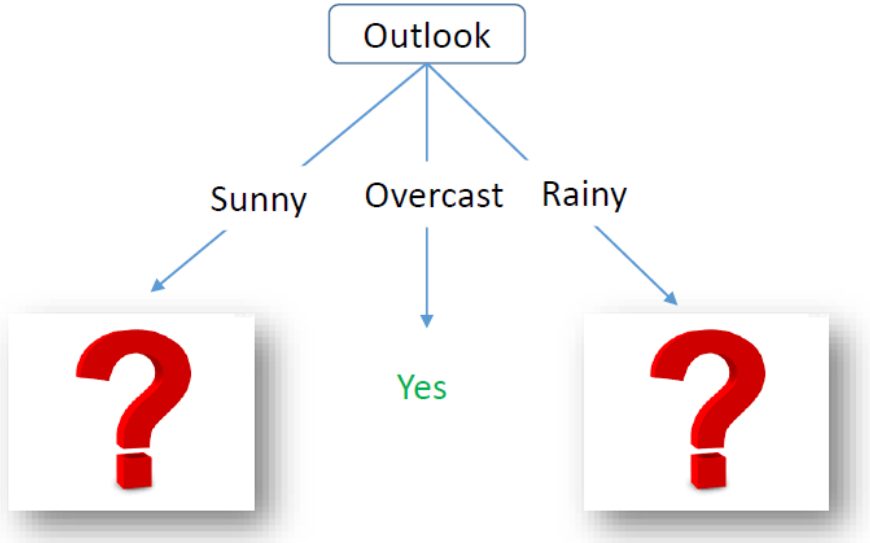
- **Overcast** outlook should be terminated with Yes.
- Because Golf was **played** when outlook was Overcast



Build Decision Tree –But what next?

- But question is which attribute should be tested at Sunny & Rainy branches.

Humidity, Temperature Or Windy?



Build Decision Tree –next is here

If we find Information Gain from Humidity, Temperature & Wind when outlook is Sunny.

Information Gain(S_{sunny}, Humidity) = 0.97- (3/5)*0.0 –(2/5)*0.0 = **0.97**

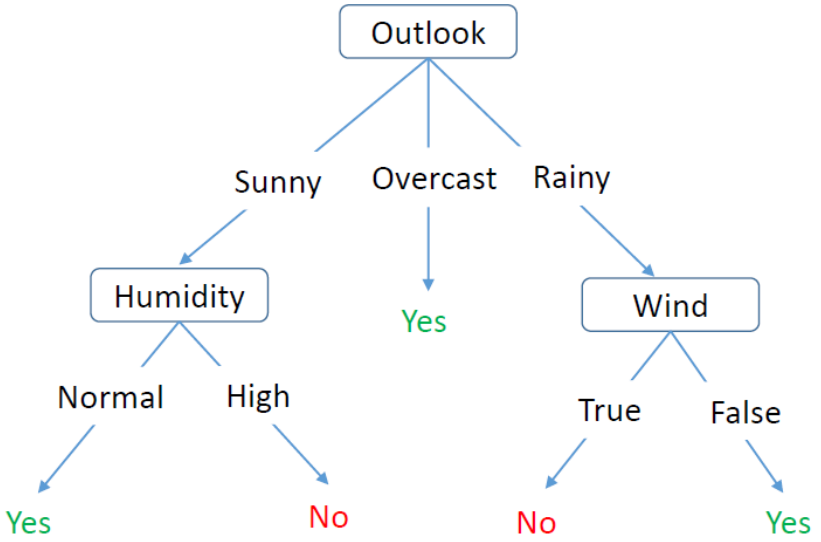
Information Gain(S_{sunny}, Temperature) = 0.97- (2/5)*0.0 - (2/5)*1.0 - (1/5)*0.0 = 0.57

Information Gain(S_{sunny}, Wind) = 0.97 - (2/5)*1.0 - (3/5)*0.918 = 0.019

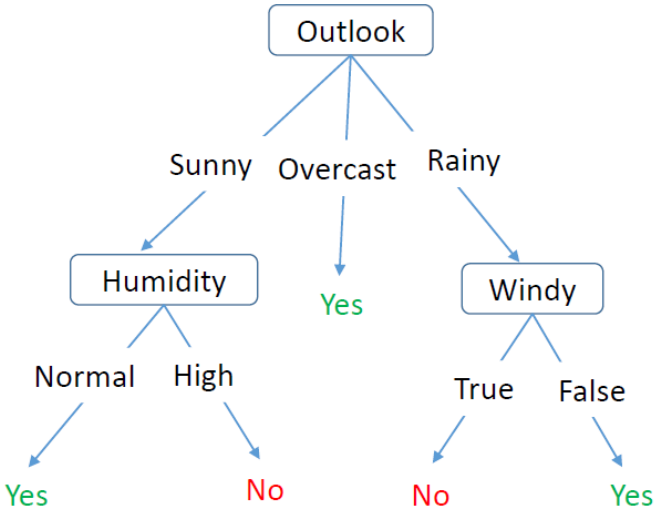
As per above information Gain, **Humidity** should be tested at Sunny branch.

How does the tree look finally?

- Proceeding with same way, we get Windy as immediate decision node for Rainy.
- The process of splitting continues until all data is classified .

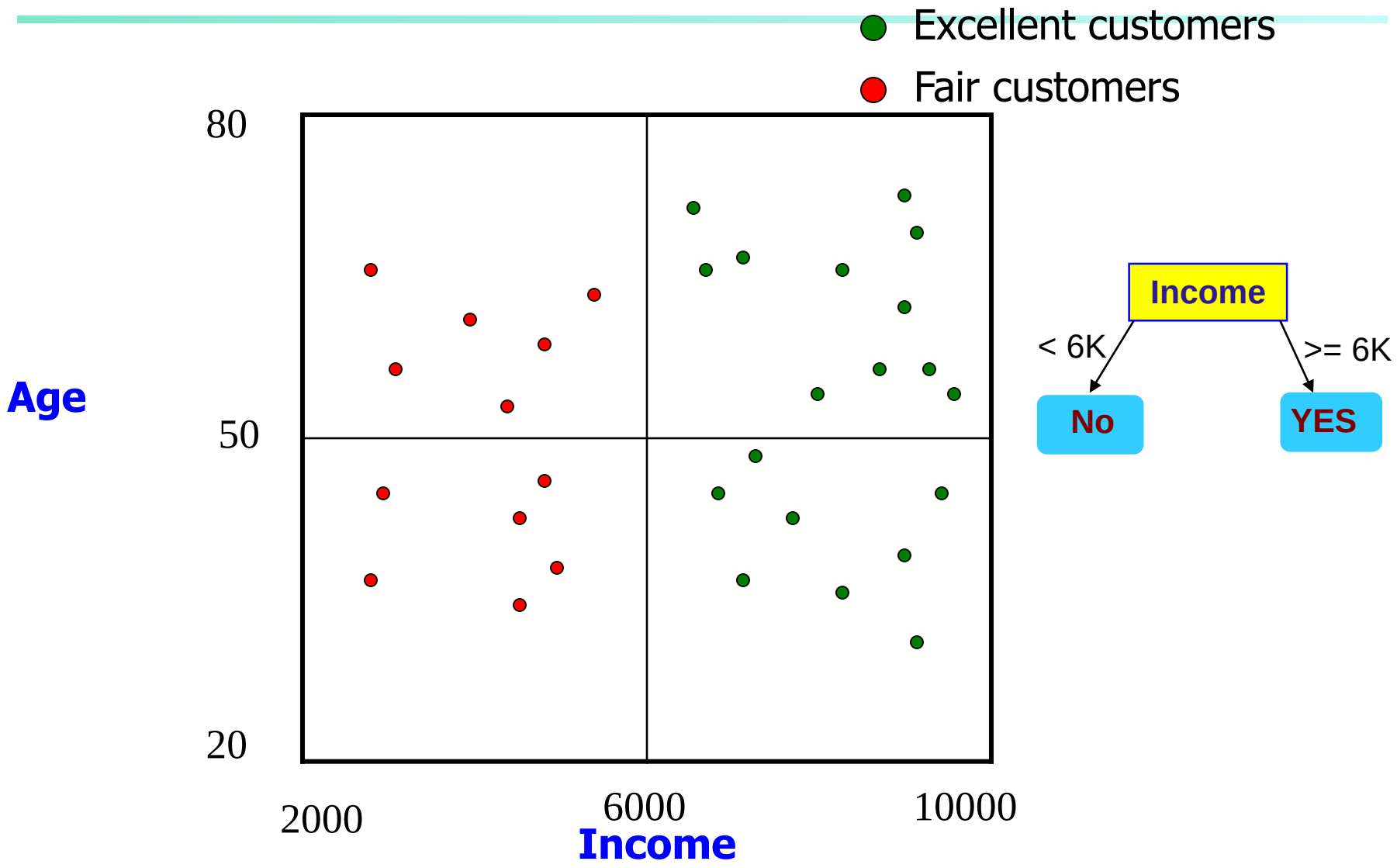


Decision rules

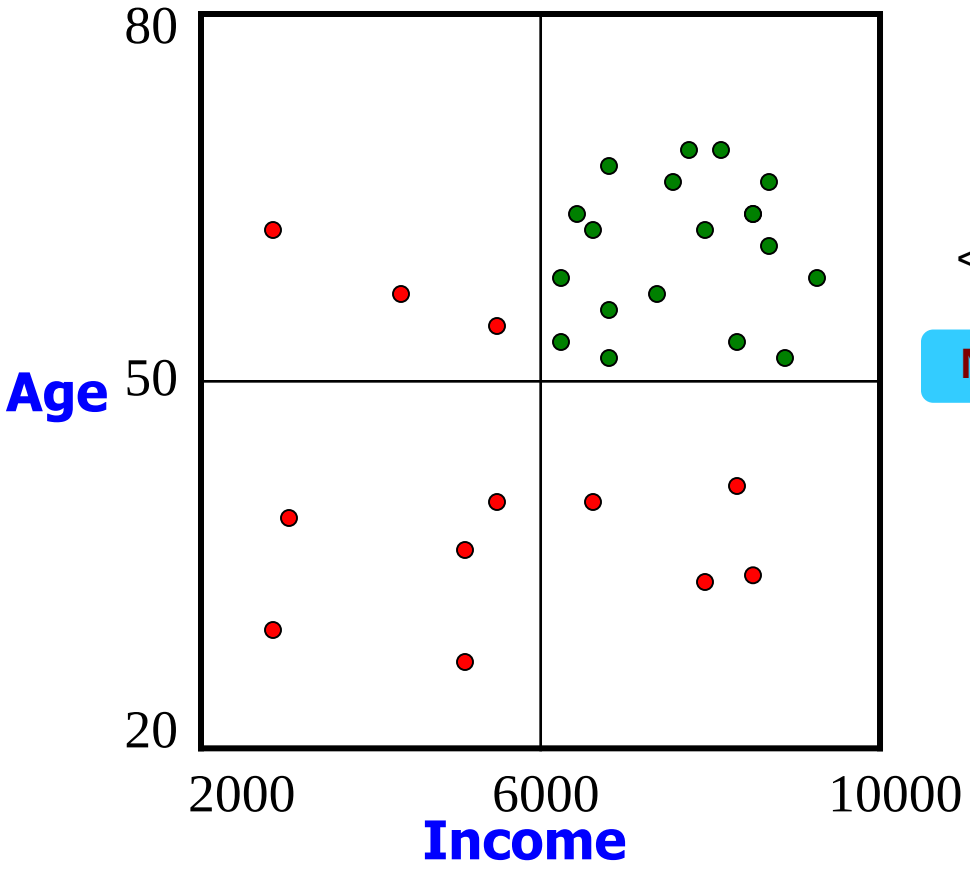


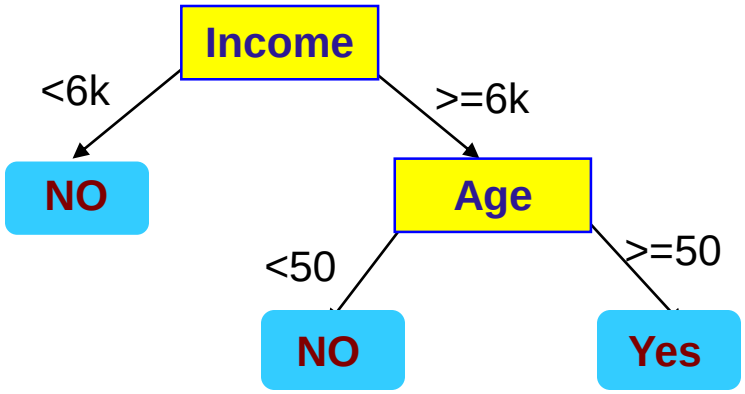
- IF outlook = sunny AND humidity = high THEN play golf = No
- IF outlook = sunny AND humidity = Normal THEN play golf =Yes
- IF outlook = rainy AND windy = True THEN play golf = No
- IF outlook = overcast THEN play golf = Yes
- IF outlook = rain AND wind = False THEN play golf = Yes

Sample Decision Tree



Sample Decision Tree





Decision-Tree Classification Methods

- The basic top-down decision tree generation approach usually consists of two phases:

1. **Tree construction**
   - At the start, all the training examples are at the root.
   - Partition examples are recursively based on selected attributes.
2. **Tree pruning**
   - Aiming at removing tree branches that may reflect noise in the training data and lead to errors when classifying test data → improve classification accuracy

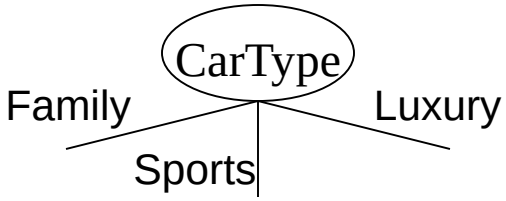
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

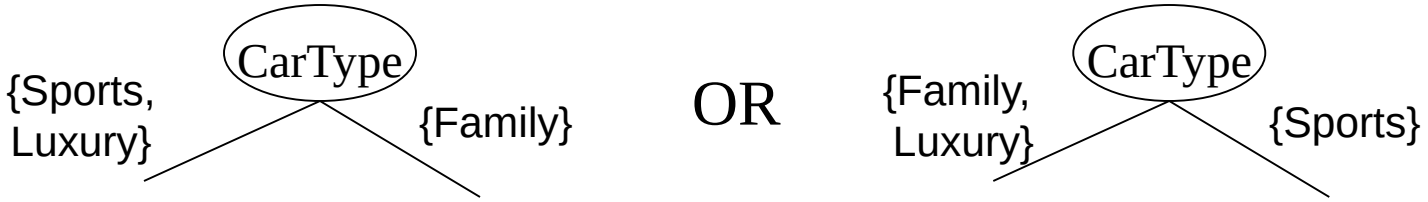
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

- Multi-way split: Use as many partitions as distinct values.

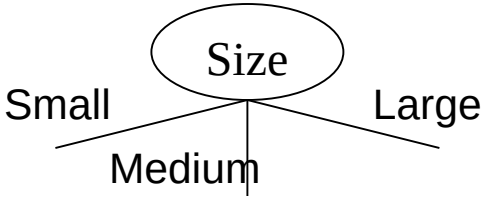


- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.

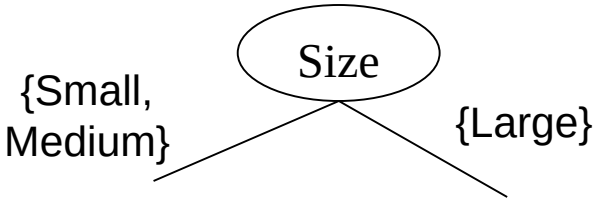


Splitting Based on Ordinal Attributes

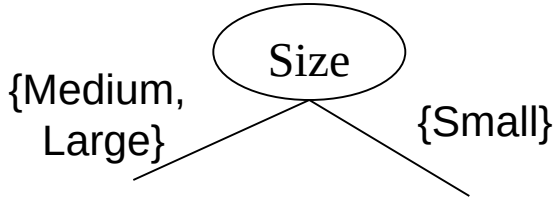
- Multi-way split: Use as many partitions as distinct values.

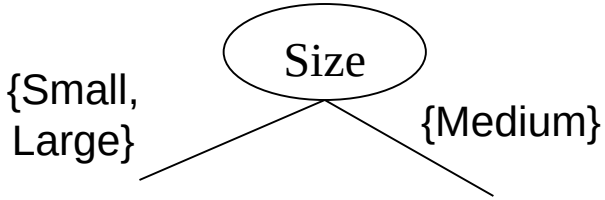


- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.



OR

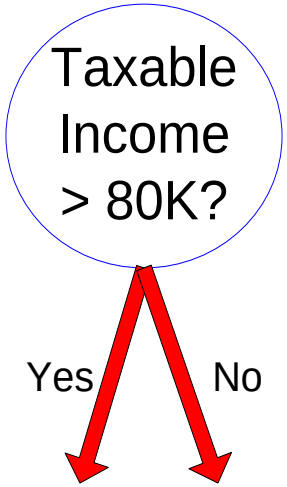




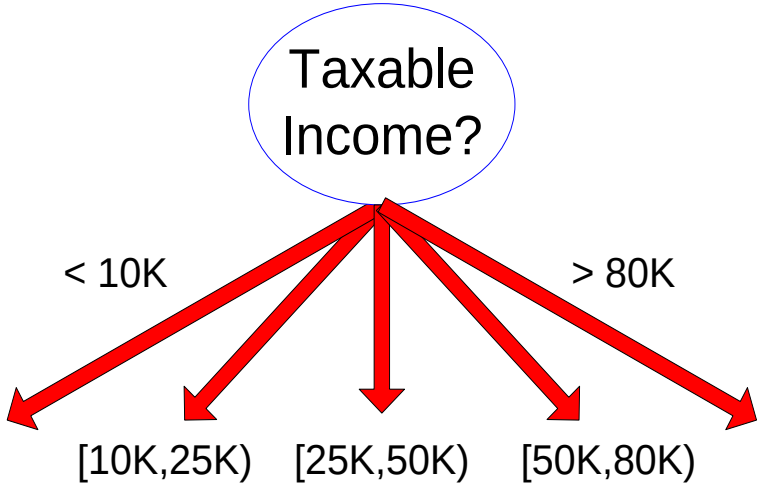
Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut

Splitting Based on Continuous Attributes



(i) Binary split



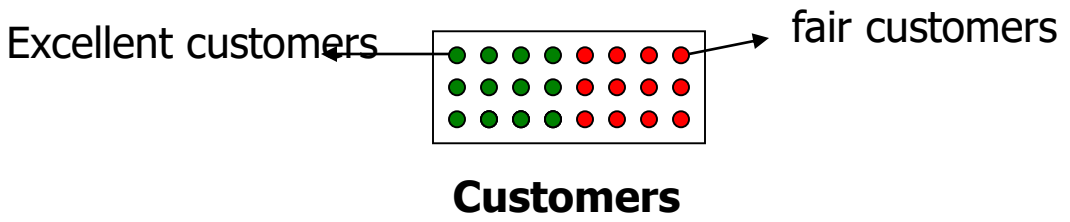
(ii) Multi-way split

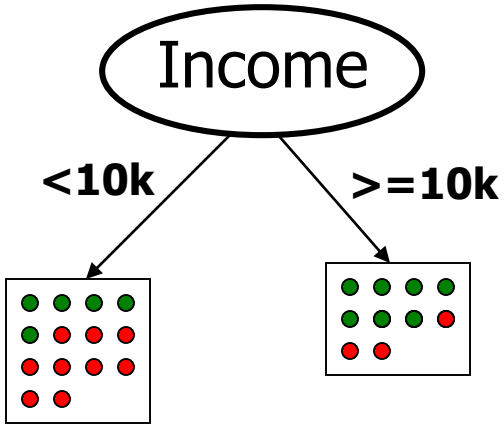
Tree Induction

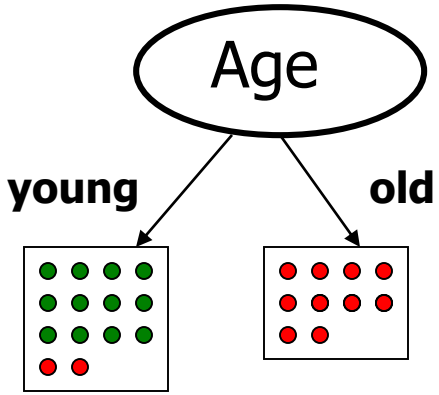
- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to determine the Best Split







Algorithm for Decision Tree Induction

- Basic algorithm
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - There are no remaining attributes for further partitioning
  - There are no samples left

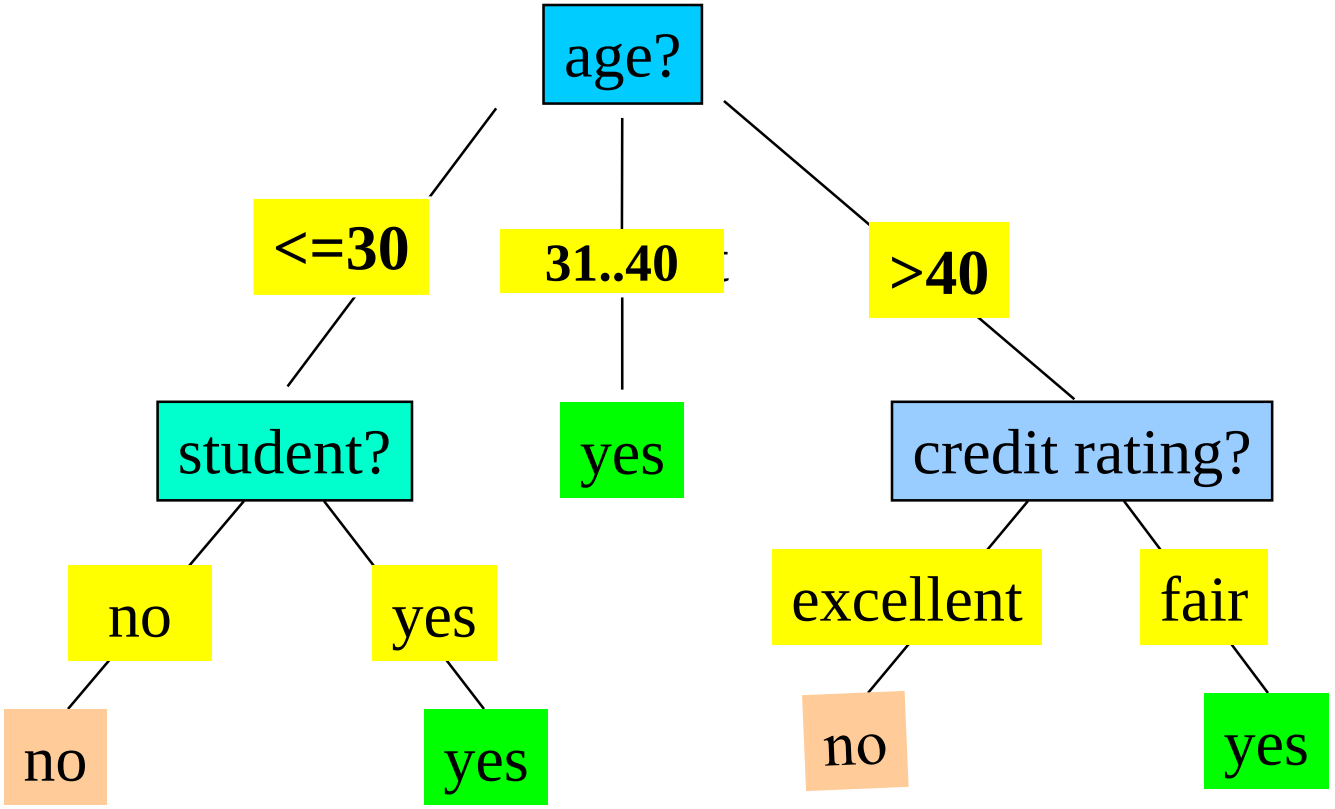
Classification Algorithms

- ID3
  - Uses information gain
- C4.5
  - Uses Gain Ratio

Decision Tree Induction: Training Dataset

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "*buys_computer*"



ID3
Attribute Selection Measure: Information Gain

- Notations:
  - Let D, the data partition, be a training set of class-labeled tuples.
  - Suppose the class label attribute has m distinct values defining m distinct classes, Ci (for $i = 1, \ldots, m$).
  - Let $C_{i,D}$ be the set of tuples of class Ci in D. Let $|D|$ and $|C_{i,D}|$ denote the number of tuples in D and $C_{i,D}$, respectively.

