UNIVERSITY OF ABERDEEN

CELEBRATING 525 YEARS 1495 – 2020

ABERDEEN 2040

Decision Trees

Data Mining & Visualisation

Lecture 11

2025

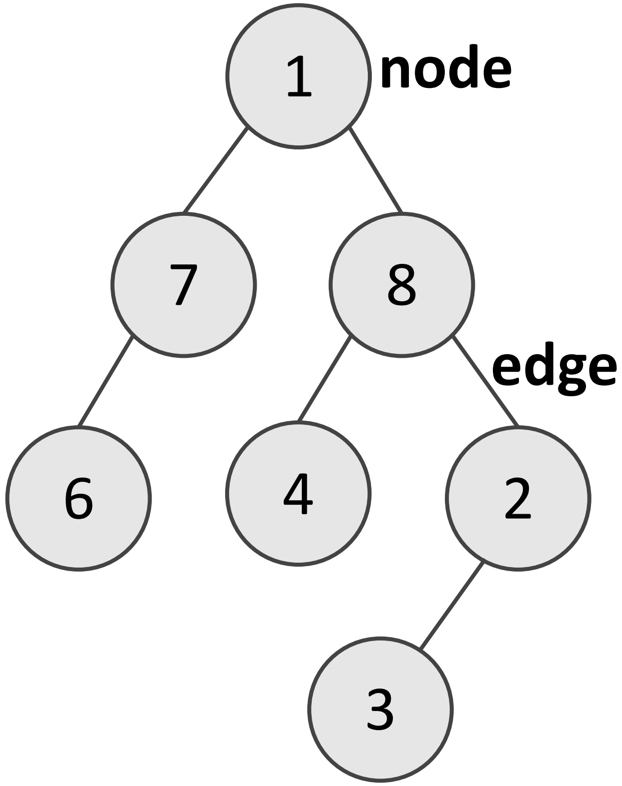
Today...

- Recap on trees
- Entropy
- Concept of ‘purity of leaf node’
- Information Gain

Trees

Trees

A tree is a non-linear data structure that consists of nodes connected by edges.

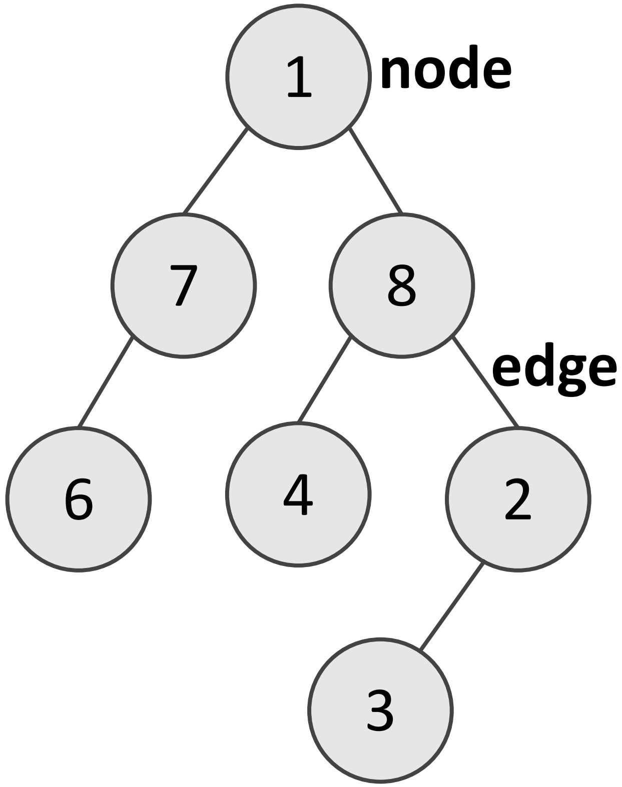


They are commonly used across various topics in computer science.

Trees

Trees are collections of *nodes*:

- They must have one root node, maximum!
- Each node has:
  - a value (e.g., a number), and
  - a list of references to other nodes
- A tree can be **empty**.

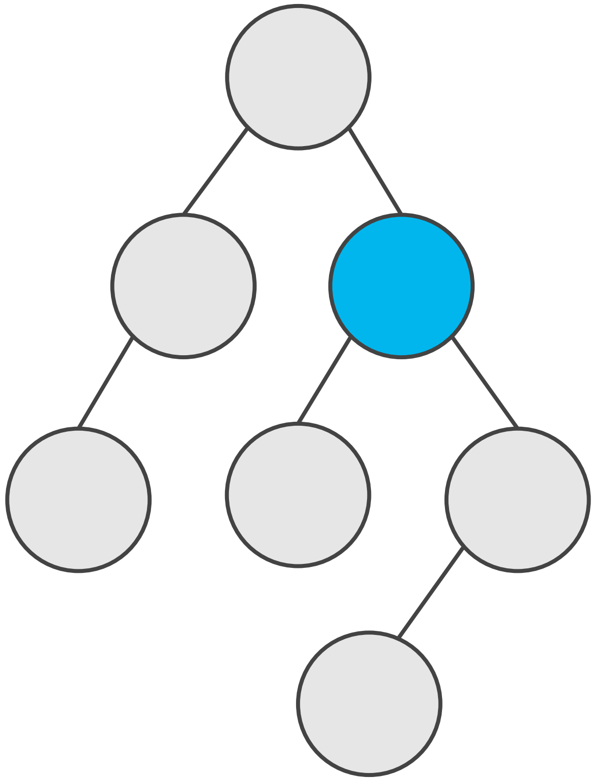


Edges represent connections between two nodes:

- They are sometimes called "arc", "branch" or "link".