Attribute Selection Measure: Information Gain

- Select the attribute with the highest information gain for current node
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information needed to classify a given tuple in D:
  (log function base 2 is used since the info. is encoded in bits.)

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- *Info (D)* is just the average amount of information needed to identify the class label of a tuple in *D*. *Info (D)* is also known as the entropy of *D*.
- Now, suppose we were to partition the tuples in *D* on some attribute *A* having *v* distinct values, (*a*1, *a*2, …. , *av*), as observed from the training data. If *A* is discrete-valued, then it gives the v outcomes of a test on *A*. Attribute *A* can be used to split *D* into *v* partitions or subsets, (*D*1, *D*2, ….., *Dv).*

Attribute Selection Measure: Information Gain

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

- The term $|Dj| / |D|$ acts as the weight of the *j* th partition. *Info A*(*D*) is the expected information required to classify a tuple from *D* based on the partitioning by *A*.
- Information gained by branching on attribute A

Information gain = entropy (parent) – [weightes average] * entropy (children)

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection: Information Gain

- In training data set, The class level attribute, *buys_computer,* has two distinct values (namely, {yes,no}); therefore, there are two distinct classes (m=2).

- Let class P correspond to *yes* and N correspond to *no.*
- there are 9 samples of class yes and 5 samples of class no.
- To compute the information gain of each attribute, we first use Equation 1, to compute the expected information needed to classify a given sample.

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$\frac{5}{14} I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

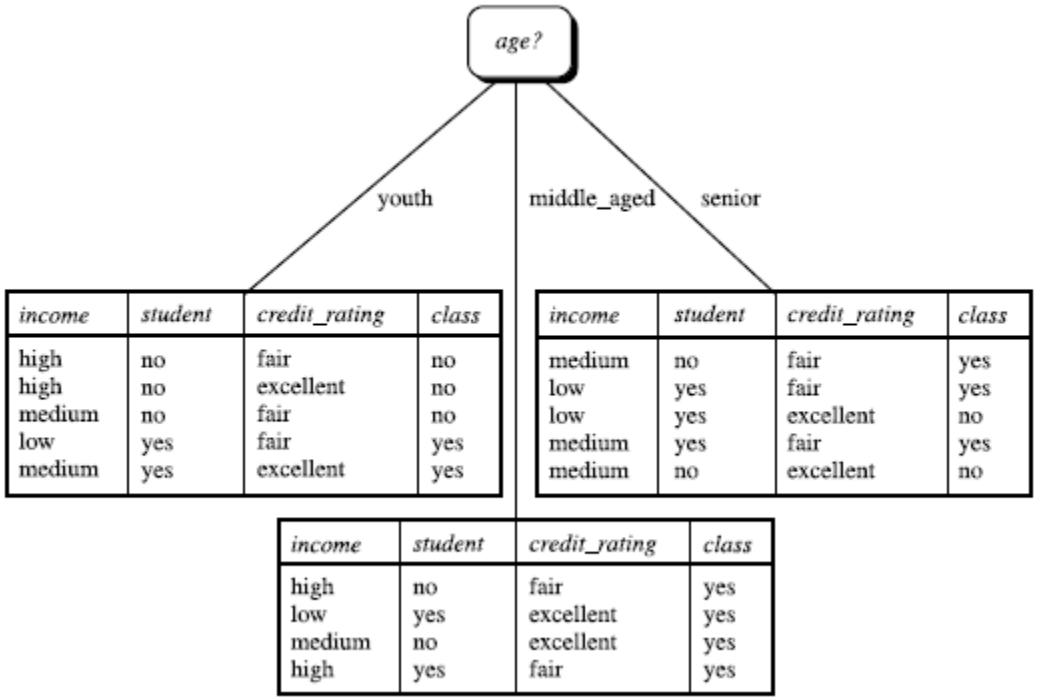
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

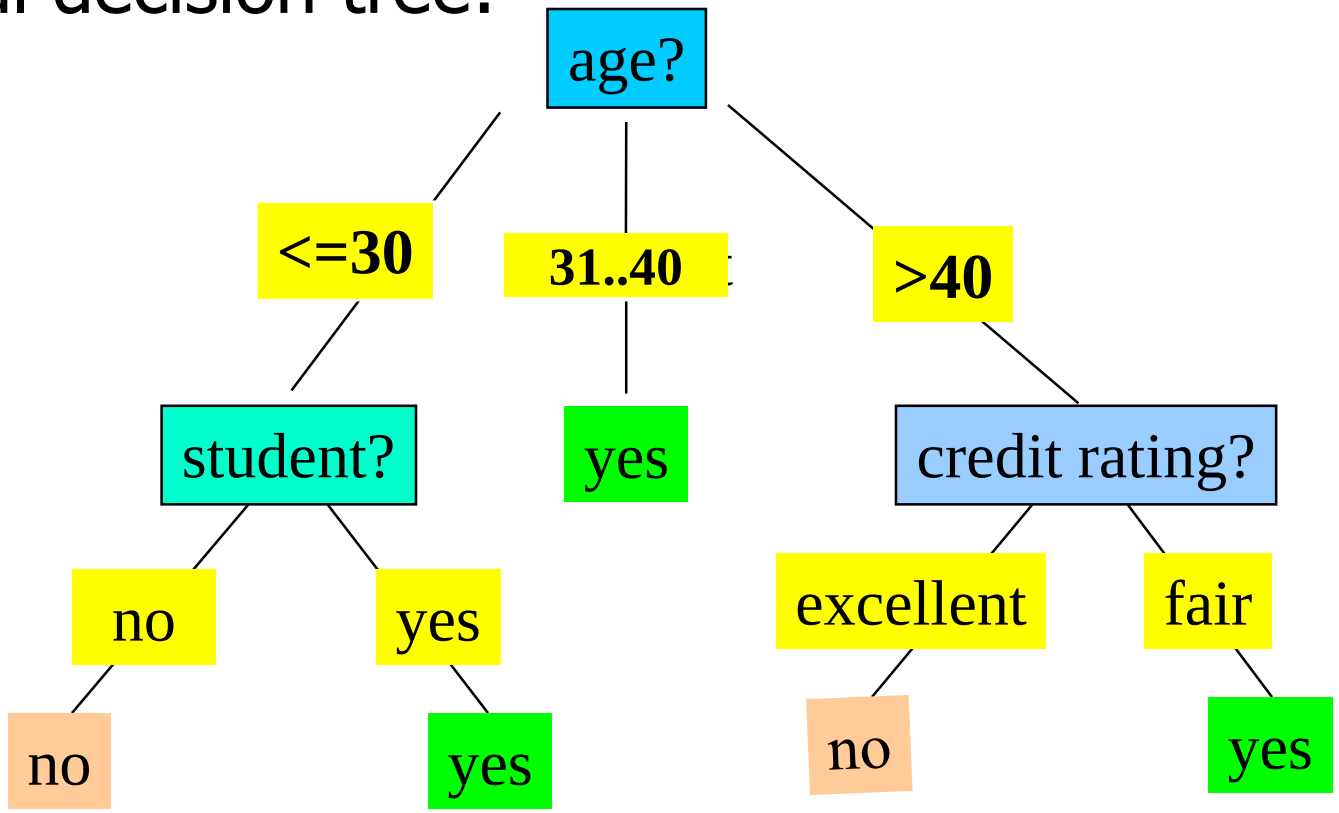
Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



- Final decision tree:



Why are decision tree classifiers so popular?

- The construction of decision tree classifiers does not require any domain knowledge or parameter setting, and therefore is appropriate for knowledge discovery.
- Decision trees can handle high dimensional data.
- Representation of acquired knowledge in tree form is generally easy to humans.
- The learning and classification steps of decision tree induction are simple and fast.
- In general, decision tree classifiers have good accuracy. However, successful use may depend on the data at hand.
- Decision tree induction algorithms have been used for classification in many application areas, such as medicine, manufacturing and production, financial analysis, and molecular biology.
- Decision trees are the basis of several commercial rule induction systems.

Gain Ratio for Attribute Selection

- The information gain measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values.

- For example, consider an attribute that acts as a unique identifier, such as *product ID*.

- A split on *product ID* would result in a large number of partitions (as many as there are values), each one containing just one tuple.

- Because each partition is pure, the information required to classify data set *D* based on this partitioning would be $Info_{product\ ID}(D) = 0$.

- Therefore, the information gained by partitioning on this attribute is maximal. Clearly, such a partitioning is useless for classification.

Gain Ratio for Attribute Selection

- C4.5, a successor of ID3, uses an extension to information gain known as *gain ratio.*
- (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)

(A test on income splits the data into three partitions, namely low, medium & high containing four,six & four tuples)

- Ex.

$$SplitInfo_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 0.926$$

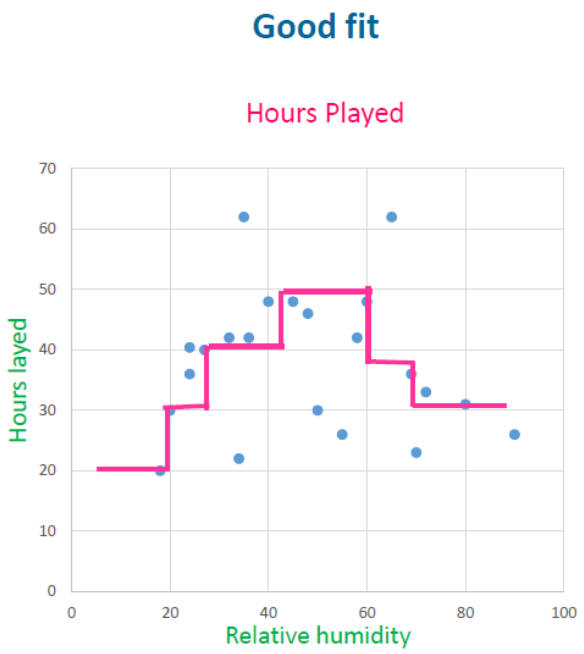
  - gain_ratio(income) = 0.029/0.926 = 0.031
- The attribute with the maximum gain ratio is selected as the splitting attribute

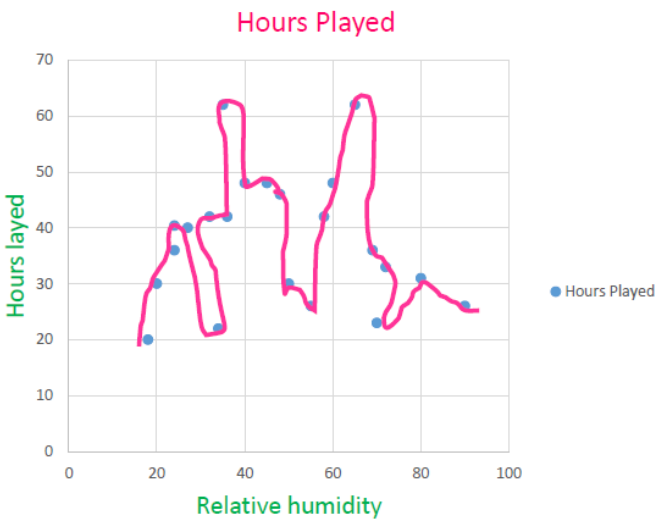
Comparing Attribute Selection Measures

- Information gain:
  - biased towards multi valued attributes
- Gain ratio:
  - tends to prefer unbalanced splits in which one partition is much smaller than the others

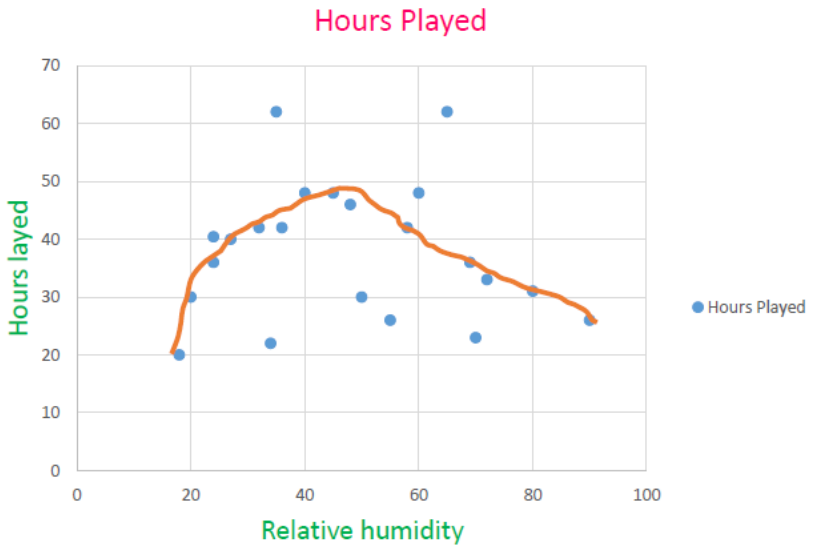
Challenge with Decision Tree models

Relative Humidity	Hours Played
55	26
50	30
60	48
58	42
65	62
70	23
48	46
69	36
72	33
45	48
40	48
35	62
80	31
90	26
27	40
36	42
24	40.4
32	42
24	36
34	22
18	20
20	30





Random Forest helps overcome this challenge



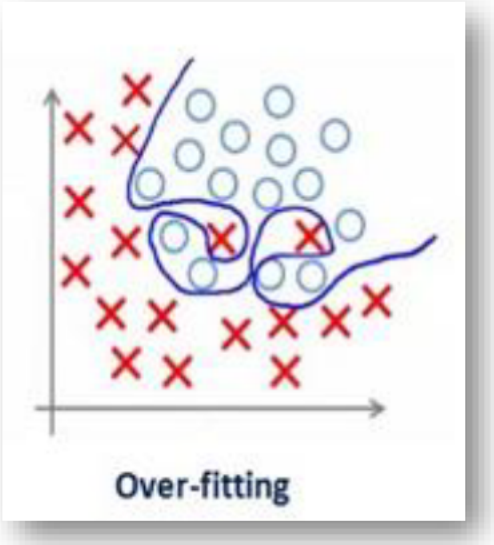
Random Forest algorithm can fit data points in such a smooth way that it neither overfits nor underfits model

Let's know what overfitting is How

How different will predictions of model be on unseen data.

Challenge with **over-fitting** model is that

- Model performs perfectly well(Overfitting) on **training** data
- But when same model is used for prediction on test data, it results in huge difference in prediction accuracy from that of training data.

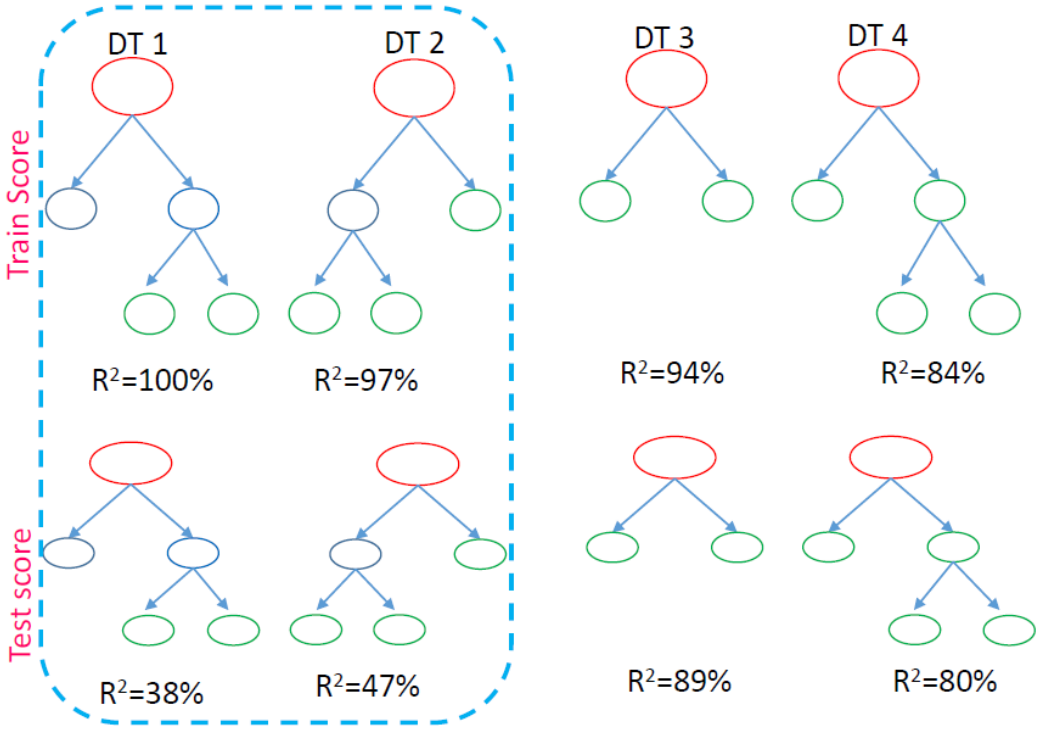


High variance

How overfitting causes challenge in Decision Tree

Random Forest to the rescue

Major advantage of Random Forest algorithm is to handle overfitting efficiently & outputs **overall better** result.



- A major drawback of *Decision tree* algorithm is
  - Decision Tree 1 & 2 have high variance(Overfitting)—performs relatively poor on test data.
- Random Forest aggregates all the trees and calculate **average score**.
- More like **averaging** out errors

R^2 is **a measure of the goodness of fit of a model**.

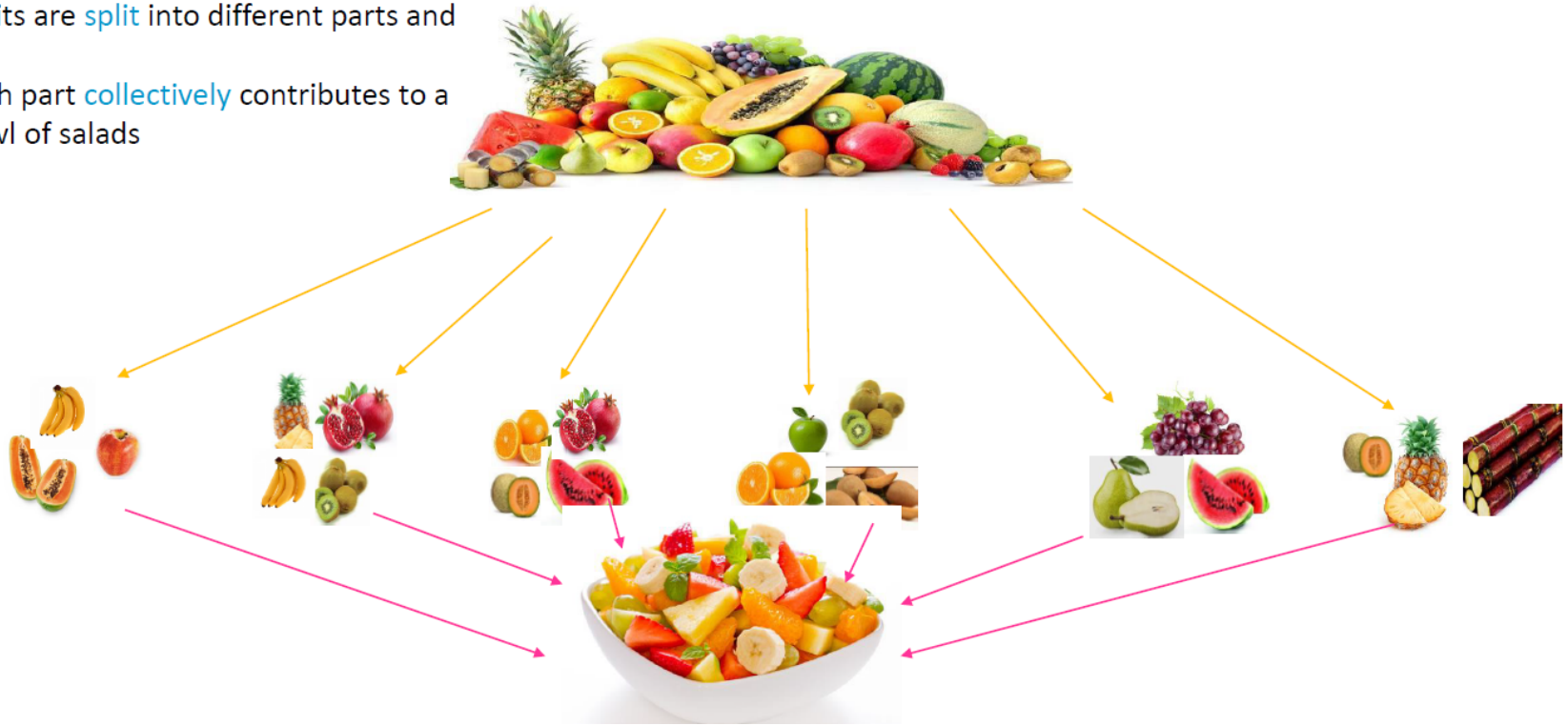
Random Forest

- Random forest is collection of many decision trees.
- Like forest is collection of many trees.

Random Forest algorithm is **example** of **bagging** technique.