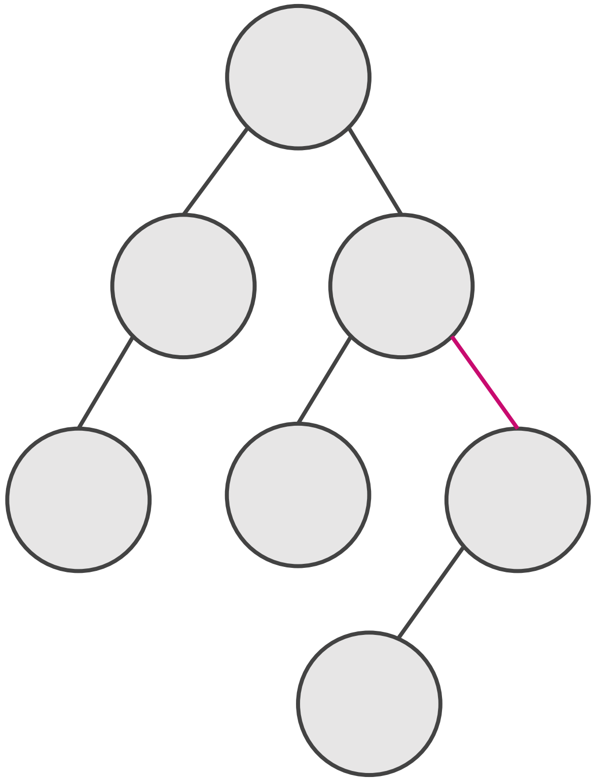
Tree Terminology

Node
Edge
Root
Leaf
Parent
Child
Siblings
Ancestor
Descendant
Subtree



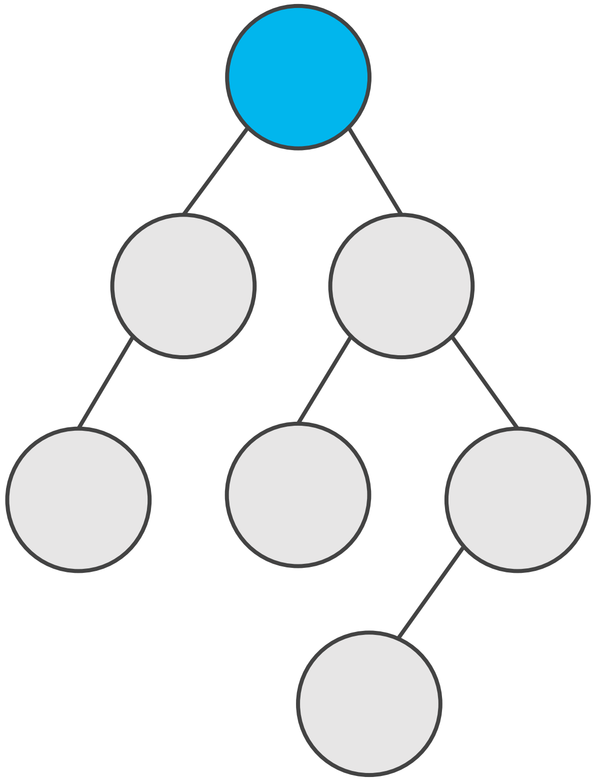
Tree Terminology

Node
Edge
Root
Leaf
Parent
Child
Siblings
Ancestor
Descendant
Subtree



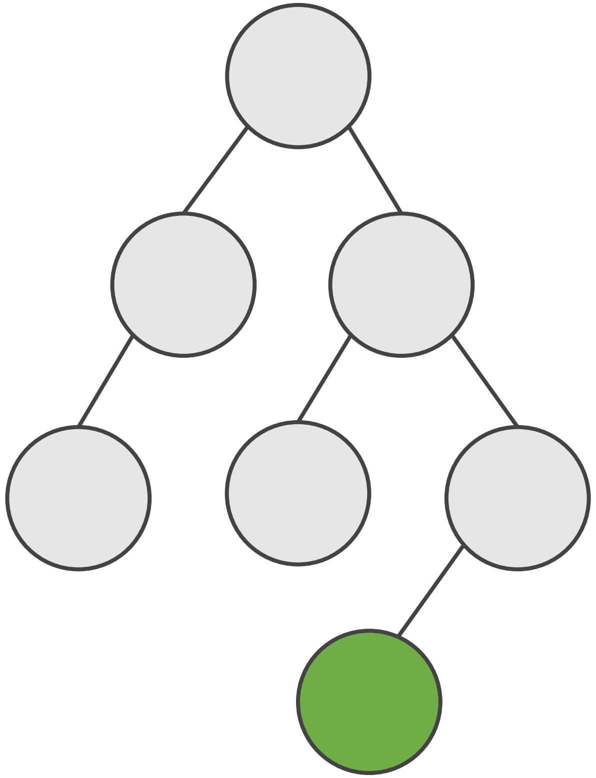
Tree Terminology

Node
Edge
Root
Leaf
Parent
Child
Siblings
Ancestor
Descendant
Subtree



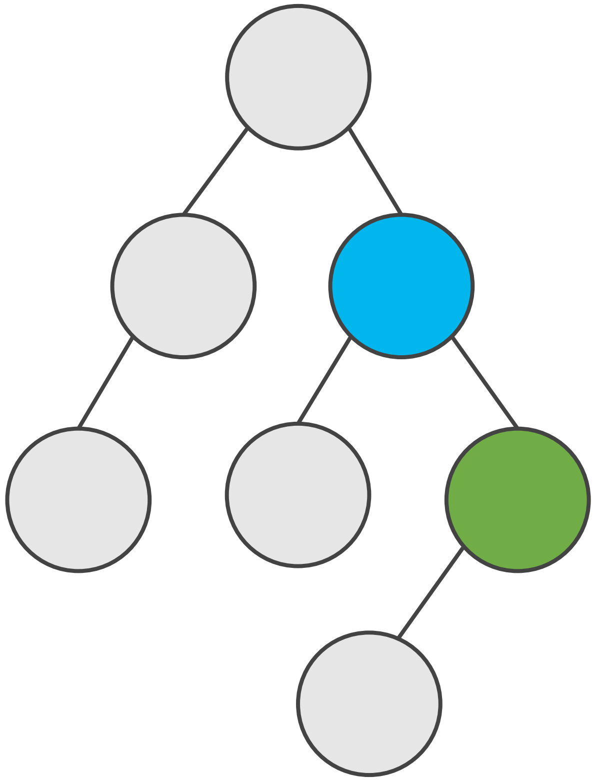
Tree Terminology

Node
Edge
Root
Leaf
Parent
Child
Siblings
Ancestor
Descendant
Subtree



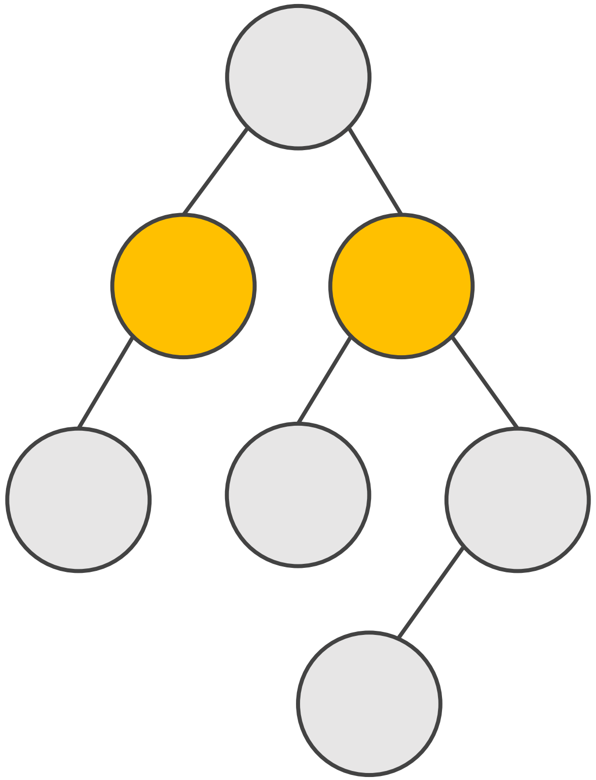
Tree Terminology

Node
Edge
Root
Leaf
Parent
Child
Siblings
Ancestor
Descendant
Subtree



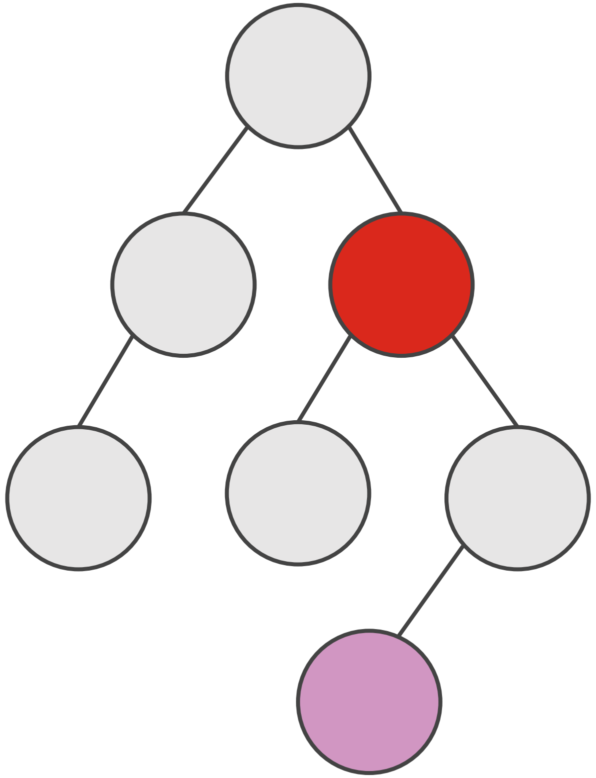
Tree Terminology

Node
Edge
Root
Leaf
Parent
Child
Siblings
Ancestor
Descendant
Subtree



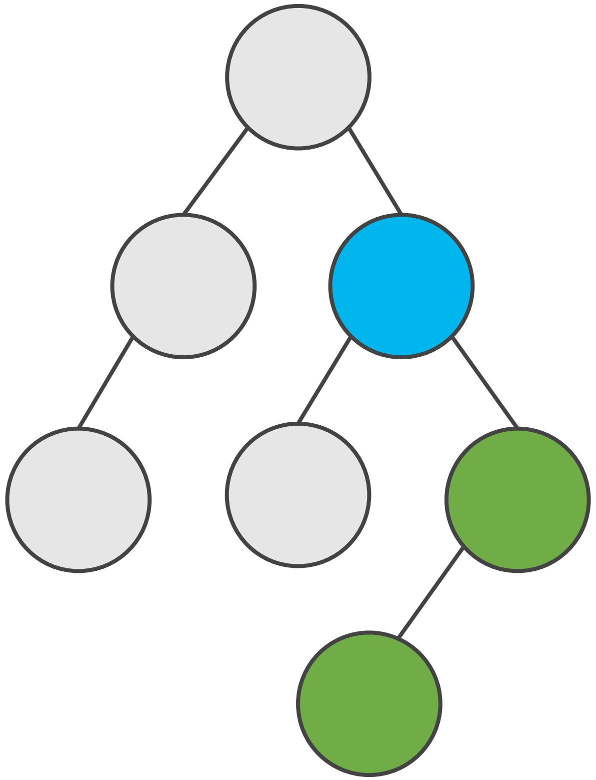
Tree Terminology

Node
Edge
Root
Leaf
Parent
Child
Siblings
Ancestor
Descendant
Subtree



Tree Terminology

Node
Edge
Root
Leaf
Parent
Child
Siblings
Ancestor
Descendant
Subtree

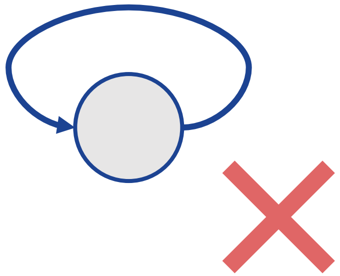


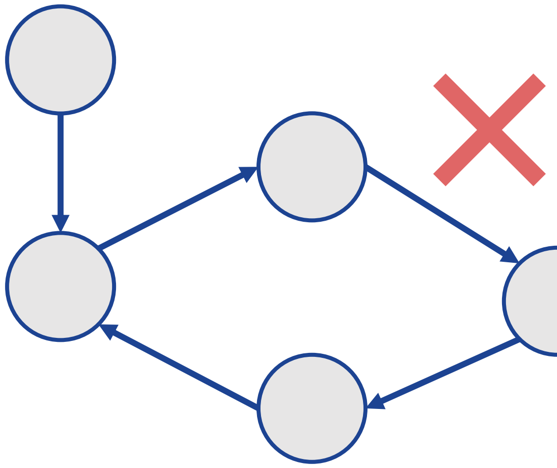
Tree Terminology

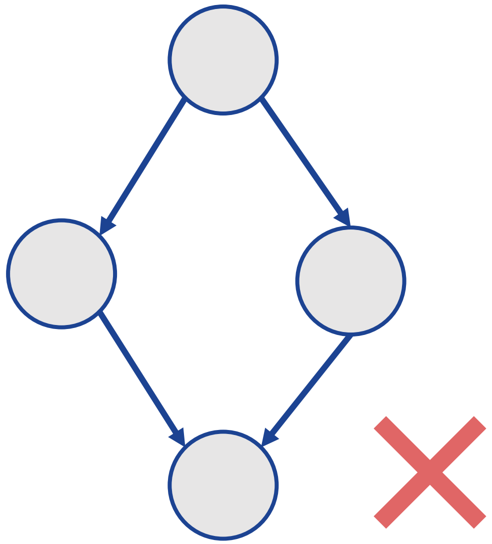
Root	The "top" node, and the only one without parents
Leaf	A node with no children
Parent	The node directly above a given node *n* in the tree
Child	The node directly below a given node *n* in the tree
Siblings	Nodes with the same parent node
Ancestor	An ancestor of *n* is a node along the path from the root to *n*
Descendant	A descendant of *n is* a node along the path from *n* to a leaf node
Subtree	A subtree of a node *n* is a tree rooted by a child node of *n*