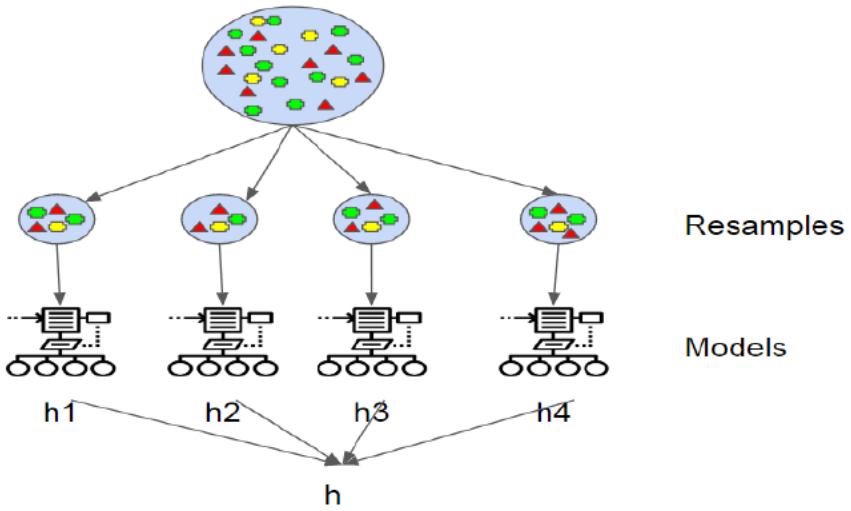
What is Bagging?

- Fruits are split into different parts and
- Each part collectively contributes to a bowl of salads

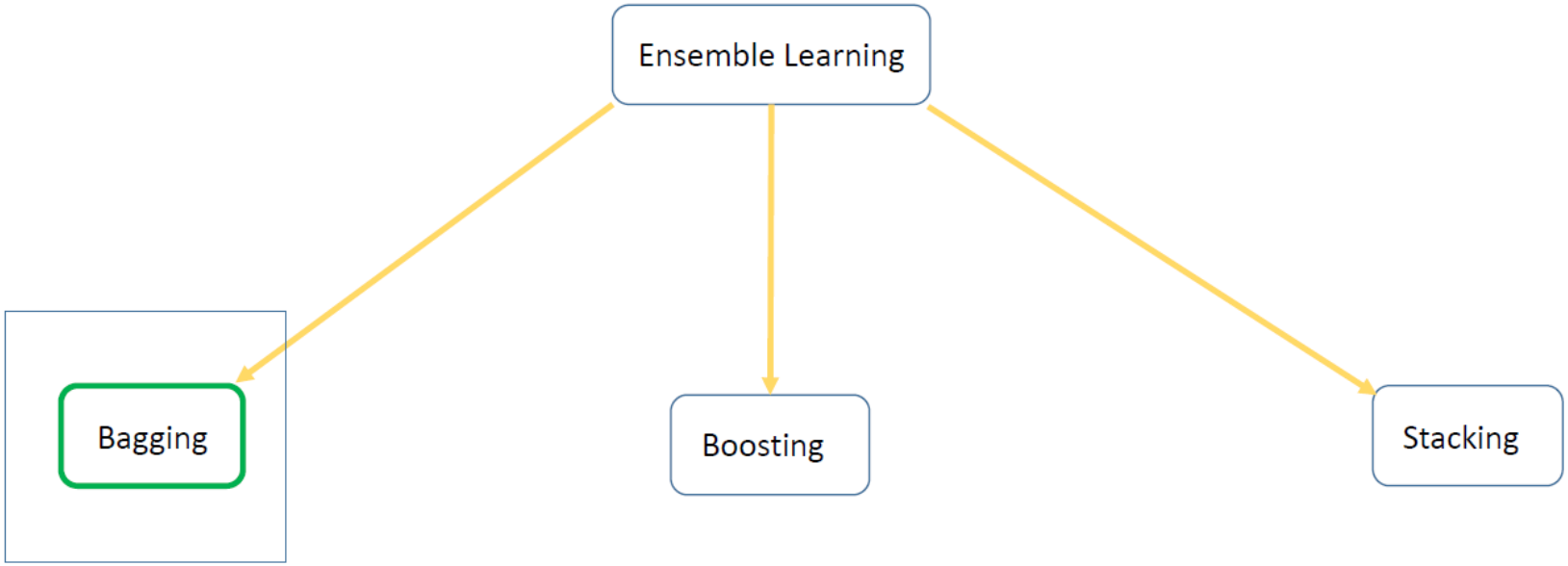


What is Bagging?

- Like fruits ,data are **split** into multiple **smaller** parts to make **multiple models**.
- Finally results from each model are **combined**.
  - **Majority** voting for classification
  - **Averaging** for regression

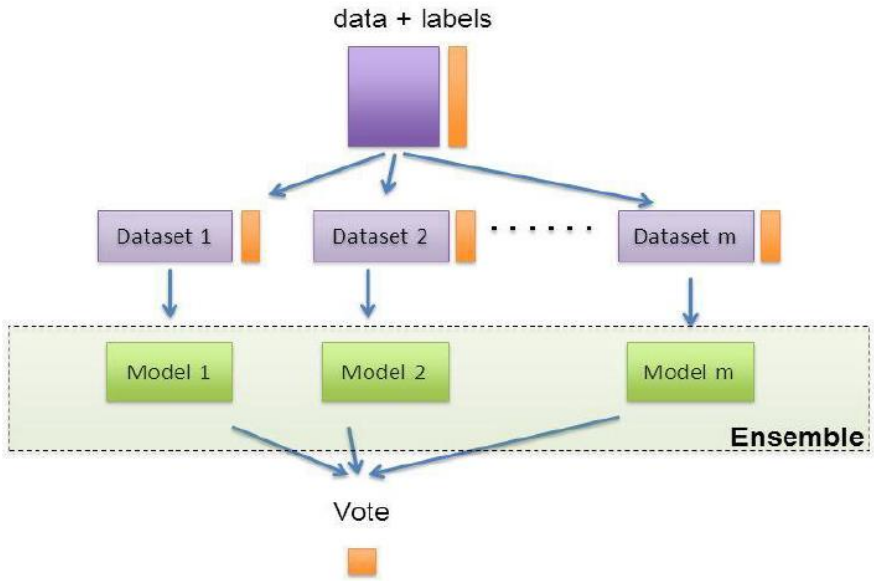


Types of Ensemble learning



What is ensemble learning?

Ensemble learning is technique that **creates** *multiple models* and then combines them to produce improved results



Ensemble

Group of individuals or items viewed as **whole** rather than individually.

Why Random Forest is called Random?

There are two types of randomness in RF algorithm—

1. Row level
2. Column level

Row level randomness in Random Forest

Let's say the data set has 1000 rows.

Row Level

- Each of decision trees will be made from **random sample**(For ex, 5%) of training data.
- It indicates that each decision tree will be made of **any 50 rows** of given data.

Passenger	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, M	male	22	1	0	A/5 2117	7.25		S
2	1	1	Cumings, I	female	38	1	0	PC 17599	71.28	C85	C
3	1	3	Heikkinen	female	26	0	0	STON/O2	7.925		S
4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr	male		0	0	330877	8.458		Q
7	0	1	McCarthy,	male	54	0	0	17463	51.86	E46	S
8	0	3	Palsson, M	male	2	3	1	349909	21.08		S
9	1	3	Johnson, M	female	27	0	2	347742	11.13		S
10	1	2	Nasser, Mr	female	14	1	0	237736	30.07		C
11	1	3	Sandstrom	female	4	1	1	PP 9549	16.7	G6	S
12	1	1	Bonnell, M	female	58	0	0	113783	26.55	C103	S
13	0	3	Saundercc	male	20	0	0	A/5. 2151	8.05		S
14	0	3	Anderssor	male	39	1	5	347082	31.28		S
15	0	3	Vestrom, M	female	14	0	0	350406	7.854		S
16	1	2	Hewlett, M	female	55	0	0	248706	16		S
17	0	3	Rice, Mast	male	2	4	1	382652	29.13		Q
18	1	2	Williams,	male		0	0	244373	13		S
19	0	3	Vander Pl	female	31	1	0	345763	18		S

Column level randomness in Random Forest

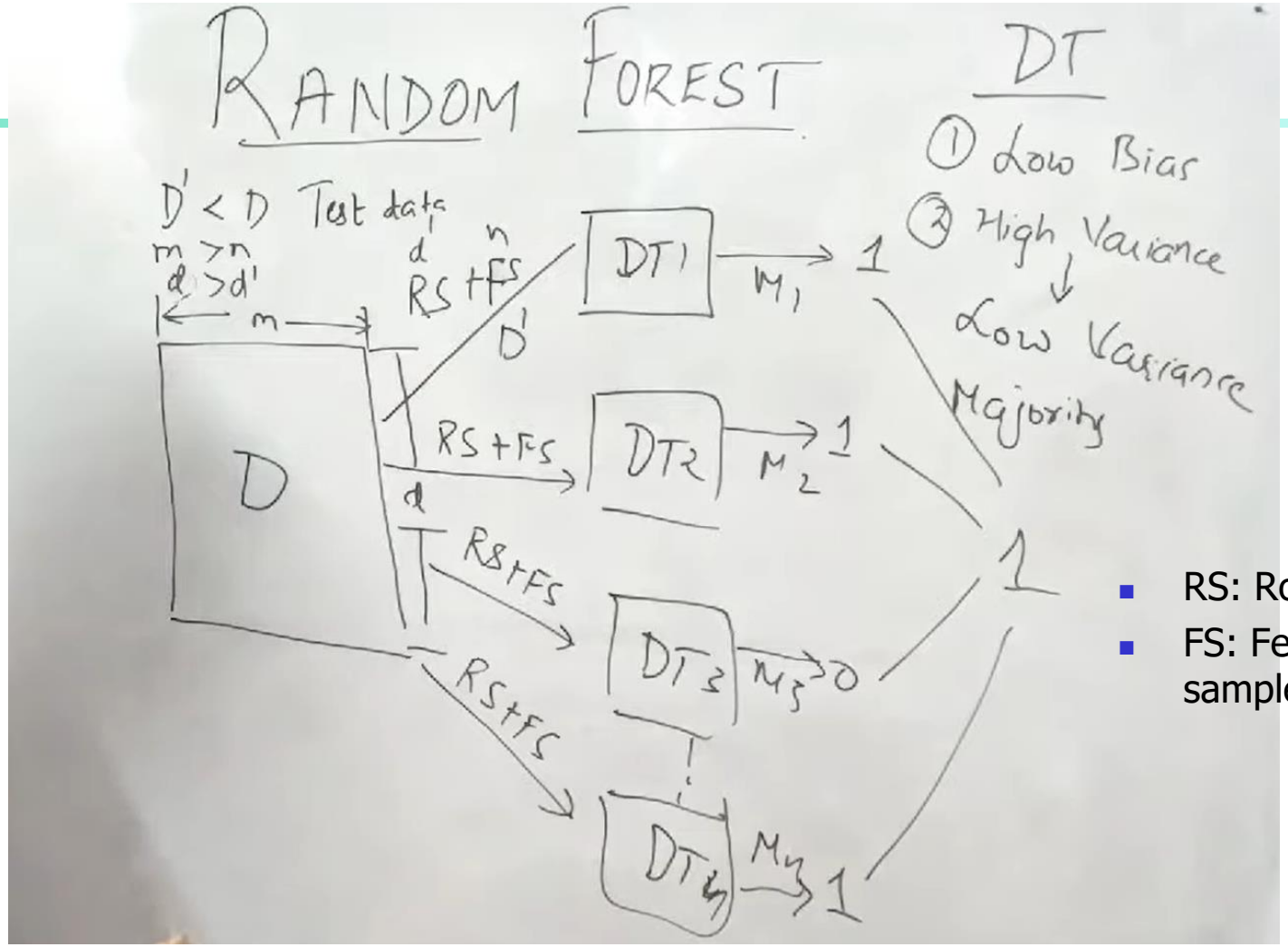
Let's say the data set has 50 columns.

Column Level

- At this level of randomness **not all the columns** are passed into training each decision tree,
- rather specified number(For ex, 10%) of columns i.e. 5 randomly chosen columns will be passed into each decision tree.

Passenger	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, M	male	22	1	0	A/5 2117	7.25		S
2	1	1	Cumings, I	female	38	1	0	PC 17599	71.28	C85	C
3	1	3	Heikkinen	female	26	0	0	STON/O2	7.925		S
4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr	male		0	0	330877	8.458		Q
7	0	1	McCarthy,	male	54	0	0	17463	51.86	E46	S
8	0	3	Palsson, M	male	2	3	1	349909	21.08		S
9	1	3	Johnson, M	female	27	0	2	347742	11.13		S
10	1	2	Nasser, Mr	female	14	1	0	237736	30.07		C
11	1	3	Sandstrom	female	4	1	1	PP 9549	16.7	G6	S
12	1	1	Bonnell, M	female	58	0	0	113783	26.55	C103	S
13	0	3	Saundercc	male	20	0	0	A/5. 2151	8.05		S
14	0	3	Anderssor	male	39	1	5	347082	31.28		S
15	0	3	Vestrom, I	female	14	0	0	350406	7.854		S
16	1	2	Hewlett, M	female	55	0	0	248706	16		S
17	0	3	Rice, Mast	male	2	4	1	382652	29.13		Q
18	1	2	Williams,	male		0	0	244373	13		S
19	0	3	Vander Pl	female	31	1	0	345763	18		S

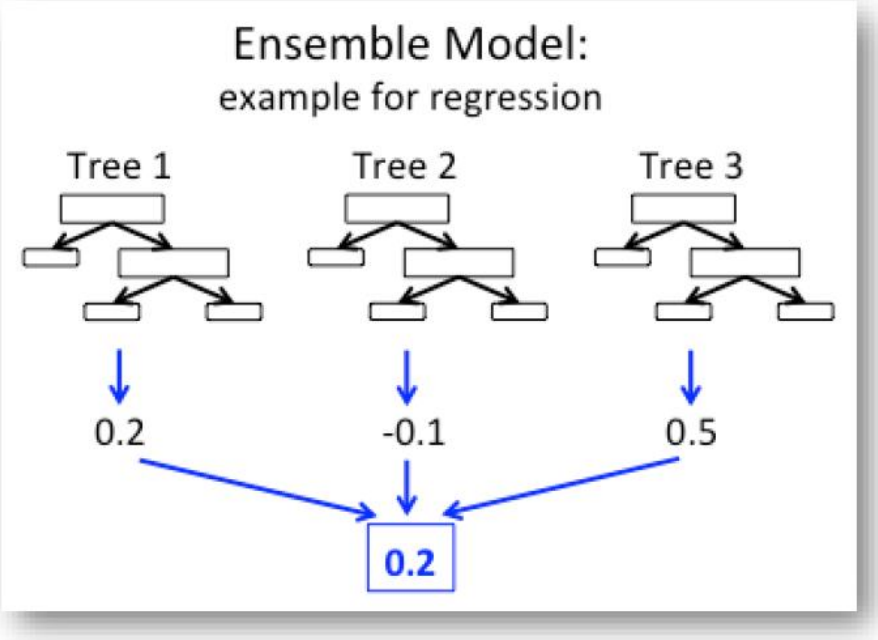
Example



- Low Bias: Basically it says that if I am creating my decision tree to its complete depth, then it will get properly trained for training dataset. So training error will be very less.
- High Variance: Whenever we get new test data, the decision tree is prone to give larger amount of error.

How does Random Forest work in Regression?

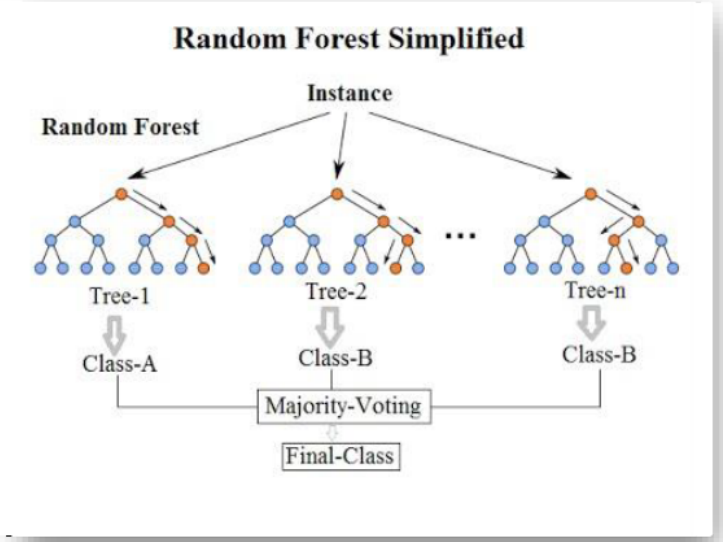
Random Forest Regression



- In Random Forest regression all values of decision trees are averaged.
- The average value is actual result for regression.

How does Random Forest work in Classification?

Random Forest classification



- To classify a new object based on attributes each tree gives classification, often termed as **tree votes** for that class.
- A class is chosen based on **majority** of voting.
- For ex, in picture left if major class is B then it'll be predicted as B.

Benefits of Random Forest

Use cases of Random Forest

Healthcare

- In healthcare domain it is used **to identify correct combination** of components in medicine
- Analyzes a patient's medical history to **identify** diseases

Finance

- In finance it **predicts prospect defaulters.**
- Used to detect **unusual transaction** in anomaly detection

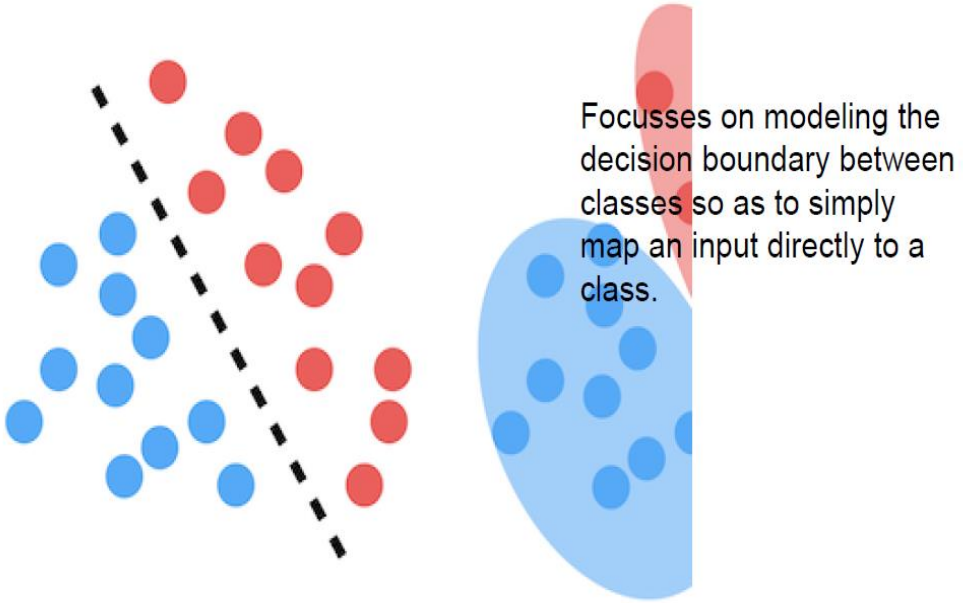
Naïve Bayes classifier

Naïve Bayes classifier

- Introduction to Naïve Bayes Classifier
- Basics of Probability
- Understanding Naïve Bayes Classifier
- Classify SMS message to be spam or not

Background

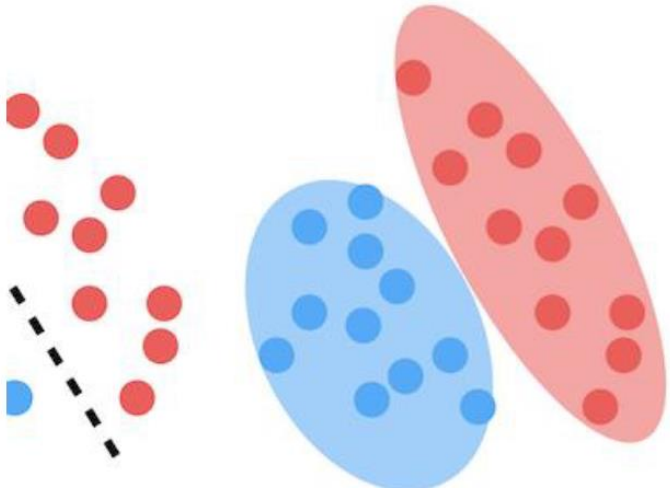
- Classification algorithms that differentiates between classes on the basis of definite decision boundaries.
- Classification algorithms that learn boundaries between classes.
- Classification algorithms that constructs decision boundaries that separates classes are called **discriminative models**.



Background

- What if we differentiate between two classes by analyzing probability distribution of data…

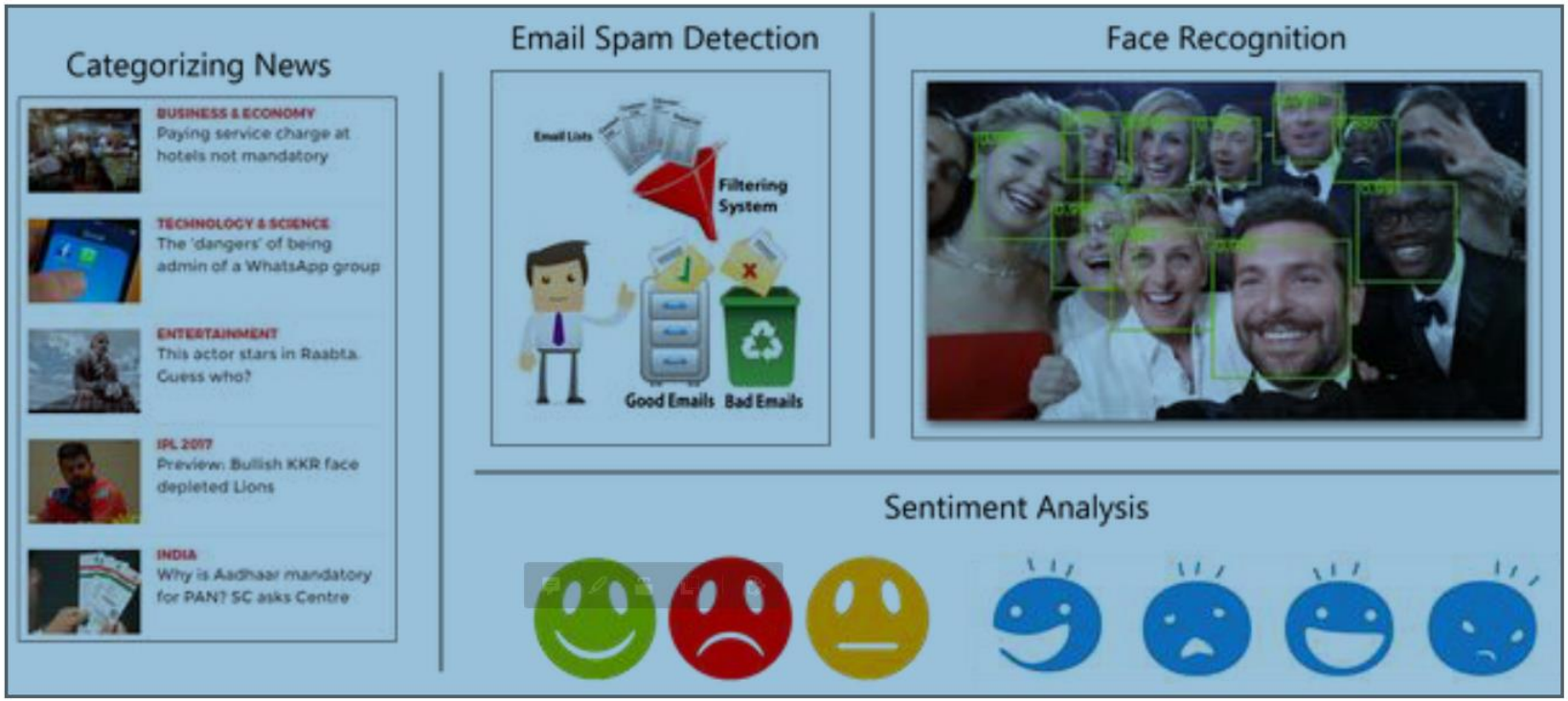
Let us build a model A that characterizes the red dots and a model B that characterizes the blue dots using probability.
Any new data point can be assigned to either class based on how likely it is to belong to either A or B. These are called generative models.