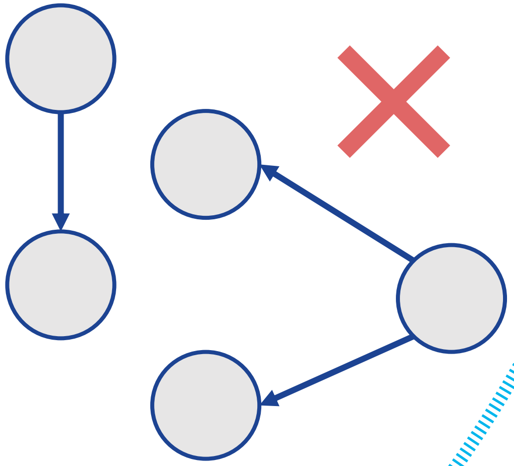
Some Non-Trees

- The root cannot be a child node
- Trees can only have one root node
- No node can have more than one parent









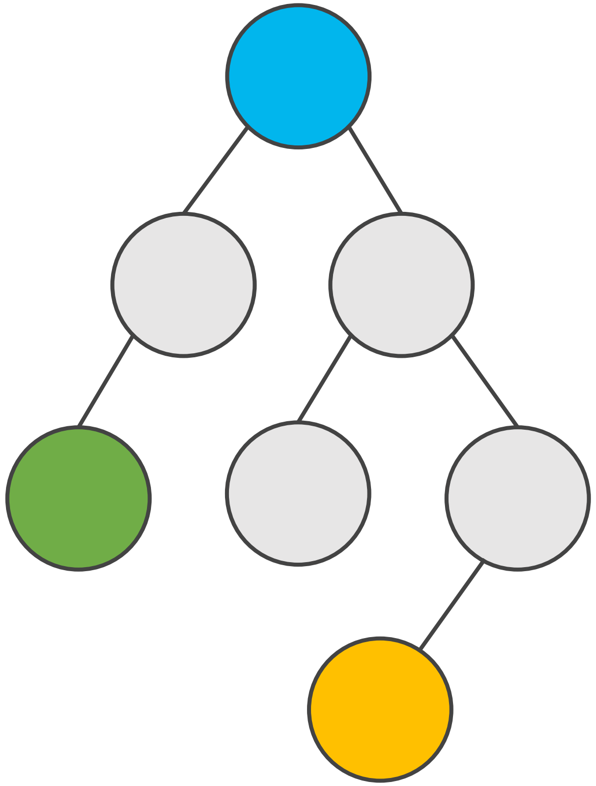
Path and Depth

A path is a sequence of connected edges.

For every node, there is a unique path to the root node (a definition of a tree!).

The depth of a node n is the length of the path (number edges).

What are the depths of the three highlighted nodes on the right?



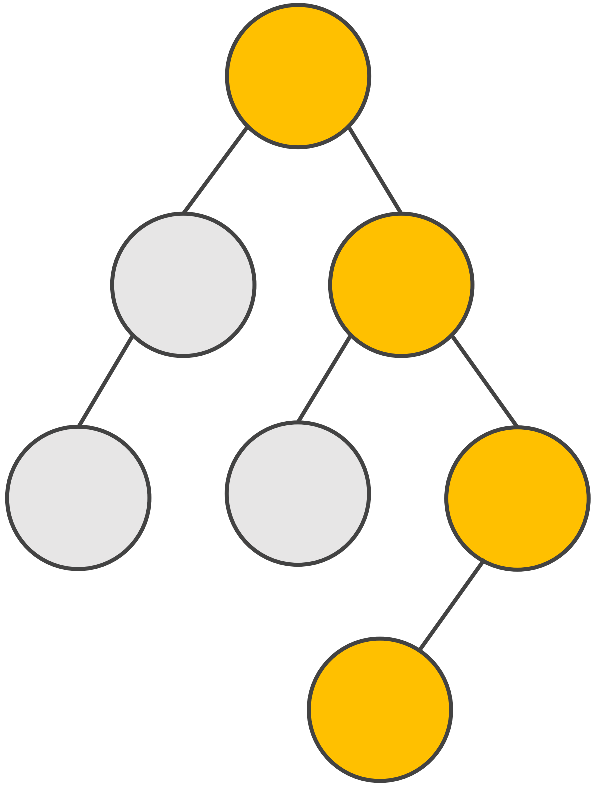
Height and Size

Height: is the longest path from root to leaf.

Size: the number of nodes in the tree.

Looking at the tree shown on the right:

- What is its height? 3
- What is its size? 7



Height and Size

Height: is the longest path from root to leaf.

Size: the number of nodes in the tree.

What about an empty tree?

- It’s size is easy - zero (no nodes)
- … but what about its height? -1

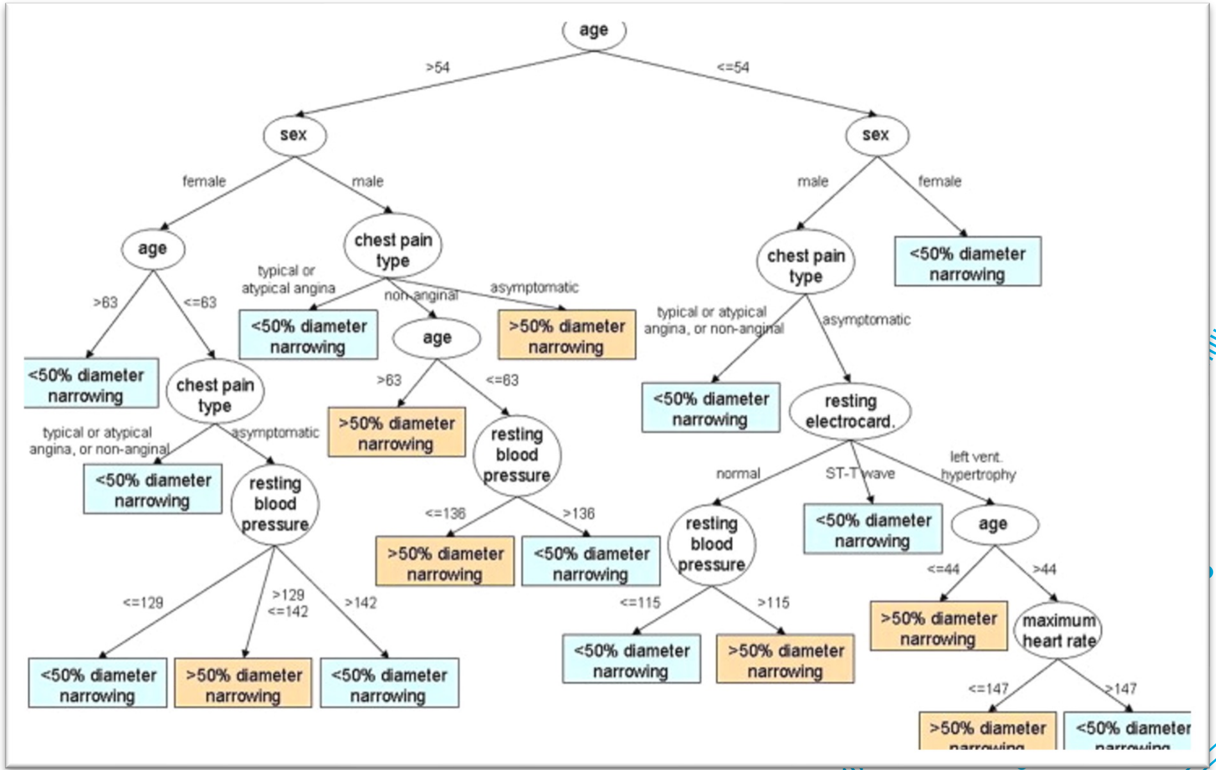
Why Use The Tree Data Structure?