What is Naïve Bayes?

- Naive Bayes classifier is an algorithm that learns the probability that an object with certain features, belong to a particular group or class.

Where is Naïve Bayes used?



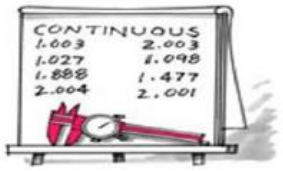
Advantages of Naïve Bayes

Simple and Easy to implement

Fast

Needs lesser training data

Scalable



Handles both continuous and discrete data

Basics of Probability

- What is Probability?

Estimate How **likely** something is to happen.

Rain is likely to fall

Obtaining a movie ticket

Will Arsenal win?

Stocks are likely to rise or fall

Boy or girl?

What is Probability?

We draw on previous experience to determine how likely something is to occur

40%

60%

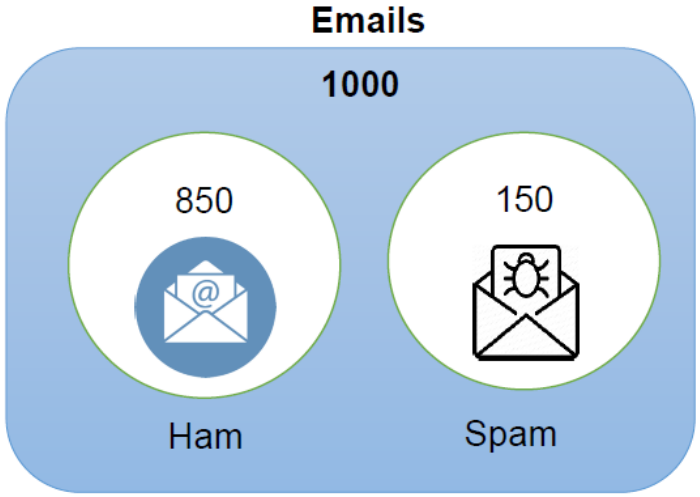
0.7

70%

0.5

Probability explained through an example

John's email id has a total of 1000 emails, 850 emails are genuine whereas 150 mails are spam.



Probability of a spam mail appearing in John's account is:

$$P(\text{spam}) = \frac{\text{Number of spam emails}}{\text{Total number of emails}} = \frac{150}{1000}$$

$$= 0.15 = 15\%$$

John’s emails have multiple occurrences of the word ‘Lottery’. Let’s analyze them closely..

Analyze Emails with word “lottery”

Scrolling through John’s mails we see the word ’lottery’ appearing frequently.

From: Information Desk <info@euroonlinelottery.com>
Subject: EU / Commonwealth Lottery Promotions

Your email address was selected to claim the sum of $ 500,000.00 in the 2011 European lottery.
To claim your prize, please contact our agent in Lagos, Nigeria.
Contact person: Mr. Marshall Ellis e-mail: marshallellis11@live.com
Phone: +2348036954742
Congratulations!
Vincent Kilkenny (Coordinator)

The National Lottery

Congratulations! Winning Notification!

UK online international Lottery Award Prize of £820,731.00 (Eight Hundred and Twenty Thousand, Seven Hundred and Thirty One British Pounds).

Email Account Owner,

Congratulations!! We are happy to announce that you have won an online lottery prize in our international lottery promotion.

Your active e-mail address attached to computer generated **ticket number: BH 05607545 7152** with reference number **UK/KA2C110EN5** has won UK Lottery 2nd category award prize.

Contact our Fiduciary agents immediately to commence release of your lottery prize by providing details as listed below.
1. Full Name:
2. Email Address:
3. Age/Occupation:
4. Reference Number/Ticket Number
5. Phone Number:
6. Country:
7. Date of draw

UK Lottery Fiduciary Agents:
Mr. Peter Wills
Foreign Service Manager
Watford Regional Centre
Tolpits Lane, Watford WD18 9RN .United Kingdom
E-mail: agent.claim@natlott.com
wiliam_boyd@yahoo.co.uk

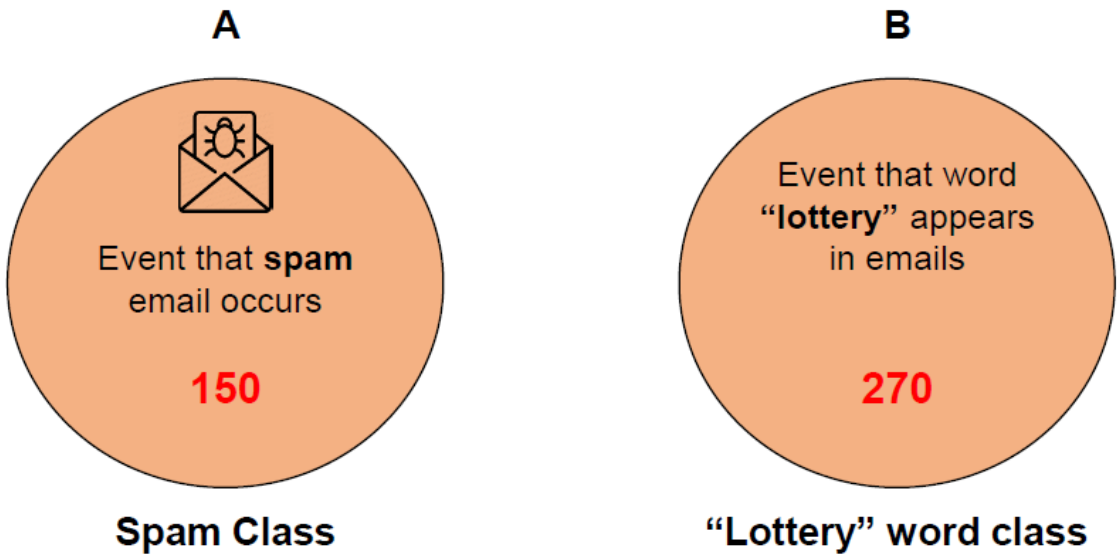
Thank you and Congratulation once again!

Yours faithfully,
Angela M. Johnson.
(Online coordinator)
The UK Lottery International Promotion Inc.

"Please do not reply back to the senders address or the from email address, this notification is sent automatically via computer virtual notification to winning email addresses and a response will not be attended by Human but computer" contact the fiduciary agents as above "

Let us consider two simple events..

Let us consider two simple events in Emails



John's mail id consists of:

- **150** spam emails
- **270** emails containing the word "Lottery"

Appearance of "lottery" in spam and genuine emails

Frequency table of "lottery" word occurring in emails

	Emails containing "lottery"	Emails not containing "lottery"	Total number of emails
Number of Spam emails	140	10	**150**
Number of Ham emails	130	720	**850**
Total number of emails	**270**	**730**	**1000**

Compute probability of word 'lottery' appearing in emails

27% of emails have the word "**lottery**".

P(lottery)=270/1000=0.27

	Emails containing "lottery"	Emails not containing "lottery"	Total number of emails
Number of Spam emails	140	10	**150**
Number of Ham emails	130	720	**850**
Total number of emails	**270**	**730**	**1000**

Let us explore different types of probabilities…

Types of Probabilities: Joint Probability

Types of Probabilities: Joint Probability

Joint probability represents **probability of two different events occurring together** at the same point in time.

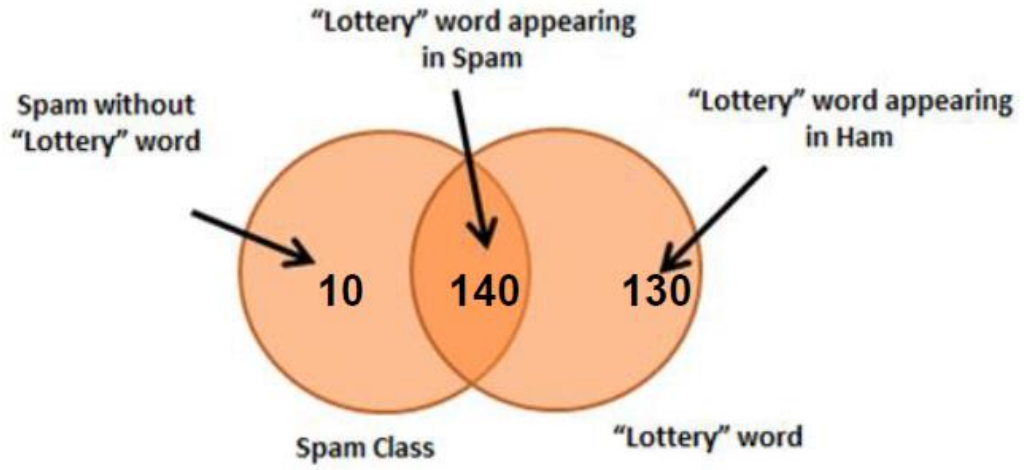
We have two events

A: **Event** that **Spam email** occurs and
B: **Event** that **word "lottery"** appears in an email

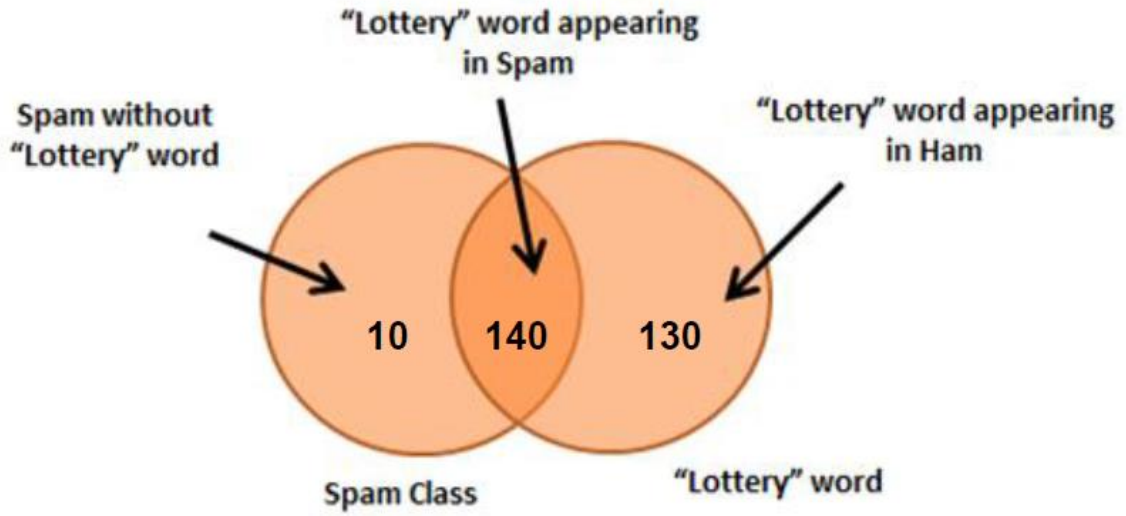
And let us assume that they are Independent events.

Venn Diagram for representing count of events

	Emails containing "lottery"	Emails not containing "lottery"	Total number of emails
Number of Spam emails	140	10	**150**
Number of Ham emails	130	720	**850**
Total number of emails	**270**	**730**	**1000**



Let us compute joint probability of word 'lottery' appearing in spam



$$P(Spam, Lottery) = 140/1000 = 0.14$$

14% of total emails are spam and contain the word lottery.

Types of Probabilities: Marginal Probability

Types of Probabilities: Marginal Probability

Probability of a single event occurring, irrespective of any other event is called marginal probability.

A: Spam email occurs

P(A)=150/1000=0.15

***B*: Word lottery appears in email**

P(B)=270/1000=0.27

Frequency table of "lottery" word occurring in email

	Emails containing "lottery"	Emails not containing "lottery"	Total number of emails
Number of Spam emails	140	10	150
Number of Ham emails	130	720	850
Total number of emails	270	730	1000

Types of Probabilities: Conditional Probability

Types of Probabilities: Conditional Probability

Suppose

A: Event that **a spam email occurs** and
B: Event that **word lottery** appears in an email

are dependent events.

How do we Predict whether email is spam given the word lottery?

Conditional Probability is a simple way to calculate probability of uncertain events given some prior information.

Probability of **an event** given that **another event** has **already occurred** is called **conditional probability**.

Conditional Probability

- For example, suppose you go out for lunch at the same place and time every Friday and you are served lunch within 15 minutes with probability 0.9. However, given that you notice that the restaurant is exceptionally busy, the probability of being served lunch within 15 minutes may reduce to 0.7. This is the conditional probability of being served lunch within 15 minutes given that the restaurant is exceptionally busy.
- The usual notation for "event A occurs given that event B has occurred" is "A | B" (A given B). The symbol | is a vertical line and does not imply division.
- P(A | B) denotes the probability that event A will occur given that event B has occurred already.

Conditional Probability

- A rule that can be used to determine a conditional probability from unconditional probabilities is:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

- where:
- P(A | B) = the (conditional) probability that event A will occur given that event B has occurred already.
- P(A∩ B) = the (unconditional) probability that event A and event B both occur.
- P(B) = the (unconditional) probability that event B occurs.

Naive Bayes classifier

- Bayesian classifiers are statistical classifiers. They can predict class membership probabilities, such as the probability that a given tuple belongs to a particular class.
- Bayesian classification is based on Bayes' Theorem.
- It is based on simplifying assumpations that the attribute values are *conditionally independent,*
- A naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature, given the class variable.

Naive Bayes classifier

- For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. *A naive Bayes classifier considers all these features to contribute independently to the probability that this fruit is an apple*, whether or not they're in fact related to each other or to the existence of the other features.

- This reduces significantly computation cost since calculating each one of the $P(a_i|v_j)$ requires only a frequency count over the tuples in the training data with class value equal to v_j .

Bayes Theorem : Basics

- Let **X** be a data sample : class label is unknown
- Let H be a *hypothesis* that X belongs to a specified class C
- For classification problems, we want to determine P(H|**X**), the probability that the hypothesis holds given the observed data sample **X**
- P(H) (*prior probability*), the initial probability
  - E.g., **X** will buy computer, regardless of age, income or any other information, for that matter.
- P(H|X) (*posteriori probability*), the probability of observing the sample **X**, given that the hypothesis holds
  - For example, suppose our world of data tuples is confined to customers described by the attributes *age* and *income*, respectively,
  - and that ***X*** is a 35-year-old customer with an income of $40,000.
  - Suppose that *H* is the hypothesis that our customer will buy a computer.
  - Then P(H|X) reflects the probability that customer ***X*** will buy a computer given that we know the customer's age and income.

Bayesian Theorem

- Given data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})}$$

- *P*(***X***|*H*) is the descriptor posterior probability of ***X*** conditioned on *H*. That is, it is the probability that a customer, ***X***, is 35 years old and earns $40,000, given that we know the customer will buy a computer.

- Predicts **X** belongs to Ci if the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes.
- Practical difficulty: require initial knowledge of many probabilities.

Bayesian Theorem

- Assume target function $f : X \to Y$ (A function *f* with domain *X* and codomain *Y*). The elements of *X* are called arguments of *f*. For each argument *x*, the corresponding unique *y* in the codomain is called the function value at *x* or the *image* of *x* under *f*.
- If, each instance X is describes by attributes <a1, a2, a3, ….an>.
- Most probable value of f(X) is: v_{MAP}
- Using *Bayes Theorem* we can write the expression as :

$$v_{MAP} = \arg \max_{v_j \in V} P\left(v_j \middle| a_1, a_2, ..., a_n\right)$$

$$= \arg \max_{v_j \in V} \frac{P\left(a_1, a_2, ..., a_n \middle| v_j\right) P\left(v_j\right)}{P\left(a_1, a_2, ..., a_n\right)}$$

$$= \arg \max_{v_j \in V} P\left(a_1, a_2, ..., a_n \middle| v_j\right) P\left(v_j\right)$$

$$v_{MAP} = \arg \max_{v_j \in V} P\left(v_j\right) \prod_i P\left(a_i \middle| v_j\right)$$

- The *denominator* does not depend on the choice of v_j and thus, it can be omitted from the arg max argument.

Bayesian Theorem

- In mathematics, **argmax** stands for the **argument of the maximum**, that is to say, the set of points of the given argument for which the given function attains its maximum value.

Towards Naïve Bayesian Classifier

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n- dimensional attribute vector **X** = (x_1, x_2,..., x_n), showing *n* measurements made on the tuple from *n* attributes.
- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

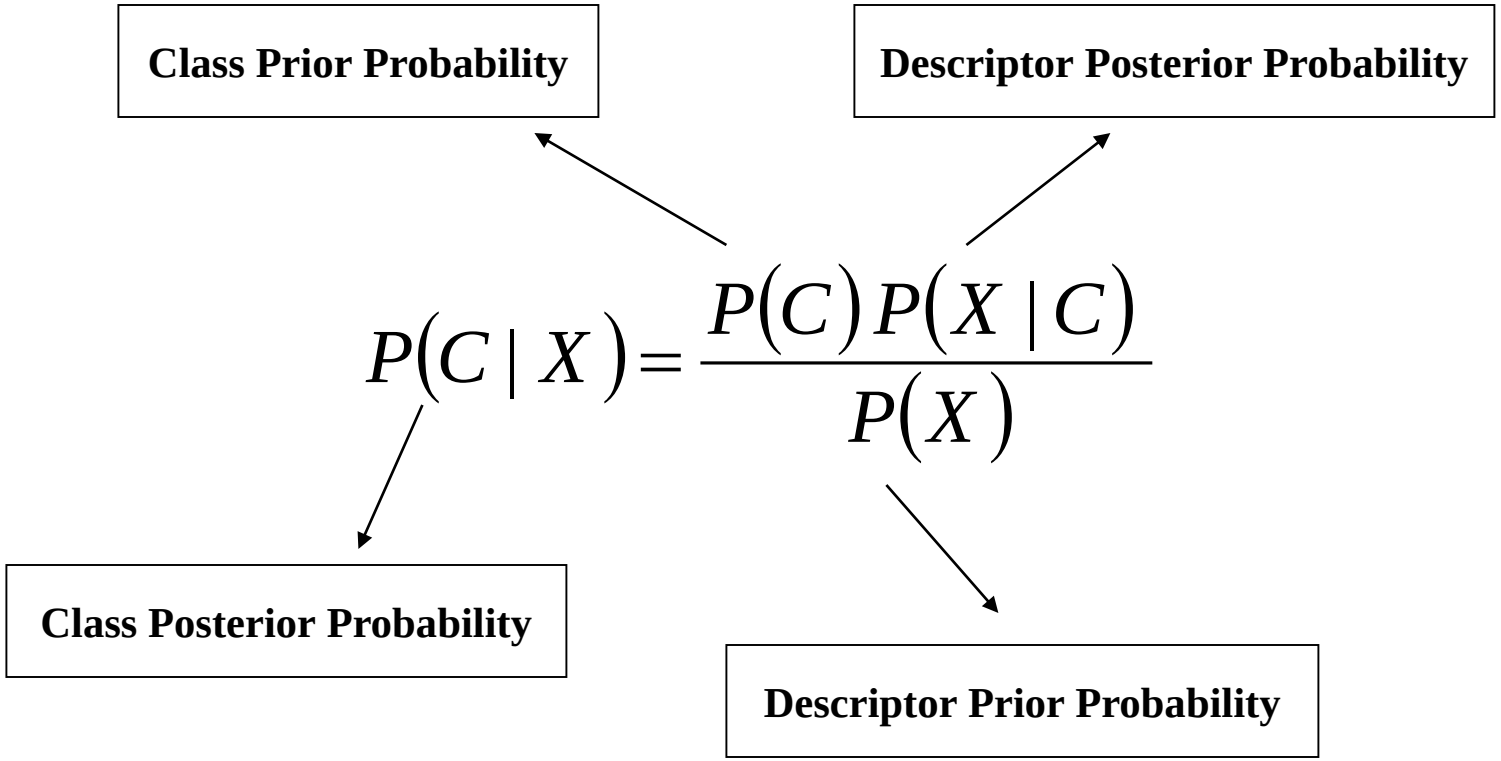
$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Bayesian Classifier – Basic Equation



Naïve Bayesian Classifier: Training Dataset

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

Data sample
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier: An Example

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357

- Compute $P(X|C_i)$ for each class
  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Training Data

Outlook	Temp	Humidity	Windy	Play?
sunny	hot	high	FALSE	No
sunny	hot	high	TRUE	No
overcast	hot	high	FALSE	Yes
rainy	mild	high	FALSE	Yes
rainy	cool	normal	FALSE	Yes
rainy	cool	Normal	TRUE	No
overcast	cool	Normal	TRUE	Yes
sunny	mild	High	FALSE	No
sunny	cool	Normal	FALSE	Yes
rainy	mild	Normal	FALSE	Yes
sunny	mild	normal	TRUE	Yes
overcast	mild	High	TRUE	Yes
overcast	hot	Normal	FALSE	Yes
rainy	mild	high	TRUE	No

P(yes) = 9/14
P(no) = 5/14

Bayesian Classifier – Probabilities for the weather data

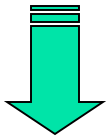
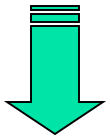
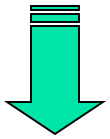
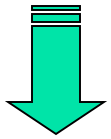
Frequency Tables

Outlook	No	Yes
Sunny	3	2
Overcast	0	4
Rainy	2	3

Temp.	No	Yes
Hot	2	2
Mild	2	4
Cool	1	3

Humidity	No	Yes
High	4	3
Normal	1	6

Windy	No	Yes
False	2	6
True	3	3

Outlook	No	Yes
Sunny	3/5	2/9
Overcast	0/5	4/9
Rainy	2/5	3/9

Temp.	No	Yes
Hot	2/5	2/9
Mild	2/5	4/9
Cool	1/5	3/9

Humidity	No	Yes
High	4/5	3/9
Normal	1/5	6/9

Windy	No	Yes
False	2/5	6/9
True	3/5	3/9

Likelihood Tables

Bayesian Classifier – Predicting a new day

Outlook	Temp.	Humidity	Windy	Play
sunny	cool	high	true	NO

X →

Class?