In order to perform any operation on a linear data structure (such as arrays, linked lists, queues, etc.), the time complexity increases with the increase with the amount of data.

Given that trees are a non-linear data structure, they are often more computationally efficient to traverse, allowing for a quicker and easier way to access the data.

Trees are Ubiquitous in Computer Science

- Compilation of computer programs (abstract syntax trees)
- Natural language processing (NLP) (parse trees)
- File systems (folders, subfolders, …)
- Biological systems (anatomy)
- Decision trees (e.g., medical diagnosis)
- Efficient searching and sorting



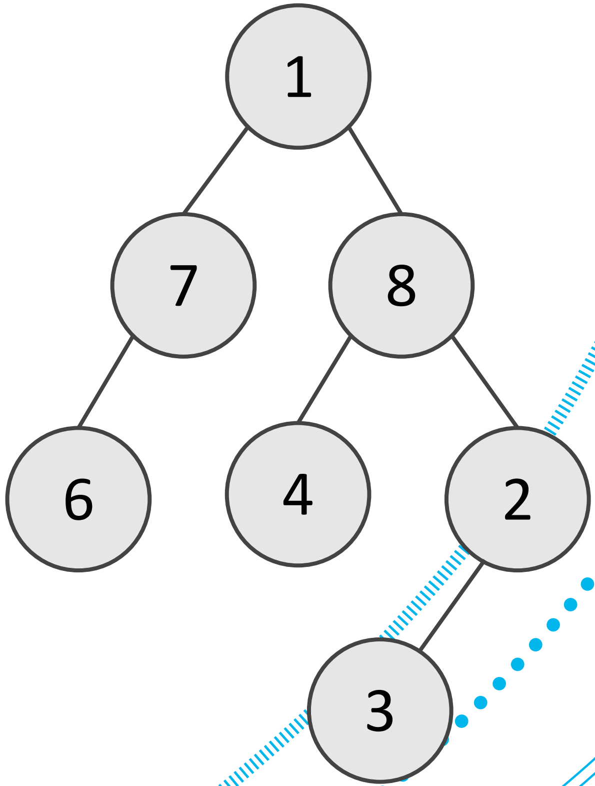
Decision Trees

Decision Trees

Decision trees are a type of predictive modelling algorithm used in statistics, data mining, and ML.

They help in making decisions by learning simple decision rules inferred from the data features.

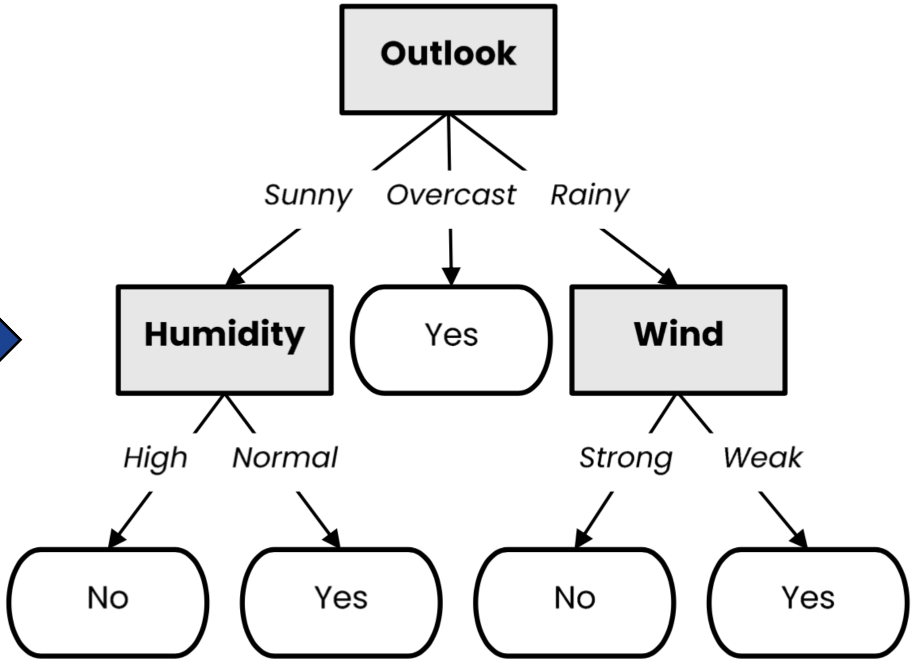
Decision trees are versatile, easy to understand, and applicable to a wide range of problems.



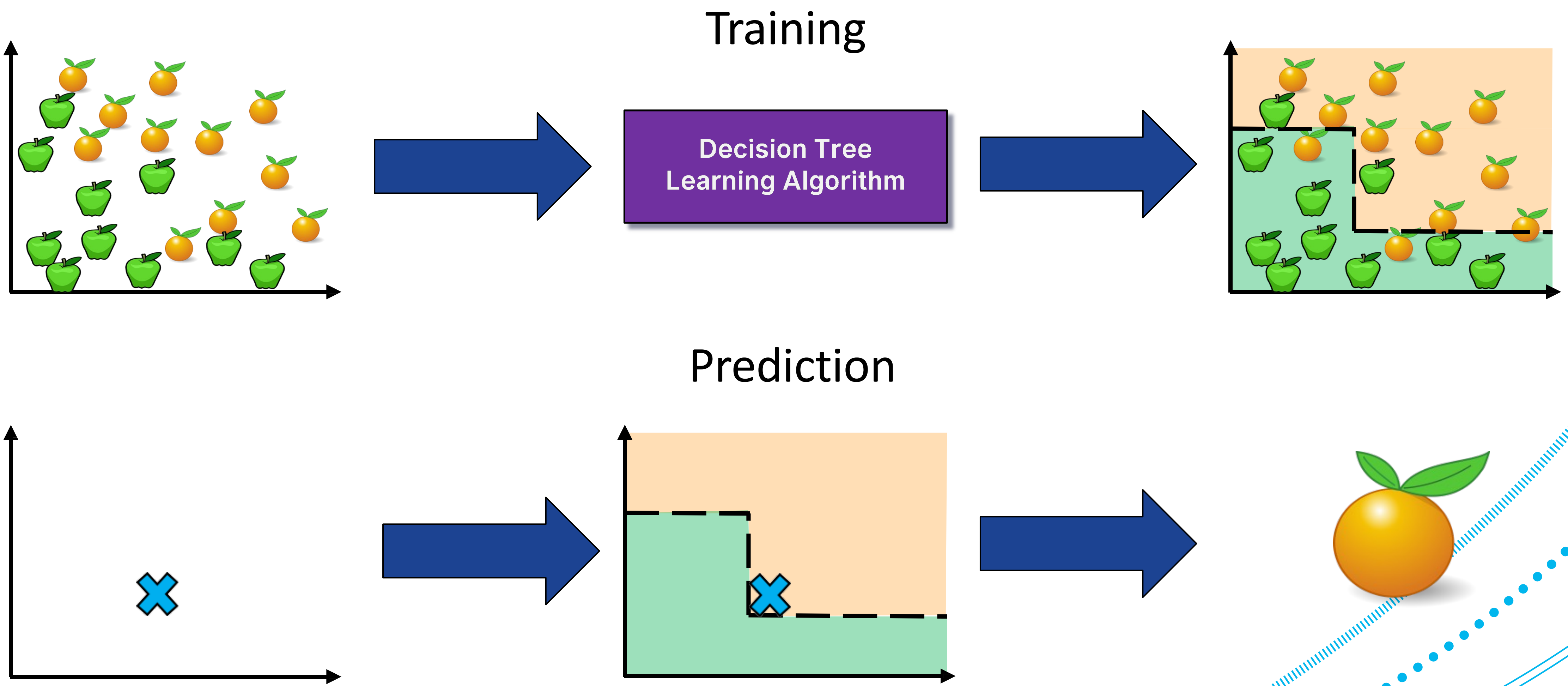
Decision Trees

We use labeled data to obtain a suitable decision tree for future predictions.

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Hot	High	Weak	**No**
Sunny	Hot	High	Strong	**No**
Overcast	Hot	High	Weak	**Yes**
Rainy	Mild	High	Weak	**Yes**
Rainy	Cool	Normal	Weak	**Yes**
Rainy	Cool	Normal	Strong	**No**
Overcast	Cool	Normal	Strong	**Yes**
Sunny	Mild	High	Weak	**No**
Sunny	Cool	Normal	Weak	**Yes**
Rainy	Mild	Normal	Weak	**Yes**
Sunny	Mild	Normal	Strong	**Yes**
Overcast	Mild	High	Strong	**Yes**
Overcast	Hot	Normal	Weak	**Yes**
Rainy	Mild	High	Strong	**No**

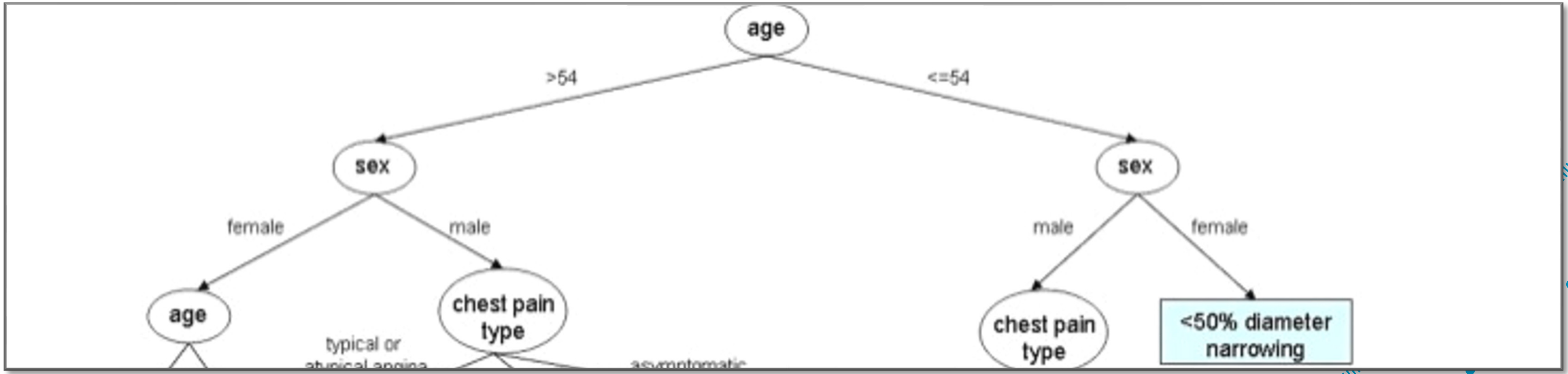


Decision Trees: Training and Prediction



Decision Trees: Interpretation

They are another example of a supervised learning approach with a high amount of interpretability.



Decision Trees: How to Train

Let's say we have a dataset of when we should play tennis.

How do we turn that into a decision tree? How do we 'split' this data up?

First, we need to learn about the 'purity' of the information.

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Hot	High	Weak	**No**
Sunny	Hot	High	Strong	**No**
Overcast	Hot	High	Weak	**Yes**
Rainy	Mild	High	Weak	**Yes**
Rainy	Cool	Normal	Weak	**Yes**
Rainy	Cool	Normal	Strong	**No**
Overcast	Cool	Normal	Strong	**Yes**
Sunny	Mild	High	Weak	**No**
Sunny	Cool	Normal	Weak	**Yes**
Rainy	Mild	Normal	Weak	**Yes**
Sunny	Mild	Normal	Strong	**Yes**
Overcast	Mild	High	Strong	**Yes**
Overcast	Hot	Normal	Weak	**Yes**
Rainy	Mild	High	Strong	**No**