P(**yes**|**X**) = p(sunny|yes) x p(cool|yes) x p(high|yes) x p(true|yes) x **p(yes)**

= 2/9 x 3/9 x 3/9 x 3/9 x **9/14** = 0.0053 => 0.0053/(0.0053+0.0206) = 0.205

P(**no**|**X**) = p(sunny|no) x p(cool|no) x p(high|no) x p(true|no) x **p(no)**

= 3/5 x 1/5 x 4/5 x 3/5 x **5/14** = 0.0206=0.0206/(0.0053+0.0206) = 0.795

Outlook	No	Yes
Sunny	3/5	2/9
Overcast	0/5	4/9
Rainy	2/5	3/9

Temp.	No	Yes
Hot	2/5	2/9
Mild	2/5	4/9
Cool	1/5	3/9

Humidity	No	Yes
High	4/5	3/9
Normal	1/5	6/9

Windy	No	Yes
False	2/5	6/9
True	3/5	3/9

Bayesian Classifier – zero frequency problem

- What if a descriptor value doesn't occur with every class value

  P(outlook=overcast|No)=0

- Remedy: add 1 to the count for every descriptor-class combination (Laplace Estimator)

Outlook	No	Yes
Sunny	3+1	2+1
Overcast	0+1	4+1
Rainy	2+1	3+1

Temp.	No	Yes
Hot	2+1	2+1
Mild	2+1	4+1
Cool	1+1	3+1

Humidity	No	Yes
High	4+1	3+1
Normal	1+1	6+1

Windy	No	Yes
False	2+1	6+1
True	3+1	3+1

Bayesian Classifier – General Equation

$$P(C_k \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C_k)\,P(C_k)}{P(\mathbf{X})}$$

Likelihood: $P(\mathbf{X} \mid C_k)$

Continues variable: $$P(x \mid C) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ -\frac{(x-\mu)^2}{2\sigma^2} \right\}$$

Bayesian Classifier – Dealing with numeric attributes

- Usual assumption: attributes have a *normal* or *Gaussian* probability distribution (given the class)
- The *probability density function* for the normal distribution is defined by two parameters:
  - The *sample mean* μ: $$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$
  - The *standard deviation* $$\sigma = \left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2\right]^{0.5}$$
  - The density function *f(x)*: $$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

EXAMPLE-I

Department	status	age	salary
Sales	senior	31. . .35	41K.. .45K
Sales	junior	26. . .30	26K.. .30K
Sales	junior	31. . .35	31K.. .35K
systems	junior	21. . .25	31K.. .35K
systems	senior	31. . .35	66K.. .70K
systems	junior	26. . .30	31K.. .35K
systems	senior	41. . .45	66K.. .70K
marketing	senior	26. . .30	46K.. .50K
marketing	junior	31. . .35	41K.. .45K
secretary	senior	46. . .50	41K.. .45K
secretary	junior	26. . .30	26K.. .30K

- Define Bayesian Classification .Given a **data tuple** having the values *"systems", "26. . . 30",* and "41K.. .45K*"* for the attributes *department, age*, and *salary*, respectively, what would be a naive Bayesian classification of the status for the **given data tuple** ?

Example- continuous attributes

- Consider the training dataset as shown in below table. Let ***Play*** be the class label attribute. There are two distinct classes, namely, ***yes*** and ***no*** and two numeric attributes namely "temp" and "humidity".

Outlook	Temp	Humidity	Windy	Play?
sunny	85	85	FALSE	**No**
sunny	80	90	TRUE	**No**
overcast	83	86	FALSE	**Yes**
rainy	70	96	FALSE	**Yes**
rainy	68	80	FALSE	**Yes**
rainy	65	70	TRUE	**No**
overcast	64	65	TRUE	**Yes**
sunny	72	95	FALSE	**No**
sunny	69	70	FALSE	**Yes**
rainy	75	80	FALSE	**Yes**
sunny	75	70	TRUE	**Yes**
overcast	72	90	TRUE	**Yes**
overcast	81	75	FALSE	**Yes**
rainy	71	91	TRUE	**No**

- Given a **data tuple** having the values *"sunny", 66, 89 and "true"* for the attributes outlook*, temp.*, humidity and windy respectively, what would be a naive Bayesian classification of the *Play* for the given tuple?

Example- continuous attributes

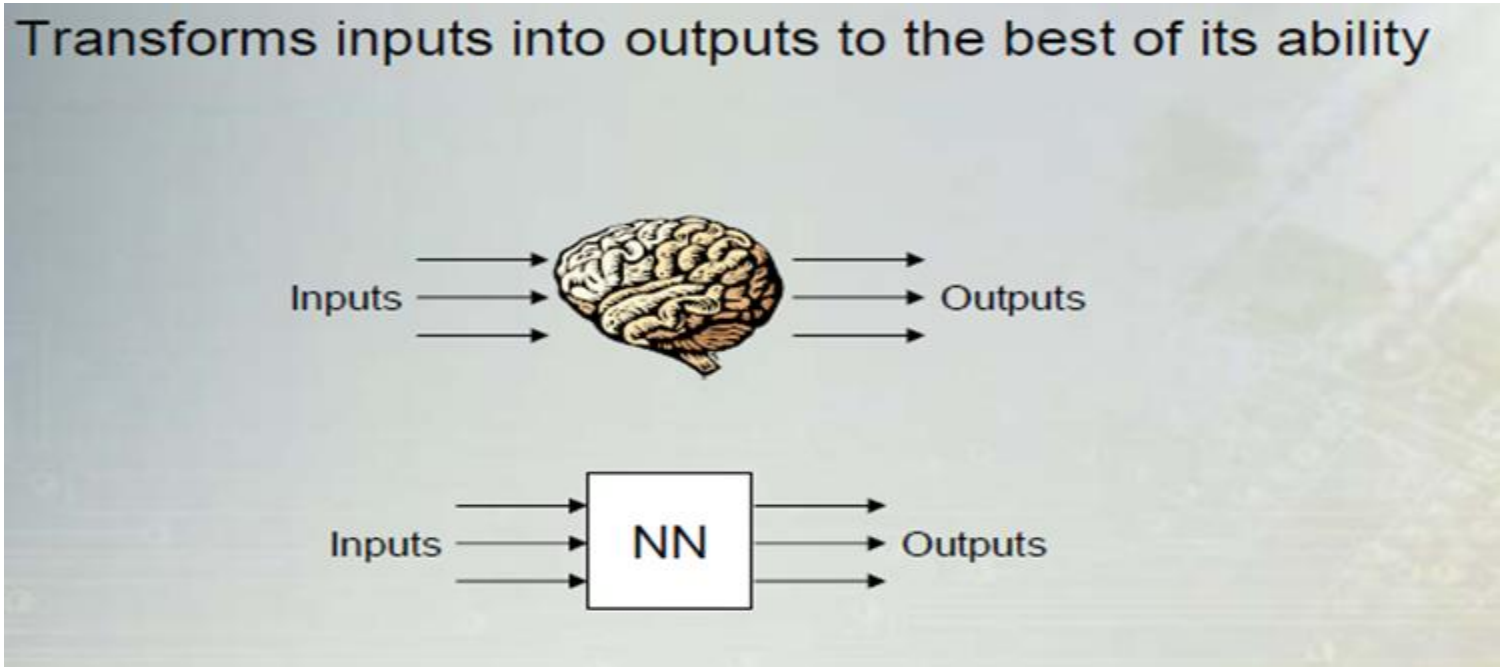
The numeric weather data with summary statistics

Outlook			temperature			humidity			windy		play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3		83	85		86	85	false	6	2	9	5
overcast	4	0		70	80		96	90	true	3	3		
rainy	3	2		68	65		80	70					
				64	72		65	95					
				69	71		70	91					
				75			80						
				75			70						
				72			90						
				81			75						
sunny	2/9	3/5	mean	73	74.6	mean	79.1	86.2	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	std dev	6.2	7.9	std dev	10.2	9.7	true	3/9	3/5		
rainy	3/9	2/5											

Artificial Neural Networks (ANN)

Neural Networks -Origin

- Inspired by the way biological nervous systems, such as the brain, process information.
- Key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems.



Best learning system known to us?

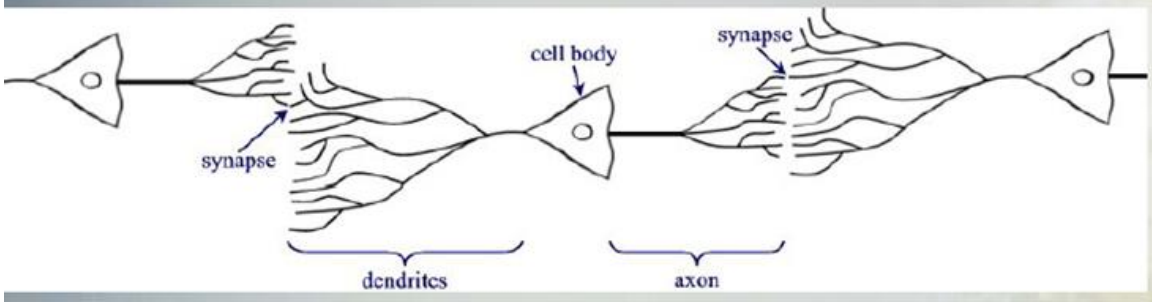
Biological system

?

Artificial system

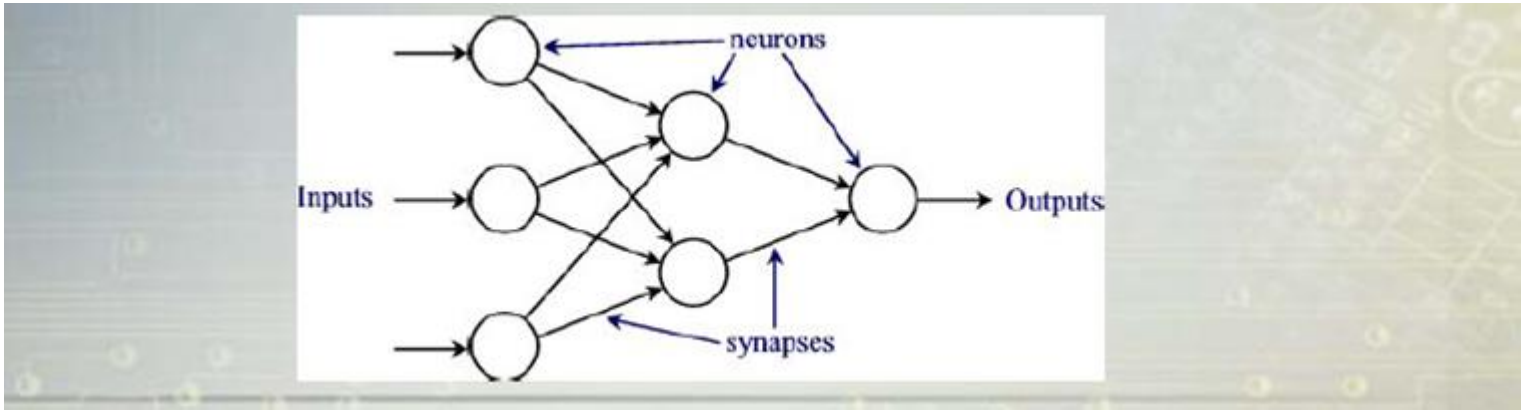
How does the brain work?

- Composed of many "neurons" that co-operate to perform the desired function



In brain a neuron has three principal components:

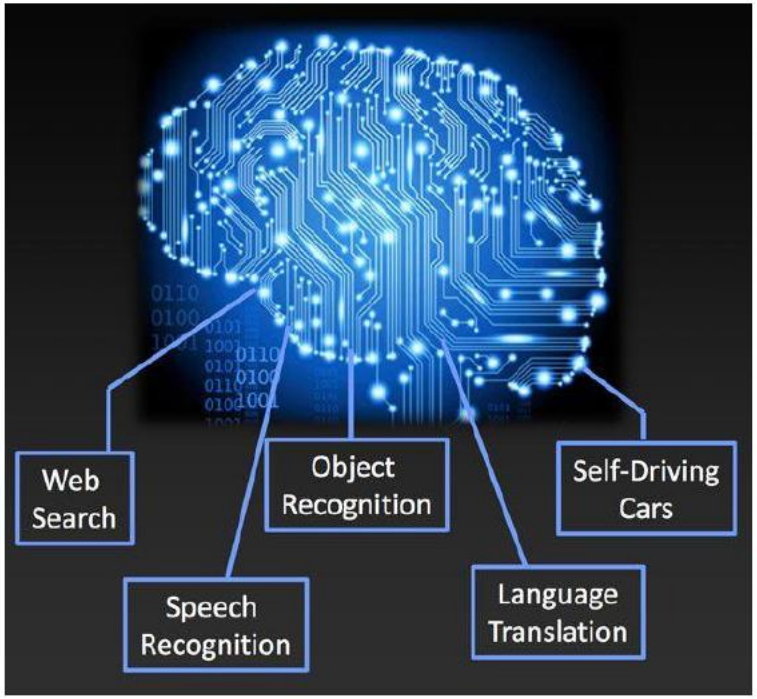
1. **Dendrites:-** that carry electrical signals into the cell body.
2. **Cell Body:-** effectively sums and thresholds these incoming signals.
3. **Axon:-** is a single long fiber that carries the signal from the cell body out to other neurons.
4. The point of contact between an axon of one cell and a dendrite of another cell is called a '**synapse'**



Background: ANN Vs Brain

ANN	Brain
It is simple (few neuron in connection)	It is complex (10^{11} Neurons and 10^{15} connections)
It is dedicated for specific purpose	It is generalized for all purpose
Response time is fast (it may be in Nanosecond)	Response time is slow (it may be in millisecond)
Design is regular	Design is arbitrary
Activities are synchronous	Activities are asynchronous

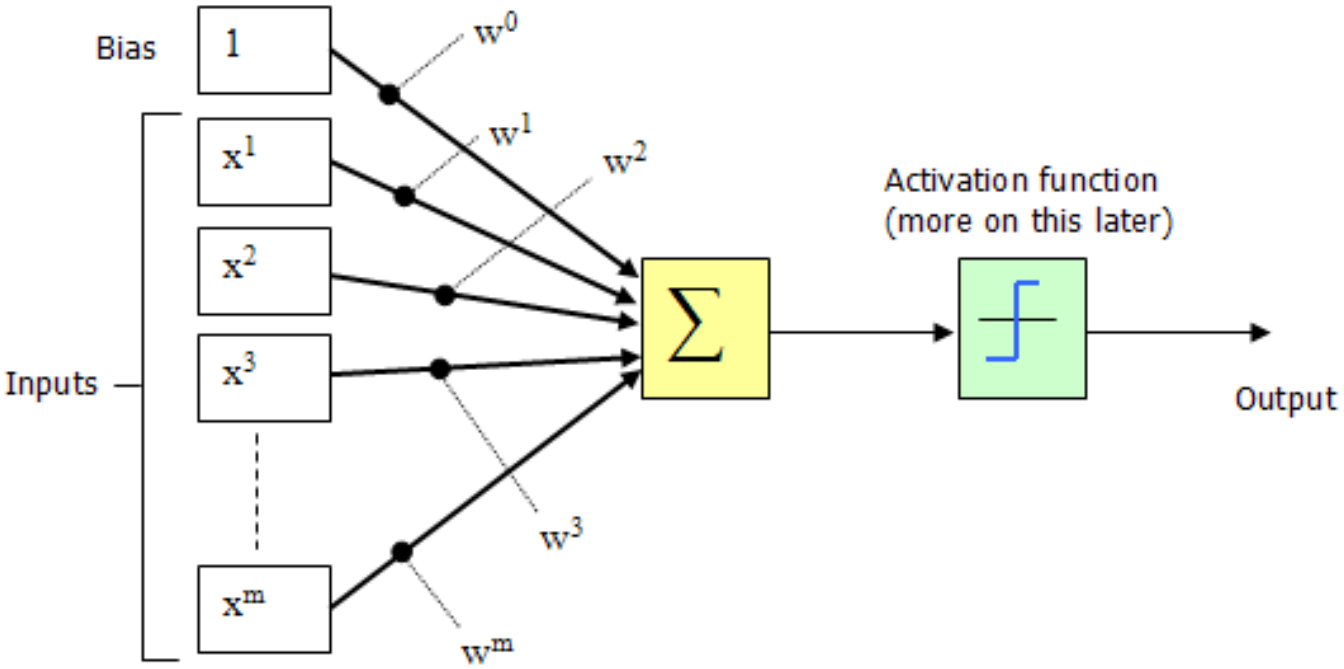
What is a neural network?



- A neural network is a "connectionist" computational system. The computational systems we write are procedural; a program starts at the first line of code, executes it, and goes on to the next, following instructions in a linear fashion. A true neural network does not follow a linear path. Rather, information is processed collectively, in parallel throughout a network of nodes (the nodes, in this case, being neurons).

- One of the key elements of a neural network is its ability to *learn*. A neural network is not just a complex system, but a complex ***adaptive*** system, meaning it can change its internal structure based on the information flowing through it. Typically, this is achieved through the adjusting of *weights.*

Perceptron



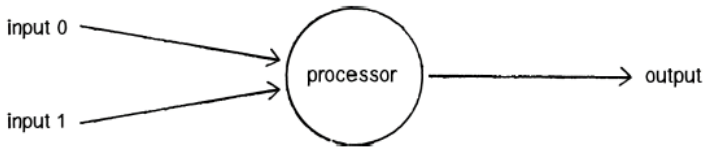
Perceptrons can only model linearly separable functions.

We need to use multi-layer perceptron to tackle non-linear problems.

Perceptron

A Perceptron is a mathematical model of a biological neuron

1. Each input gets scaled up or down by multiplying with weights
2. All signals are summed up
3. Activation



Invented in 1957 by Frank Rosenblatt at the Cornell Aeronautical Laboratory, a perceptron is the simplest neural network possible: a computational model of a single neuron. A perceptron consists of one or more inputs, a processor, and a single output.

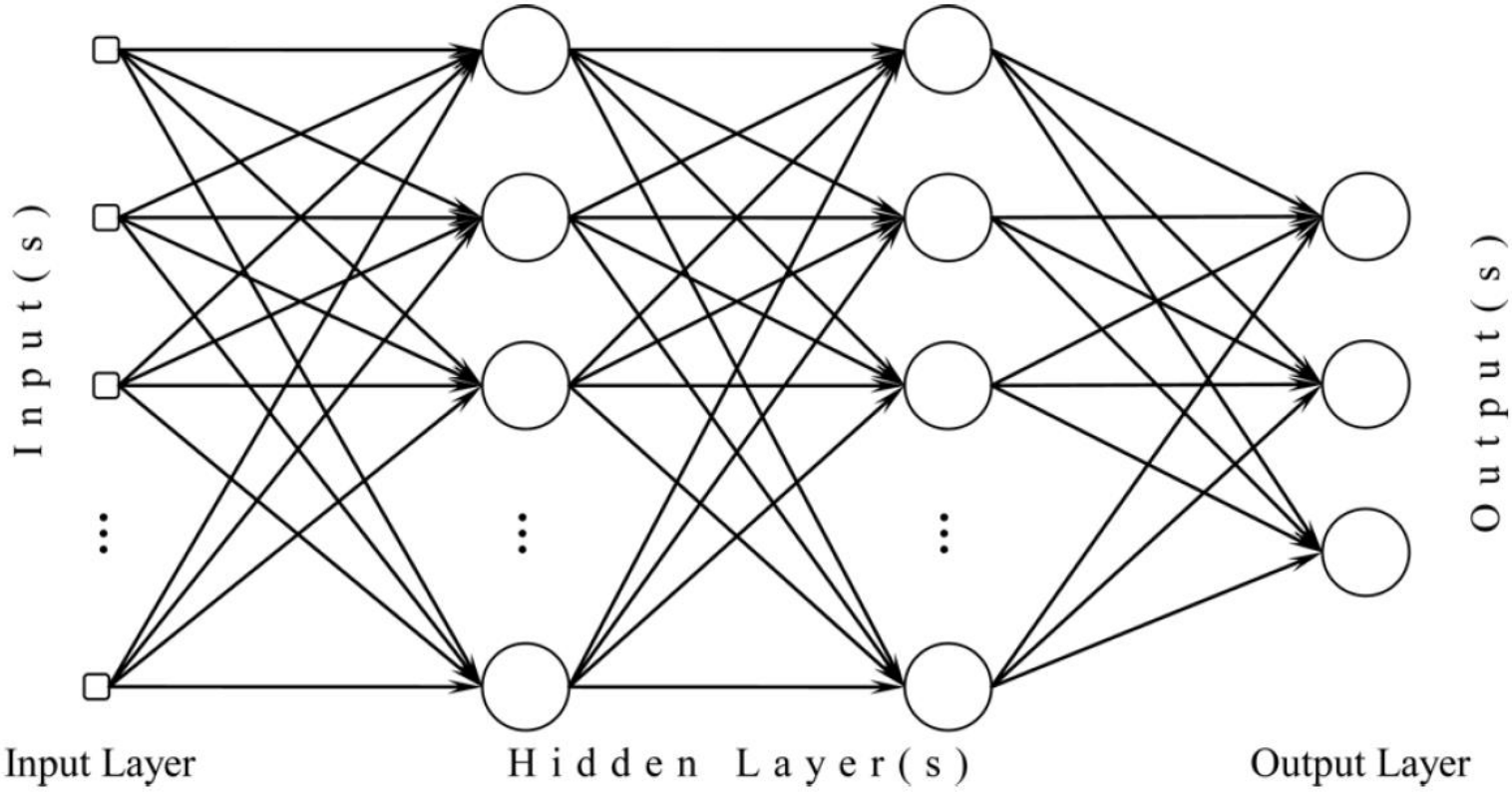
A perceptron follows the "feed-forward" model, meaning inputs are sent into the neuron, are processed, and result in an output. In the diagram above, this means the network (one neuron) reads from left to right: inputs come in, output goes out.

Activation Functions

Activation functions also known as transfer function is used to map input nodes to output nodes in certain fashion.