Entropy

Entropy

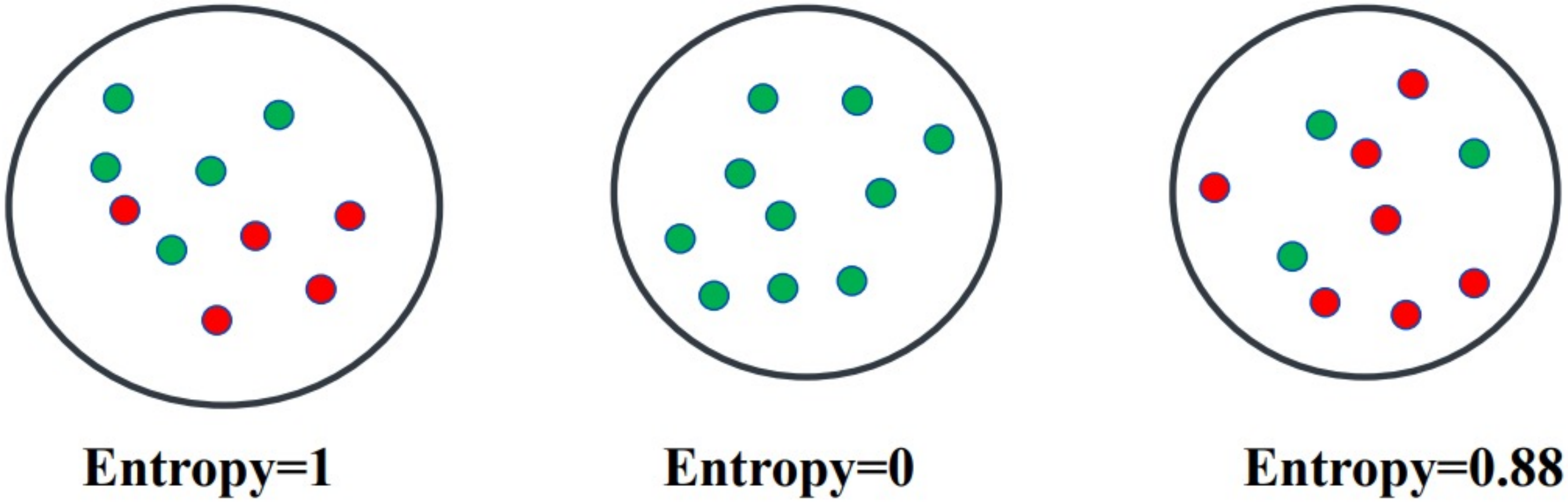
Entropy is the machine learning metric that measures the **impurity**, or *unpredictability*, in the system.

It is the measurement of disorder in the information being processed.

We use this measurement to determine how our decision tree should split the data.

Entropy

Different cases

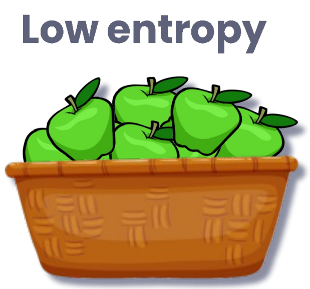


If a dataset contains an equal # of positive and negative data points, entropy is 1.

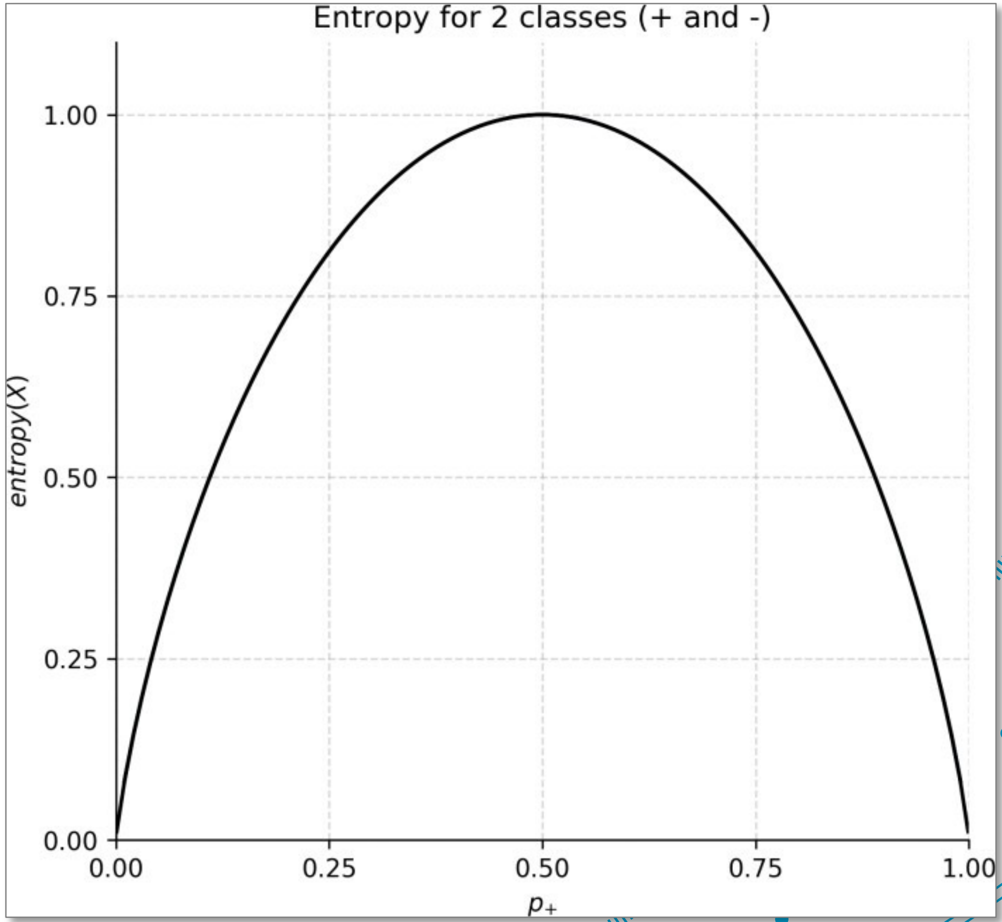
If a dataset contains only positive or only negative data points, then entropy is 0.

Entropy

Entropy measures the degree of randomness in data.



<ins>Lower entropy implies greater predictability!</ins>



Calculating Entropy

For a set of samples X with k classes:

$$entropy(X) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

...where p_i is the proportion of elements of class i.

Calculating Entropy

In general, the entropy of a random variable V with values $\mathbf{v}_k$, each with probability $P(v_k)$, is defined as Entropy:

$$H(V) = -\sum_k P(v_k) \log_2 P(v_k)$$

The entropy of a fair coin flip:

$$H(Fair) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$$

Entropy

A random variable with only one value (e.g. a coin that always comes up heads), has no uncertainty.

Thus its entropy is defined as zero; We gain no information by observing its value.

$$- (1 * \log_2 1 \;+\; 0 * \log_2 0) \;=\; 0$$

(heads) (tails)

Calculating Entropy

$$H(V) = -\sum_k P(v_k) \log_2 P(v_k)$$

Example: If we had a total 10 data points in our dataset with 3 belonging to positive class and 7 belonging to negative class:

$$-\ (3/10 * \log_2 (3/10)\ +\ 7/10 * \log_2 (7/10))\ \approx\ 0.876$$

High entropy means a low level of purity.

Information Gain

Information Gain

Entropy is a useful measure of disorder.

However, we still need a measure of “good” and “bad” for attributes. One way to do is to compute the information gain.

Information gain is defined as the pattern observed in the dataset, and the reduction in the entropy.