- There are many activation functions
- Sigmoid, tanh , relu, softmax are the most

Multi Layer Perceptron



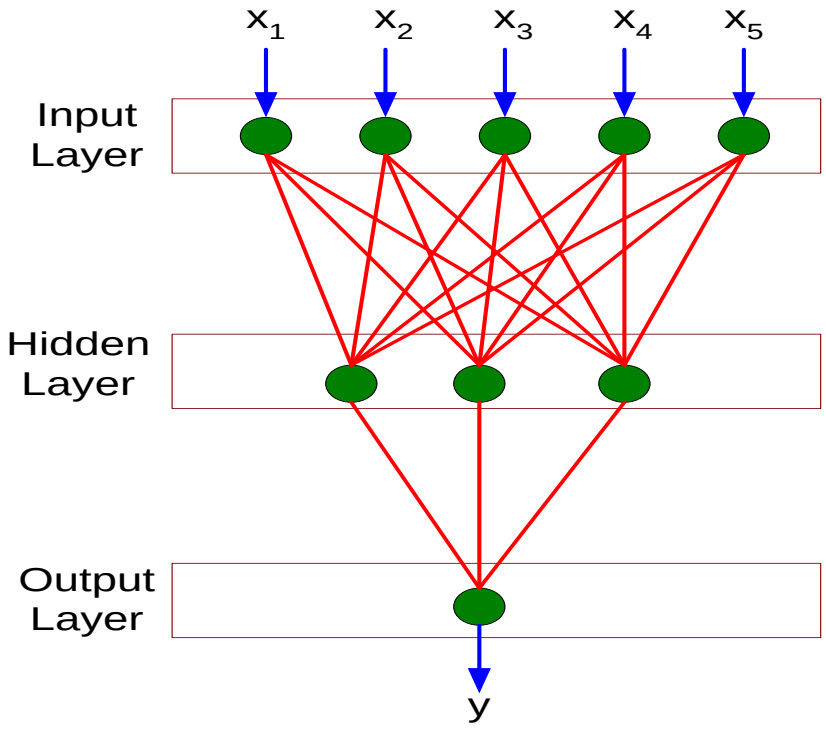
Multi Layer Perceptron

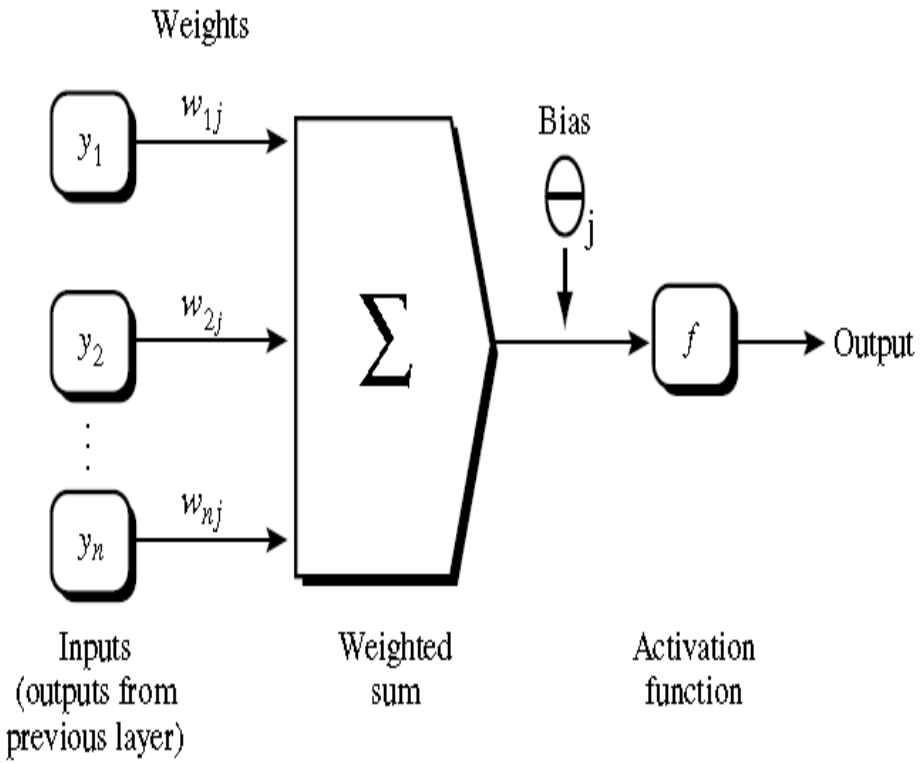
Input Layer
- Visible layer or the left most layer in the network
- Number of nodes in input layer is equal to number of variables in input data
- No activation function

Hidden Layer
- Layers between input and output layers
- Adds non linearity to the network
- Activation/Squashing function

Output Layer
- Right most layer in the network
- Fires the output
- Activation function

General Structure of ANN

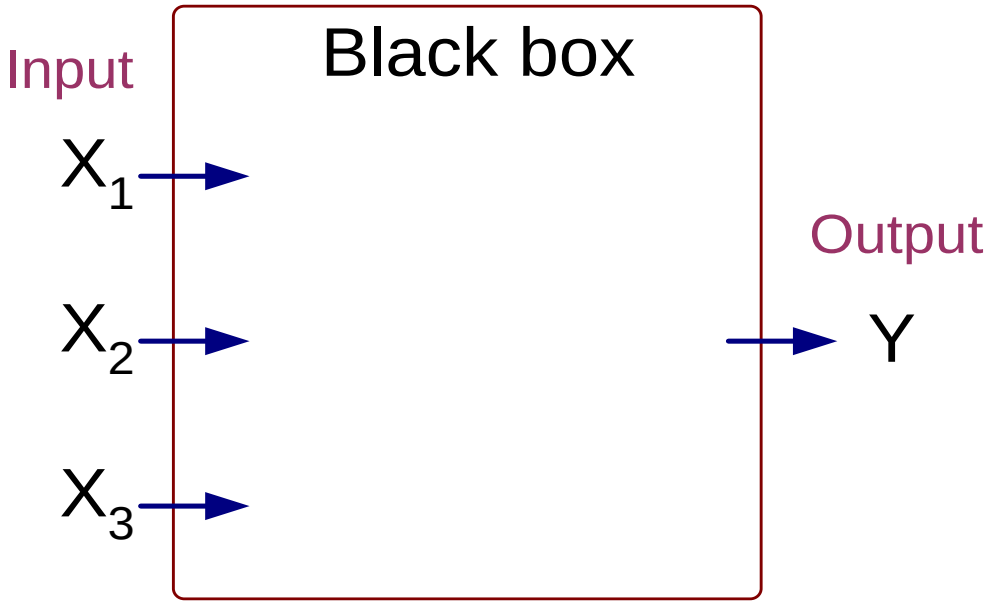




θ_j is the bias of the unit. The bias acts as a threshold, which is used to adjust the output along with the weighted sum of the inputs to the neuron. Therefore bias is a constant which helps the model in a way that it can fit best for the given data..

ANN

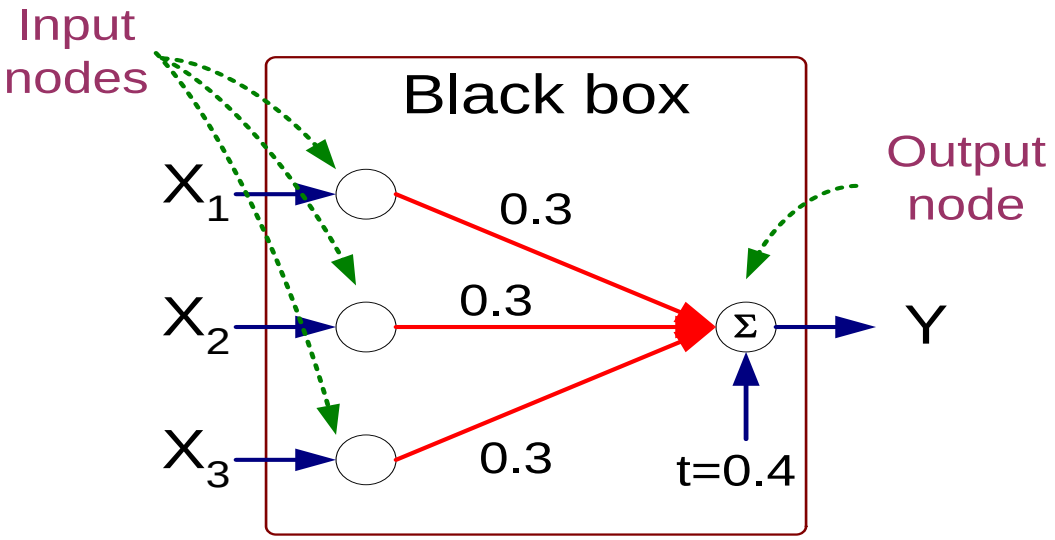
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

ANN

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

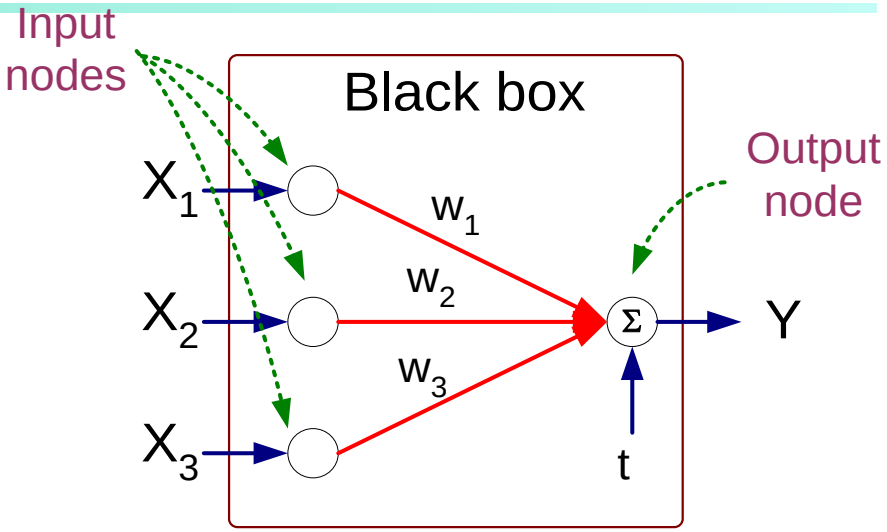


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Artificial Neural Networks

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold t



Perceptron Model

$$Y = I(\sum_i w_i X_i - t)$$

Given the net input *Ij to unit j, then Oj, the output of unit j, is computed as,*

$$O_j = \frac{1}{1 + e^{-I_j}}.$$

This function is also referred to as a *squashing function, because it maps a large input* domain onto the smaller range of 0 to 1.

Where Do The Weights Come From?

- The weights in a neural network are the most important factor in determining its function
- Training is the act of presenting the network with some sample data and modifying the weights to better approximate the desired function
- There are two main types of training
  - Supervised Training
    - Supplies the neural network with inputs and the desired outputs
    - Response of the network to the inputs is measured
      - The weights are modified to reduce the difference between the actual and desired outputs

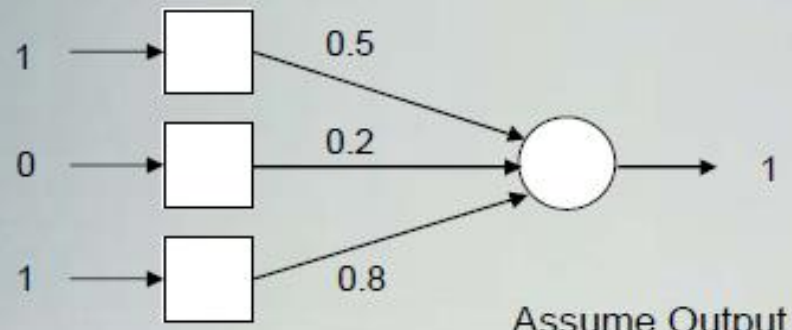
Where Do The Weights Come From?

- Unsupervised Training
  - Only supplies inputs
  - The neural network adjusts its own weights so that similar inputs cause similar outputs
    - The network identifies the patterns and differences in the inputs without any external assistance

How Do Perceptrons Learn?

- Uses supervised training
- If the output is not correct, the weights are adjusted according to the formula:
  - $W_{new} = W_{old} + \alpha(\text{desired} - \text{output})*\text{input}$

α is the learning rate



$1 * 0.5 + 0 * 0.2 + 1 * 0.8 = 1.3$

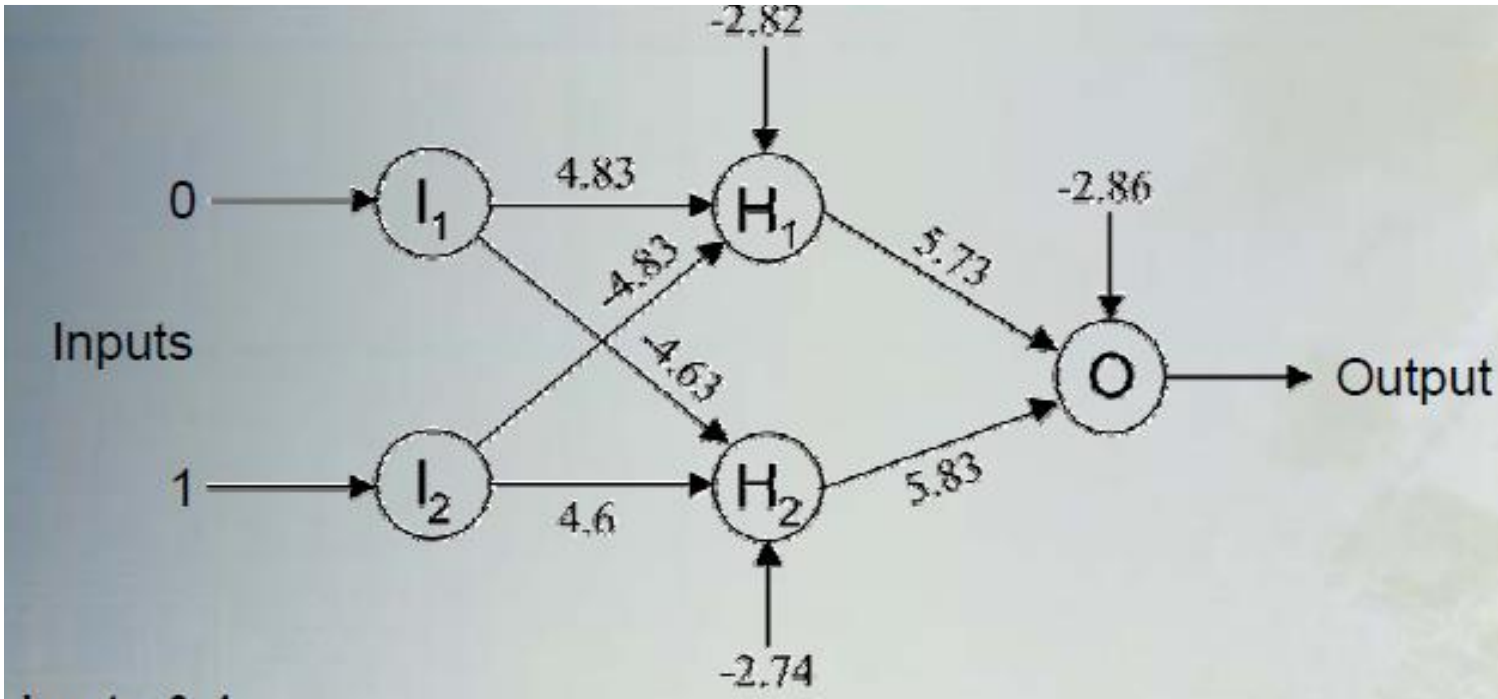
Assuming Output Threshold = 1.2

$1.3 > 1.2$

Assume Output was supposed to be 0
→ update the weights

Assume $\alpha = 1$

$W_{1new} = 0.5 + 1*(0-1)*1 = -0.5$
$W_{2new} = 0.2 + 1*(0-1)*0 = 0.2$
$W_{3new} = 0.8 + 1*(0-1)*1 = -0.2$



Inputs: 0, 1

H_1: Net = 0(4.83) + 1(-4.83) – 2.82 = -7.65
Output = $1 / (1 + e^{7.65}) = 4.758 \times 10^{-4}$

H_2: Net = 0(-4.63) + 1(4.6) – 2.74 = 1.86
Output = $1 / (1 + e^{-1.86}) = 0.8652$

O: Net = $4.758 \times 10^{-4}(5.73) + 0.8652(5.83) – 2.86 = 2.187$
Output = $1 / (1 + e^{-2.187}) = 0.8991 \equiv$ "1"

Learning Algorithms:
Back propagation for classification

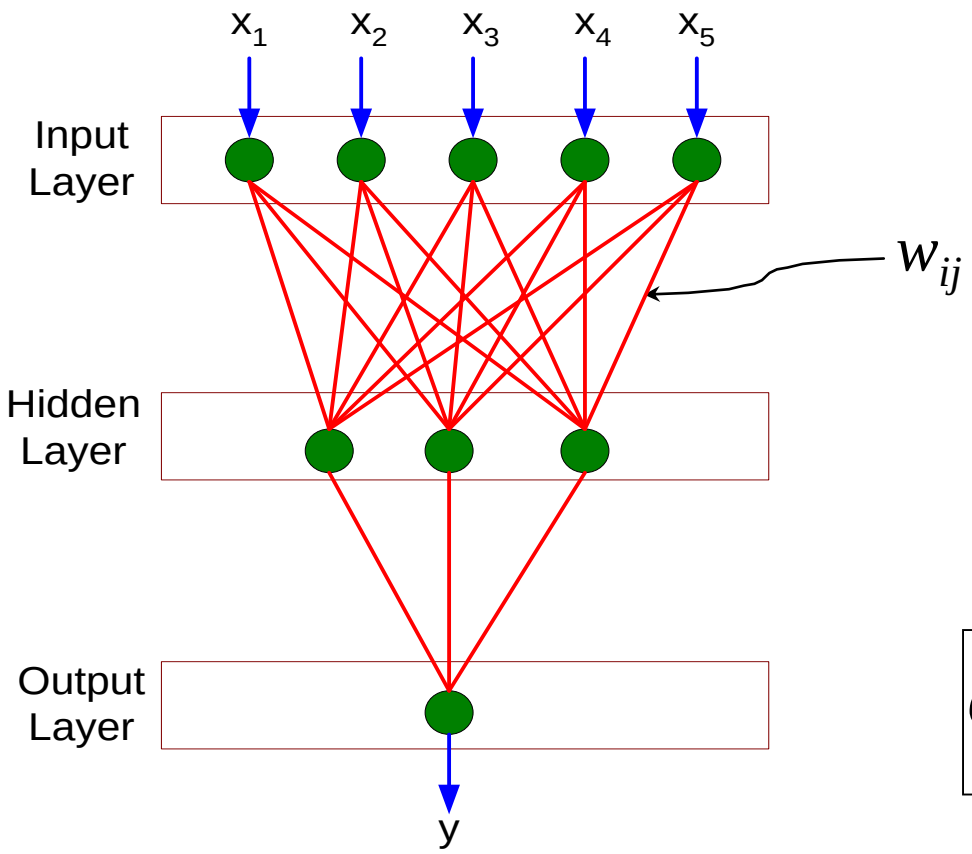
What is backpropagation

- Backpropagation is a neural network learning algorithm.
- There are many different kinds of neural networks and neural network algorithms.
- The most popular neural network algorithm is *backpropagation, which gained repute* in the 1980s.
- Multilayer feed-forward networks is type of neural network on which the backpropagation algorithm performs.
- Backpropagation learns for a set of weights that fits the training data so as to minimize the mean squared distance between the network's class prediction and the known target value of the tuples.

Major Steps for Back Propagation Network

- Constructing a network
  - input data representation
  - selection of number of layers, number of nodes in each layer.
- Training the network using training data
- Pruning the network
- Interpret the results

A Multi-Layer Feed-Forward Neural Network



$$I_j = \sum_i w_{ij} O_i + \theta_j$$

$$O_j = \frac{1}{1 + e^{-I_j}}$$

How A Multi-Layer Neural Network Works?

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which gives out the network's prediction
- The network is **feed-forward** in that none of the weights cycles back to an input unit or to an output unit of a previous layer

Defining a Network Topology

- First decide the **network topology:** # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalizing the input values for each attribute measured in the training tuples to [0.0—1.0]
- One **input** unit per domain value
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a *different network topology* or a *different set of initial weights*

Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the "**backwards**" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "**backpropagation**"
- Steps
  - Initialize weights and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

Backpropagation: Algorithm

Algorithm: Backpropagation. Neural network learning for classification or prediction, using the backpropagation algorithm.

Input:

- D, a data set consisting of the training tuples and their associated target values;
- l, the learning rate;
- *network*, a multilayer feed-forward network.

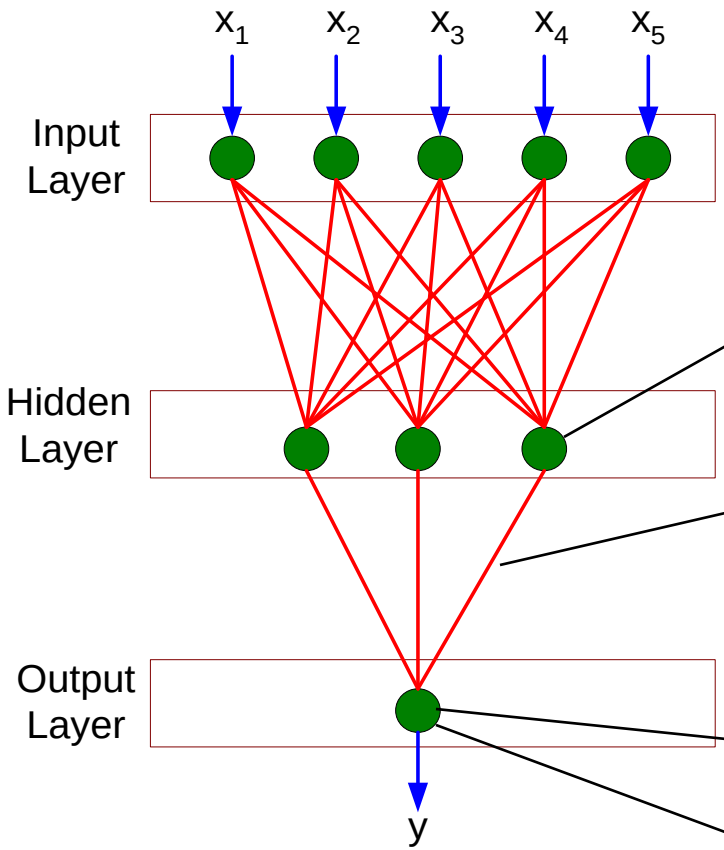
Output: A trained neural network.

Method:

(1) Initialize all weights and biases in *network*;
(2) **while** terminating condition is not satisfied {
(3) **for** each training tuple $\boldsymbol{X}$ in D {
(4) // Propagate the inputs forward:
(5) **for** each input layer unit j {
(6) $O_j = I_j$; // output of an input unit is its actual input value
(7) **for** each hidden or output layer unit j {
(8) $I_j = \sum_i w_{ij} O_i + \theta_j$; //compute the net input of unit j with respect to the previous layer, i
(9) $O_j = \frac{1}{1+e^{-I_j}}$; } // compute the output of each unit j
(10) // Backpropagate the errors:
(11) **for** each unit j in the output layer
(12) $Err_j = O_j(1 - O_j)(T_j - O_j)$; // compute the error
(13) **for** each unit j in the hidden layers, from the last to the first hidden layer
(14) $Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$; // compute the error with respect to the next higher layer, k
(15) **for** each weight w_{ij} in *network* {
(16) $\Delta w_{ij} = (l) Err_j O_i$; // weight increment
(17) $w_{ij} = w_{ij} + \Delta w_{ij}$; } // weight update
(18) **for** each bias θ_j in *network* {
(19) $\Delta \theta_j = (l) Err_j$; // bias increment
(20) $\theta_j = \theta_j + \Delta \theta_j$; } // bias update
(21) } }

Backpropagation algorithm.

Backpropagation



$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

$$w_{ij} = w_{ij} + (l)Err_j O_i$$

$$\theta_j = \theta_j + (l)Err_j$$

$$Err_j = O_j(1-O_j)(T_j - O_j)$$

Generated value

Correct value

Example - Sample calculations for learning by the backpropagation algorithm

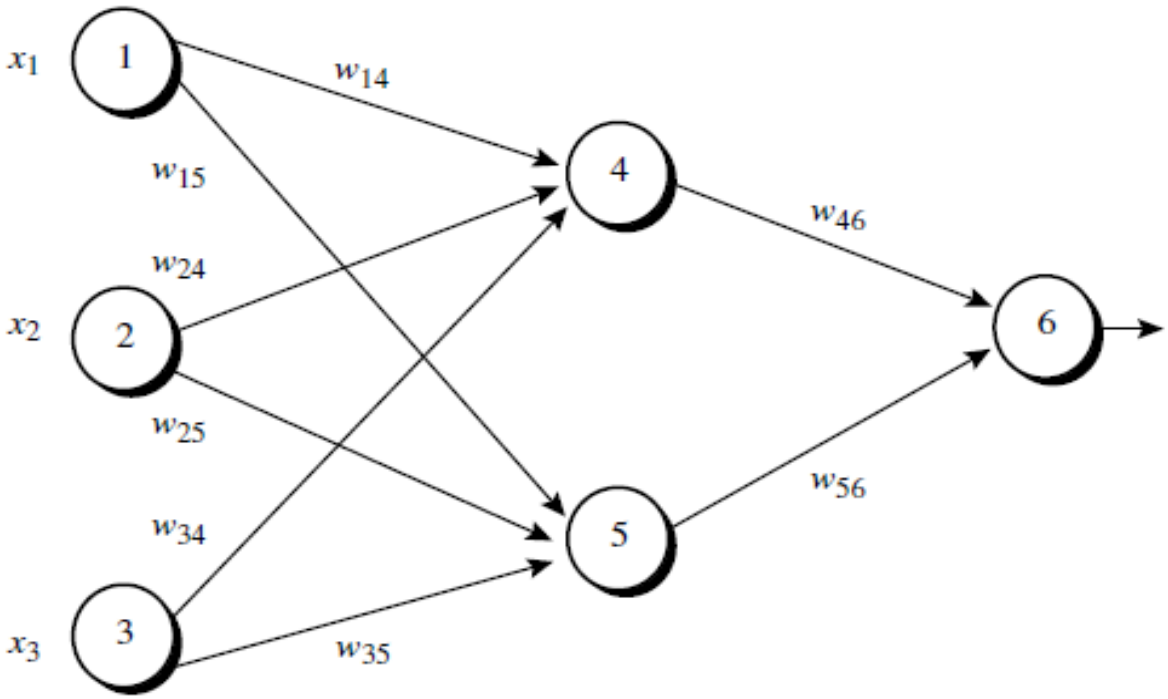
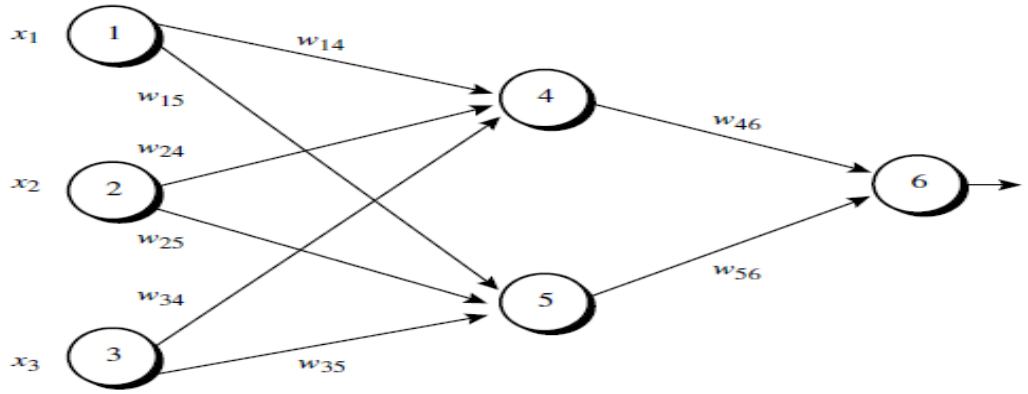


Figure shows : a multilayer feed-forward neural network.
Let the learning rate be 0.9.
The first training tuple, ***X = (1, 0, 1), whose class label is 1.***



Initial input, weight, and bias values.

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1

The net input and output calculations.

Unit j	*Net input,* I_j	*Output,* O_j
4	$0.2+0-0.5-0.4=-0.7$	$1/(1+e^{0.7})=0.332$
5	$-0.3+0+0.2+0.2=0.1$	$1/(1+e^{-0.1})=0.525$
6	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105})=0.474$

Calculation of the error at each node.

Unit j	Err_j
6	$(0.474)(1-0.474)(1-0.474)=0.1311$
5	$(0.525)(1-0.525)(0.1311)(-0.2)=-0.0065$
4	$(0.332)(1-0.332)(0.1311)(-0.3)=-0.0087$

for each unit j in the output layer
    $Err_j = O_j(1-O_j)(T_j-O_j)$; // compute the error
for each unit j in the hidden layers, from the last to the first hidden layer
    $Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$; // compute the error with respect to the next higher layer, k

```
for each weight w_ij in network {
    Δw_ij = (l)Err_j O_i; // weight incremen
    w_ij = w_ij + Δw_ij; } // weight update
for each bias θ_j in network {
    Δθ_j = (l)Err_j; // bias increment
    θ_j = θ_j + Δθ_j; } // bias update
```

Calculations for weight and bias updating.

Weight or bias	*New value*
w_{46}	$-0.3 + (0.9)(0.1311)(0.332) = -0.261$
w_{56}	$-0.2 + (0.9)(0.1311)(0.525) = -0.138$
w_{14}	$0.2 + (0.9)(-0.0087)(1) = 0.192$
w_{15}	$-0.3 + (0.9)(-0.0065)(1) = -0.306$
w_{24}	$0.4 + (0.9)(-0.0087)(0) = 0.4$
w_{25}	$0.1 + (0.9)(-0.0065)(0) = 0.1$
w_{34}	$-0.5 + (0.9)(-0.0087)(1) = -0.508$
w_{35}	$0.2 + (0.9)(-0.0065)(1) = 0.194$
θ_6	$0.1 + (0.9)(0.1311) = 0.218$
θ_5	$0.2 + (0.9)(-0.0065) = 0.194$
θ_4	$-0.4 + (0.9)(-0.0087) = -0.408$

Neural Network as a Classifier

- Weakness
  - Long training time
  - Require a number of parameters, e.g., the network topology or ``structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of ``hidden units" in the network

- Strength
  - High tolerance to noisy data as well as their ability to classify patterns on which they have not been trained.
  - They are well-suited for continuous-valued inputs *and outputs*, unlike most decision tree algorithms.
  - They have been successful on a wide array of real-world data, including handwritten character recognition, pathology and laboratory medicine, and training a computer to pronounce English text.
  - Neural network algorithms are inherently parallel; parallelization techniques can be used to speed up the computation process.

These above factors contribute toward the usefulness of neural networks for classification and prediction in machine learning.

K Nearest Neighbor

- Understand fundamentals of KNN
- Basis of KNN: Judge someone by the neighborhood they belong
- Deep dive into working of a KNN algorithm internally
- **Case Study**: Identify whether a website is malicious or not

Lazy vs. Eager Learning

- The classification methods —decision tree induction, Bayesian classification, classification by backpropagation, support vector machines—are all examples of *eager learners*.

Lazy vs. Eager Learning

- Lazy vs. eager learning
  - **Lazy learning (instance-based learning):** Simply stores training data (or only minor processing) and waits until it is given a test tuple.
  - **Eager learning :** Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify.

    We can think of the learned model as being ready and eager to classify previously unseen tuples.

- Lazy: less time in training but more time in predicting so lazy learners can be computationally expensive.

Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified.
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference.

The *k*-Nearest Neighbor Algorithm

- The *k*-nearest-neighbor method was first described in the early 1950s.
- It has since been widely used in the area of pattern recognition.

- Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuple with training tuples that are similar to it.

- The training tuples are described by *n* attributes.

- Each tuple represents a point in an *n*-dimensional space.
- In this way, all of the training tuples are stored in an *n*-dimensional pattern space.
- When given an unknown tuple, a ***k***-nearest-neighbor classifier searches the pattern space for the *k* training tuples that are closest to the unknown tuple. These *k* training tuples are the *k* "nearest neighbors" of the unknown tuple.

The *k*-Nearest Neighbor Algorithm

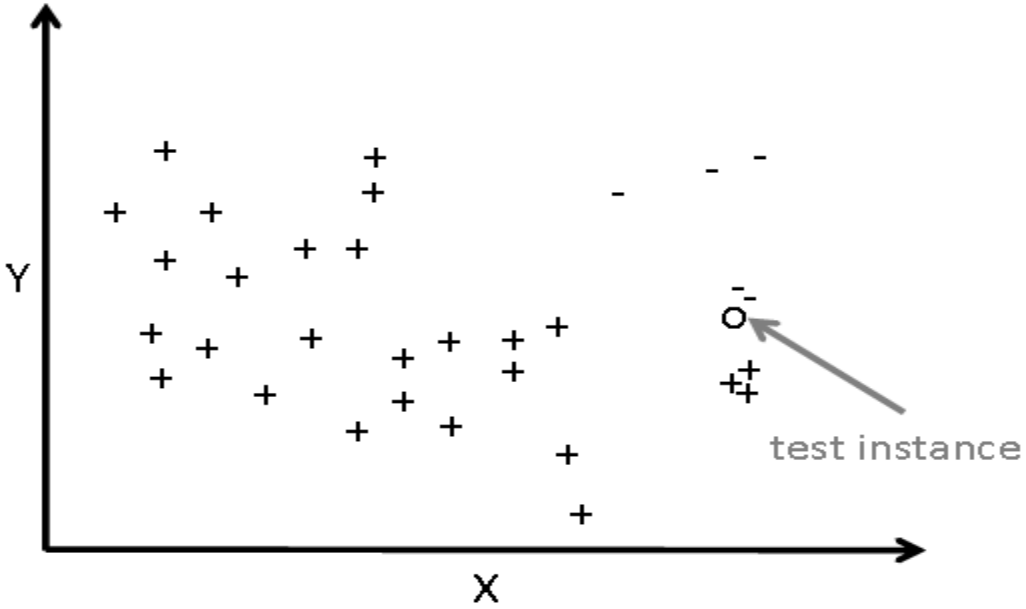
- "Closeness" is defined in terms of a distance metric, such as Euclidean distance. The Euclidean distance between two points or tuples, say, $\boldsymbol{X}1 = (x11, x12, \ldots\ldots, x1n)$ and $\boldsymbol{X}2 = (x21, x22, \ldots\ldots, x2n)$, is

$$dist(\boldsymbol{X_1}, \boldsymbol{X_2}) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}.$$

- For *k*-nearest-neighbor classification, the unknown tuple is assigned the most common class among its *k* nearest neighbors. When $k = 1$, the unknown tuple is assigned the class of the training tuple that is closest to it in pattern space.
- Nearest neighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown tuple.

Example-
Instance-Based Classification

- A KNN classifier assigns a test instance the majority class associated with its K nearest training instances. Distance between instances is measured using the Euclidean distance.
- Suppose we have the following training set of positive (+) and negative (-) instances and a single test instance (o).
- All instances are projected onto a vector space of two real-valued features: X and Y.



Contd…

(a) What would be the class assigned to this test instance for K=1 .

KNN assigns a test instance the target class associated with the majority of the test instance's K nearest neighbors. For K=1, this test instance would be predicted negative because it's single nearest neighbor is negative.

(b) What would be the class assigned to this test instance for K=3.

KNN assigns a test instance the target class associated with the majority of the test instance's K nearest neighbors. For K=3, this test instance would be predicted negative. Out of its three nearest neighbors, two are negative and one is positive.

Advantages of KNN

The **simplest** yet one of the most **commonly** used Machine Learning algorithms.

Easy to implement

Advantages of KNN

- KNN has no model– No complex function like other ML algorithms such as **Linear Regression, Logistic Regression , Decision Tree, Naïve Bayes etc.**
- It just **stores data,** so **no learning** is required.

Advantages of KNN

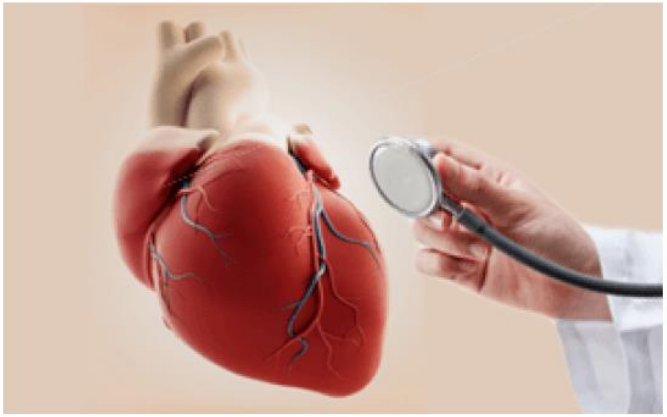
Training **not required** hence all work is done during prediction.

- No **prior knowledge** or any knowledge about distribution of data required
- Defers data processing until receives request to classify unlabeled data – **Lazy Learner**

Example of application of KNN

KNN is simple but powerful algorithm & many times outperforms other algorithms.

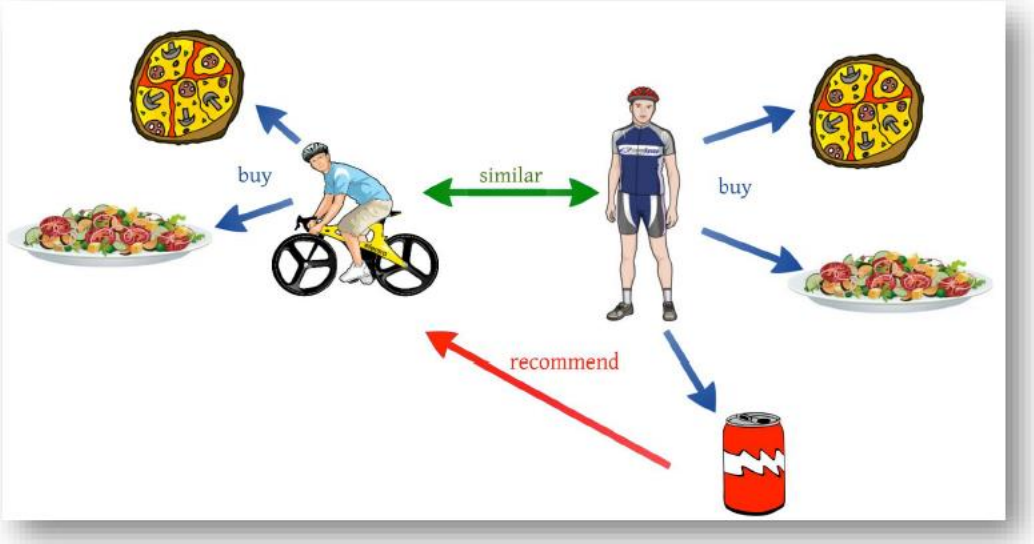
Is my heart functioning well?



- In medical research lots of data are generated.
- KNN is capable of handling huge amount of data provided we've sufficient resources,
- And it doesn't get affected by outliers.
- So based on similarity among cases of diagnosis it can **predict** diseases.

Example of application of KNN

- Since it works on basis of similarity between instances so it has wide application in Recommender System.
- Similarity between users' ratings, products' ratings can efficiently be recognized by KNN algorithm.



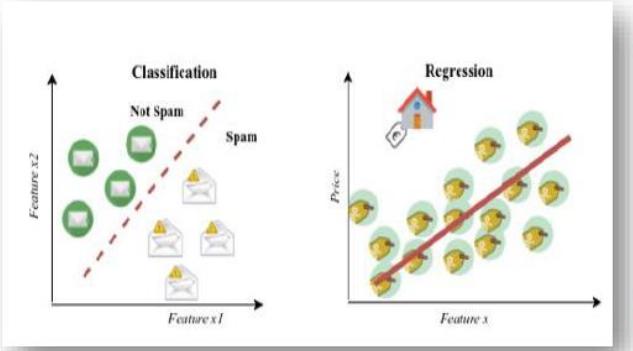
N.B: We'll cover application of KNN algorithm in Recommender system.

KNN(K Nearest Neighbor) in a nutshell

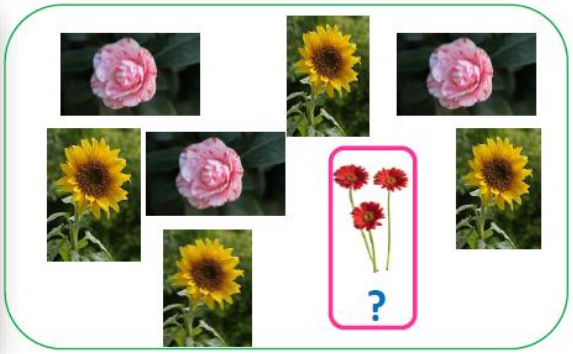
A form of **supervised learning**

Data processed by KNN has input features & output feature.

Used in both **Classification & Regression**



Uses **nearest majority** of neighbors to predict data points

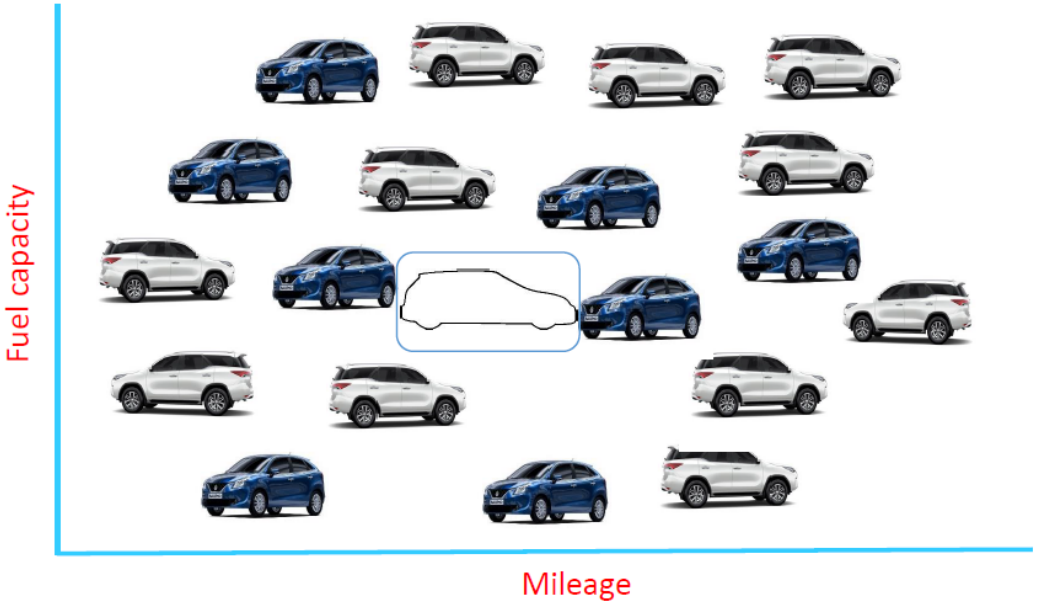


How does KNN work?

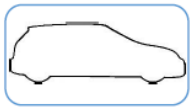
- **Step 1:** Build a neighborhood with **K members** (K= 1 to any number, Discussed at end)
- **Step 2:** Find distance from **query point(**Test data**)** to each point of neighborhood.
- **Step 3:** Assign query point to that class having **maximum members around (Majority voting)**.

Let's take a simple example of Classification

- Classification is done in KNN based on **majority of neighbors**.
- Where K denotes number of **neighbors.**

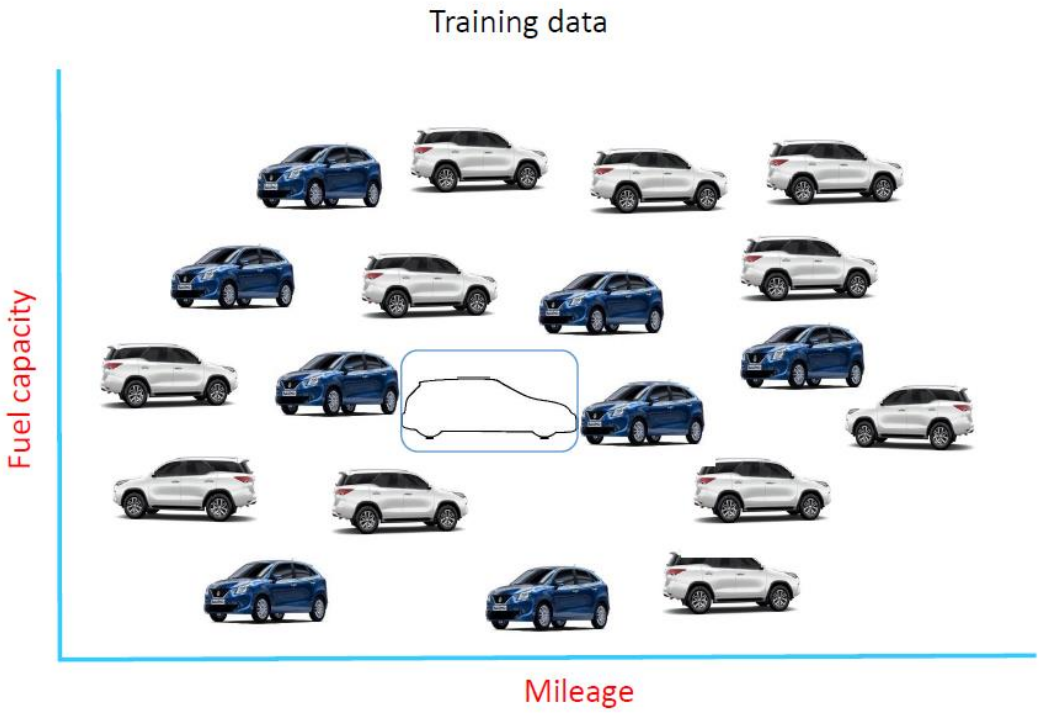


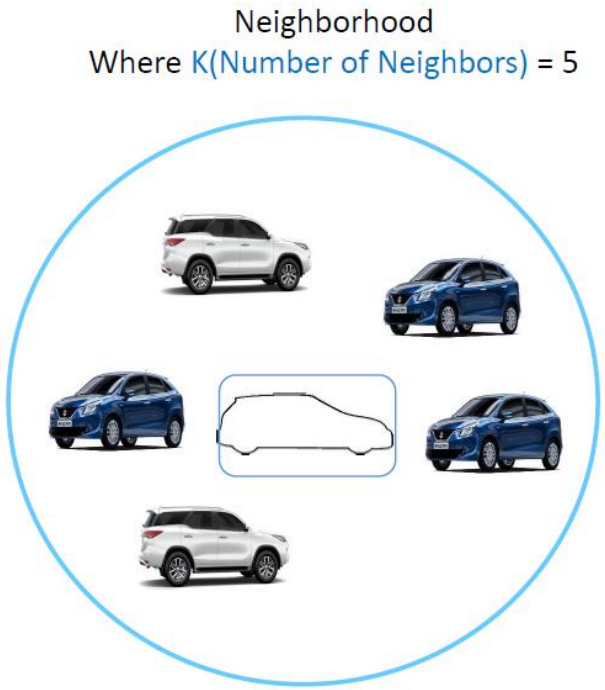
Query Point



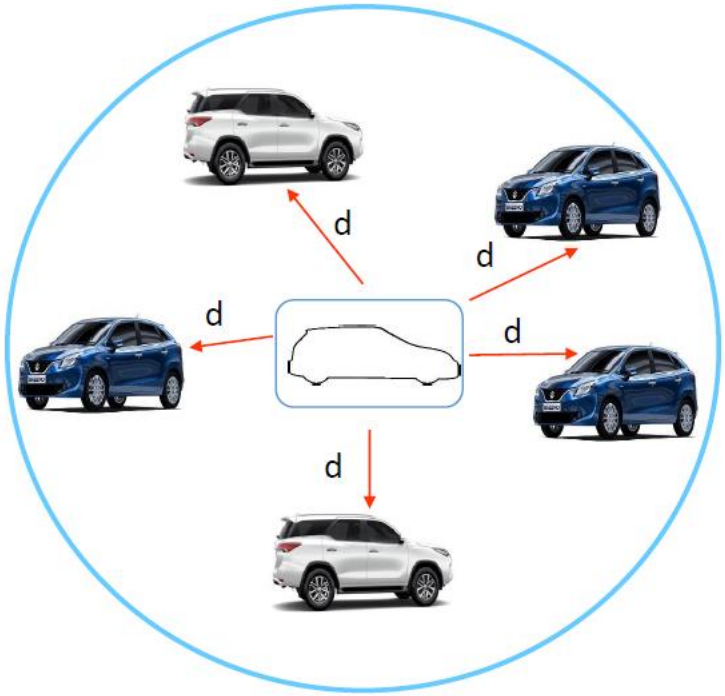
Predict class of query point.

Step 1: Build neighborhood





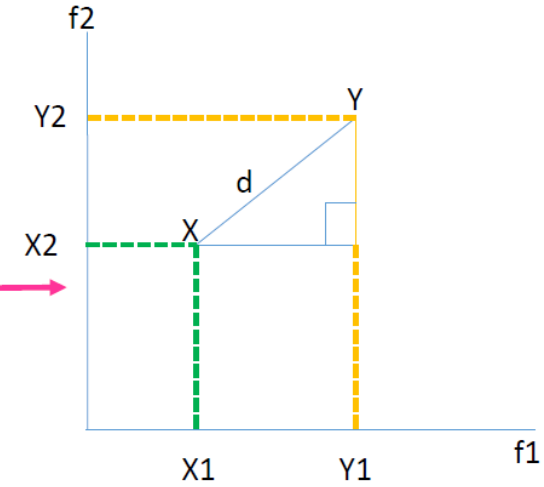
Step 2: Find distance from query point to each point in neighborhood



FYI: Distance measures for continuous data

- Distance is measured from **query point to each data point** in neighborhood.
- There are a number of distance measures for continuous data such as **Euclidean , Manhattan , Minkowski**—Most common being Euclidean Distance.

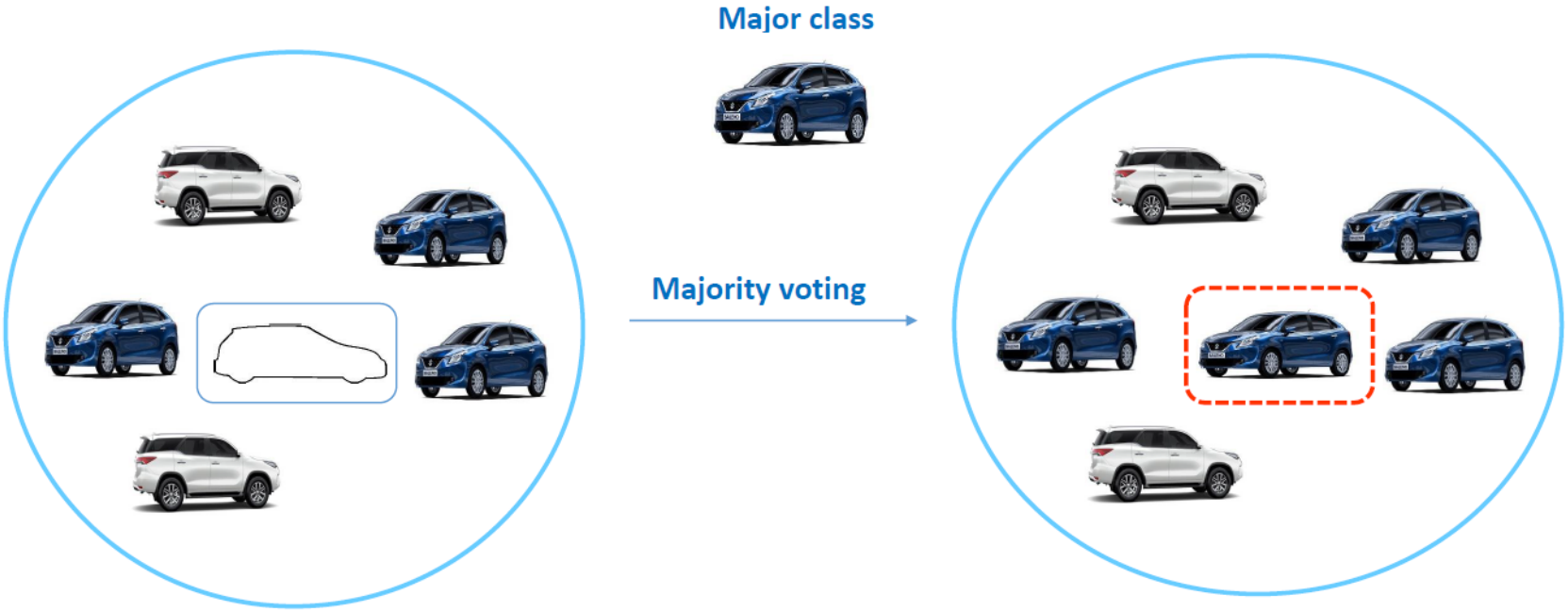
Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$



- Euclidean distance(d) measures shortest dist. between two points(X & Y).
- **X(X1,X2) & Y(Y1,Y2)**
- f1= X1,Y1 & f2= X2,Y2

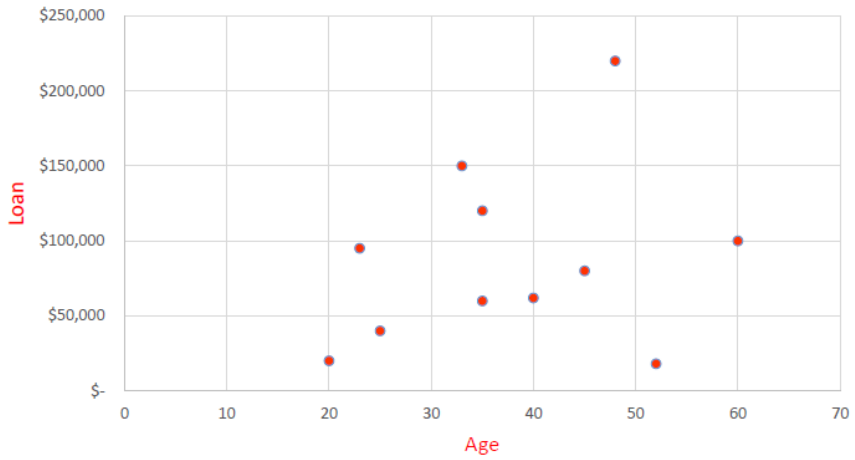
$d = \sqrt{(Y1 - X1)^2 + (Y2 - X2)^2}$

Step 3: Assign to class



Classification with KNN: Loan default data

Age	Loan	Default
25	$ 40,000	N
35	$ 60,000	N
45	$ 80,000	N
20	$ 20,000	N
33	$ 150,000	Y
52	$ 18,000	N
23	$ 95,000	Y
40	$ 62,000	Y
60	$ 100,000	Y
48	$ 220,000	Y
35	$ 120,000	N
48	$ 142,000	?



We've to **predict** whether this person will **default** or not?

Step 1: Build neighborhood

K=3, indicates we want a neighborhood having **3 neighbors of Query point**.

How to decide the value of K?

- As a starting point we've taken value of K arbitrarily
- Generally we take square root of total number of observations.
- We should take odd number as value of K i.e 3,5,7,9 & so on(Discussed later).

Classification with KNN: Build neighborhood

K=3, indicates we want a neighborhood having **3 neighbors of Query point**.

Age	Loan	Default
25	$ 40,000	N
35	$ 60,000	N
45	$ 80,000	N
20	$ 20,000	N
33	$ 150,000	Y
52	$ 18,000	N
23	$ 95,000	Y
40	$ 62,000	Y
60	$ 100,000	Y
48	$ 220,000	Y
35	$ 120,000	N
48	$ 142,000	?

Step 2: Measure distance from each data point

Using Euclidean distance we can measure distance from **each data point** to **Query point**.

Age	Loan	Default	Distance(d)
25	$ 40,000	N	102000
35	$ 60,000	N	82000
45	$ 80,000	N	62000
20	$ 20,000	N	122000
33	$ 150,000	Y	8000
52	$ 18,000	N	124000
23	$ 95,000	Y	47000
40	$ 62,000	Y	80000
60	$ 100,000	Y	42000
48	$ 220,000	Y	78000
35	$ 120,000	N	22000
48	$ 142,000	?	

Let's calculate Euclidean distance for one observation

$$d = \sqrt{(Y1 - X1)^2 + (Y2 - X2)^2}$$

$$= \sqrt{(48 - 33)^2 + (142{,}000 - 150{,}000)^2}$$

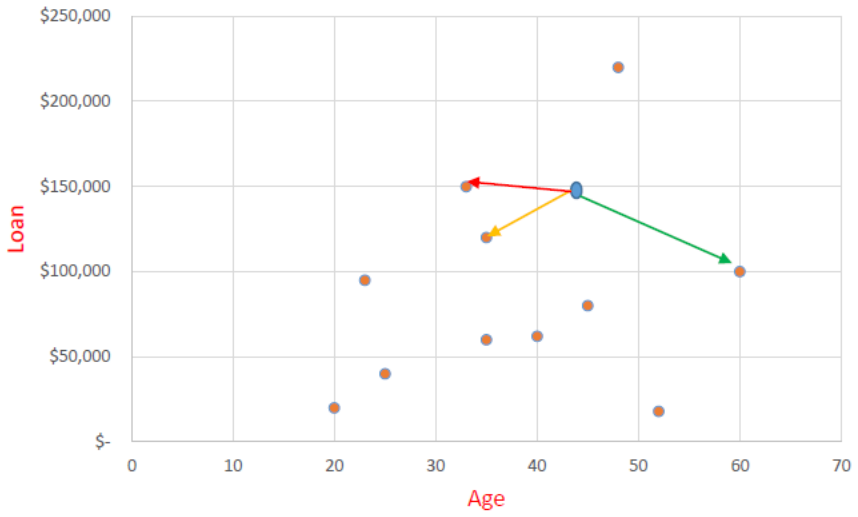
$$= \sqrt{225 + 64000000}$$

$$= 8000.01$$

We'll apply the same process to calculate **'d'** for all other observations.

Step 2: Graphical representation of distance

Age	Loan	Default	Distance(d)
25	$ 40,000	N	102000
35	$ 60,000	N	82000
45	$ 80,000	N	62000
20	$ 20,000	N	122000
33	$ 150,000	Y	8000
52	$ 18,000	N	124000
23	$ 95,000	Y	47000
40	$ 62,000	Y	80000
60	$ 100,000	Y	42000
48	$ 220,000	Y	78000
35	$ 120,000	N	22000
48	$ 142,000	?	



Step 3: Assign to class based on majority vote

Age	Loan	Default	Distance(d)	
25	$ 40,000	N	102000	
35	$ 60,000	N	82000	
45	$ 80,000	N	62000	
20	$ 20,000	N	122000	
33	$ 150,000	Y	8000	1st
52	$ 18,000	N	124000	
23	$ 95,000	Y	47000	
40	$ 62,000	Y	80000	
60	$ 100,000	Y	42000	3rd
48	$ 220,000	Y	78000	
35	$ 120,000	N	22000	2nd
48	$ 142,000	?		

- K=3, we will consider only 3 nearest neighbors
- Then assign Query point to majority neighbors.
- From given three nearest neighbors, **2 neighbors(Majority)** belong to Yes(**1st & 3rd**) class.
- So as per majority vote, we'll assign Query point to Yes(Default)

KNN for Regression: Let's work withLoan data set

Age	Loan	Income
25	$ 40,000	$ 4,700
35	$ 60,000	$ 5,000
45	$ 80,000	$ 5,200
20	$ 20,000	$ 4,300
33	$ 150,000	$ 7,000
52	$ 18,000	$ 4,000
23	$ 95,000	$ 4,900
40	$ 62,000	$ 4,200
60	$ 100,000	$ 5,150
48	$ 220,000	$ 6,000
35	$ 120,000	$ 5,400
48	$ 142,000	?

Regression in KNN works on basis of **mean** calculation of outputs of nearest neighbors.

Income + Income + ... Income

n (No. of neighbors)

We've to **predict** Income of the **Query point**.

Step 1: Define 'K'(number of neighbors)

We take K=3 i.e, 3 nearest neighbors will build the neighborhood.

Age	Loan	Income
25	$ 40,000	$ 4,700
35	$ 60,000	$ 5,000
45	$ 80,000	$ 5,200
20	$ 20,000	$ 4,300
33	$ 150,000	$ 7,000
52	$ 18,000	$ 4,000
23	$ 95,000	$ 4,900
40	$ 62,000	$ 4,200
60	$ 100,000	$ 5,150
48	$ 220,000	$ 6,000
35	$ 120,000	$ 5,400
48	$ 142,000	?

Step 2: Measure distance from each data point

Age	Loan	Income	Distance
25	$ 40,000	$ 4,700	10200
35	$ 60,000	$ 5,000	82000
45	$ 80,000	$ 5,200	62000
20	$ 20,000	$ 4,300	122000
33	$ 150,000	$ 7,000	8000
52	$ 18,000	$ 4,000	124000
23	$ 95,000	$ 4,900	47000
40	$ 62,000	$ 4,200	80000
60	$ 100,000	$ 5,150	42000
48	$ 220,000	$ 6,000	78000
35	$ 120,000	$ 5,400	22000
Y1	Y2		
48	$ 142,000	?	

We'll calculate Euclidean distance for one observation & we'll repeat the same process for all other observations.

$$d = \sqrt{(Y1 - X1)^2 + (Y2 - X2)^2}$$

$$= \sqrt{(48 - 35)^2 + (142{,}000 - 120{,}000)^2}$$

$$= \sqrt{169 + 484000000}$$

$$= 22000.01$$

Regression with KNN: Predict income of Query point

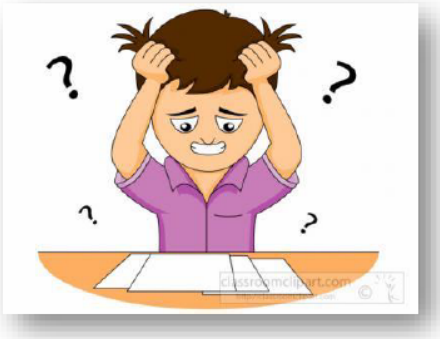
Age	Loan	Income	Distance	
25	$ 40,000	$ 4,700	10200	
35	$ 60,000	$ 5,000	82000	
45	$ 80,000	$ 5,200	62000	
20	$ 20,000	$ 4,300	122000	
33	$ 150,000	$ 7,000	8000	1st Nearest neighbor
52	$ 18,000	$ 4,000	124000	
23	$ 95,000	$ 4,900	47000	
40	$ 62,000	$ 4,200	80000	
60	$ 100,000	$ 5,150	42000	3rd Nearest neighbor
48	$ 220,000	$ 6,000	78000	
35	$ 120,000	$ 5,400	22000	2nd Nearest neighbor
Y1	Y2			
48	$ 142,000	?		

Average income of nearest 3 neighbors to predict Income of Query point.

$$\text{Income(Query Point)} = \frac{7000+5400+5150}{3} = 5850$$

What should be the value of K?

Value of K shouldn't be even because it ends up in tie.

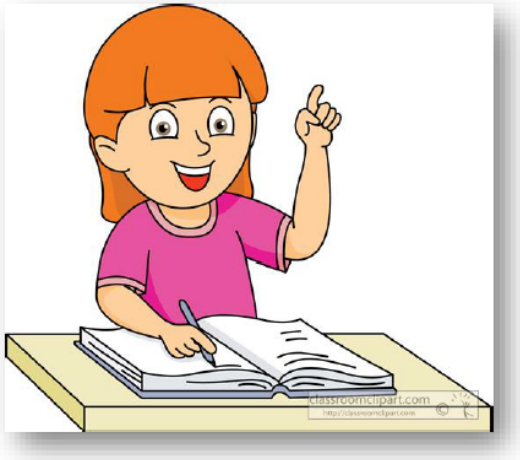


Which class should I **assign** Query point to?

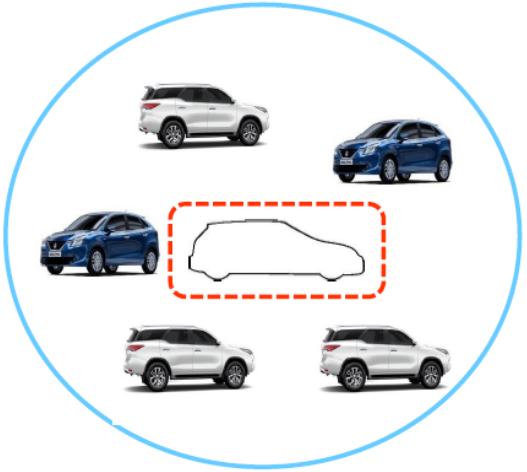
What should be the value of K?

Value of K should be odd, for ex, 3,5,7 and so on.

Because one of given classes will be more or less than others.



Query point belongs to SUV because SUV is major class



Case Study: Identify whether a website is malicious or not

Phishing is an illegitimate practice of fraudulently obtaining confidential information through redirecting to malicious websites, where users are induced to reveal their credentials.

If we develop a predictive model with past data having pertinent attributes, can help us NOT to visit any malicious websites.

Identify whether a website is malicious or not: Data Attributes

SFH's type is nominal, range is ('1', '-1', '0')
popUpWidnow's type is nominal, range is ('-1', '0', '1')
SSLfinal_State's type is nominal, range is ('1', '-1', '0')
Request_URL's type is nominal, range is ('-1', '0', '1')
URL_of_Anchor's type is nominal, range is ('-1', '0', '1')
web_traffic's type is nominal, range is ('1', '0', '-1')
URL_Length's type is nominal, range is ('1', '-1', '0')
age_of_domain's type is nominal, range is ('1', '-1')
having_IP_Address's type is nominal, range is ('0', '1')

Result's type is nominal, range is ('0', '1', '-1'))

Data coding is as follows:
"1" - Legitimate Website
"0" - Suspicious
"-1" - Malicious Website

Metrics for Performance Evaluation of Classifier

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation of Classifier

		PREDICTED CLASS	
		Class=Yes (Positive)	Class=No (Negative)
ACTUAL CLASS	Class=Yes (Positive)	a	b
	Class=No (Negative)	c	d

- The entries in the confusion matrix have the following meaning :
  - a is the number of **correct** predictions that an instance is **positive,**
  - *b* is the number of **incorrect** of predictions that an instance **negative**,
  - *c* is the number of **incorrect** predictions that an instance is **positive**, and
  - d is the number of **correct** predictions that an instance is **negative.**

Metrics for Performance Evaluation of Classifier

- The *accuracy* (*AC*)- is the proportion of the total number of predictions that were correct. It is determined using the equation:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

Contd…

- The *recall* or *true positive rate* (*TP*) is the proportion of positive cases that were correctly identified, as calculated using the equation:

$$\mathrm{TP} = \frac{a}{a+b} = \frac{TP}{TP+FN}$$

- The *false positive rate* (*FP*) is the proportion of negatives cases that were incorrectly classified as positive, as calculated using the equation:

$$\mathrm{FP} = \frac{c}{c+d} = \frac{FP}{FP+TN}$$

Contd…

- The *true negative rate* (*TN*) is defined as the proportion of negatives cases that were classified correctly, as calculated using the equation:

$$\mathrm{TN} = \frac{d}{d+c} = \frac{TN}{TN+FP}$$

- The *false negative rate* (*FN*) is the proportion of positives cases that were incorrectly classified as negative, as calculated using the equation:

$$\mathrm{FN} = \frac{b}{b+a} = \frac{FN}{FN+TP}$$

Contd.....

- *The precision* (*P*) is the proportion of the predicted positive cases that were correct, as calculated using the equation:

$$\mathrm{P} = \frac{a}{c+a} = \frac{TP}{FP+TP}$$

Example

- Suppose we train a model to predict whether an email is **Spam** or **Not Spam**. After training the model, we apply it to a test set of 500 new email messages (also labeled) and the model produces the contingency matrix below.

		True Class	
		Spam	Not Spam
Predicted Class	Spam	70	10
	Not Spam	40	380

(a) Compute the precision of this model with respect to the **Spam class.**

Precision with respect to SPAM = # correctly predicted as SPAM / # predicted as SPAM

= 70 / (70 + 10) = 70 / 80.

Cond…

(b) Compute the recall of this model with respect to the **Spam class.**

recall with respect to SPAM = # correctly predicted as SPAM / # truly SPAM

= 70 / (70 + 40) = 70 / 110.

- **High-precision and low recall with respect to SPAM:** whatever the model classifies as SPAM is probably SPAM. However, many emails that are truly SPAM are misclassified as NOT SPAM i.e <False Negative (False Acceptance)>

- **High recall and low precision with respect to SPAM:** the model filters all the SPAM emails, but also incorrectly classifies some genuine emails as SPAM i.e. <False Positive (False Rejectance)>.

End of Presentataion