Information Gain

Mathematically, information gain can be expressed with this formula:

Information Gain = (Entropy of parent node) - (Entropy of child node)

But essentially, we’re looking at the entropy of a node <u>after we split on an attribute</u>, compared to <u>before we split</u>.

In other words, how much **information** do we **gain** by splitting?

Building a Decision Tree Using Information Gain

At every step in our decision tree, we want to maximise the information gained.

- An attribute with the highest information gain from a set should be selected as the parent (root) node.
- We then build child nodes for every value of that attribute.
- And then repeat iteratively until we construct the whole tree.

Choosing the Best Attribute

Example: Let's say we are looking to build a decision tree for whether we should wait for a table at a restaurant.

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

WillWait represents a binary variable, **our DV**, indicating whether or not we should wait.

Source: S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach. Fourth Ed.

Choosing the Best Attribute

Example: Let’s say we are looking to build a decision tree for whether we should wait for a table at a restaurant.

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Pat represents the number of Patrons; i.e. how many people are in the restaurant.

The possible values are: **None**, **Some**, and **Full**.

Source: S. Russell and P. Norvig, “Artificial Intelligence: A Modern Approach. Fourth Ed.

Choosing the Best Attribute

At the root node of the restaurant problem, there are:

- 6 **Yes** cases
- 6 **No** cases

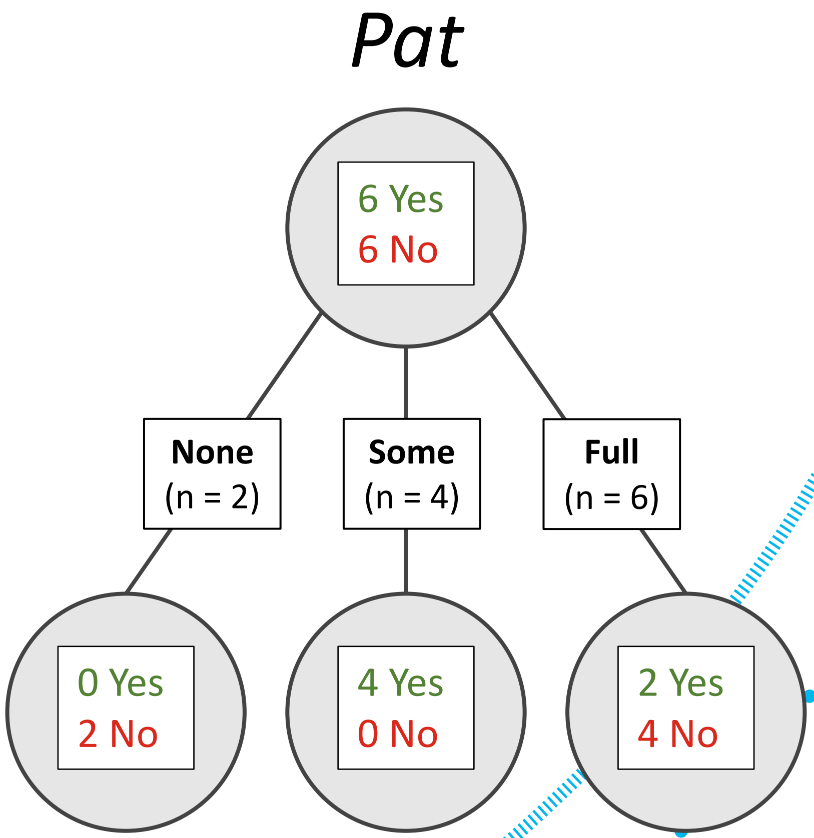
So we know that the Entropy(Parent), i.e. before we split, is 1.

Output *WillWait*
$y_1 = Yes$
$y_2 = No$
$y_3 = Yes$
$y_4 = Yes$
$y_5 = No$
$y_6 = Yes$
$y_7 = No$
$y_8 = Yes$
$y_9 = No$
$y_{10} = No$
$y_{11} = No$
$y_{12} = Yes$

Choosing the Best Attribute

So what happens if we split on the *Patron* attribute?

We start off by calculating the Entropy for each value of Patron.



Choosing the Best Attribute

E(*Pat* = None) = 0

E(*Pat* = Some) = 0

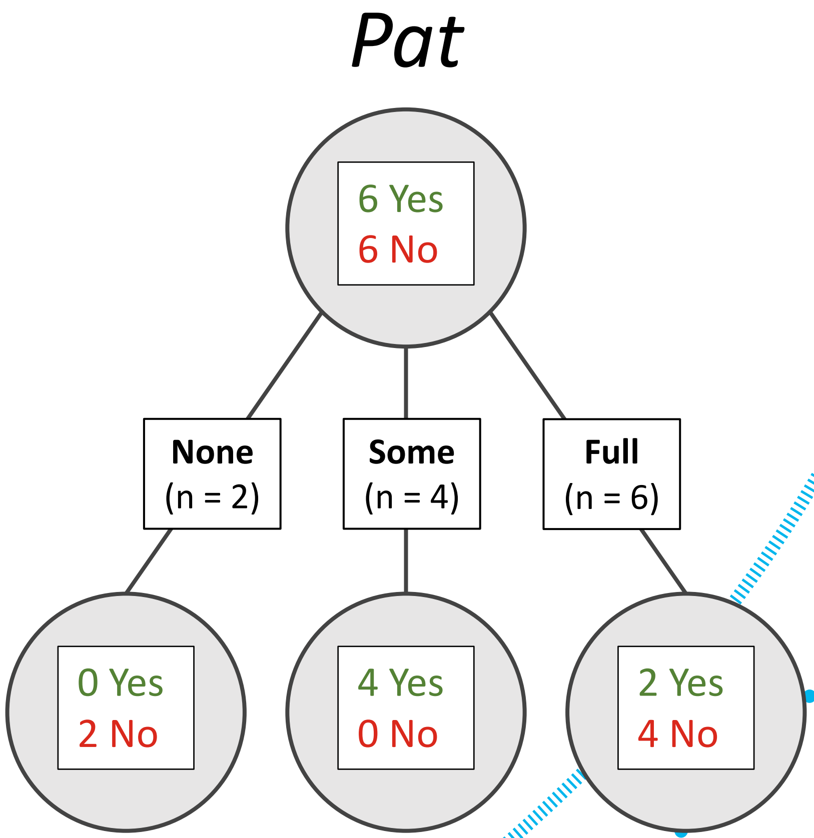
E(*Pat* = Full)

$= -(2/6 * \log_2 (2/6) + 4/6 * \log2 (4/6))$

$\approx -(0.33 * -1.59 + 0.67 * -0.59)$

$\approx -(-0.52 + -0.40)$

≈ 0.92



Choosing the Best Attribute

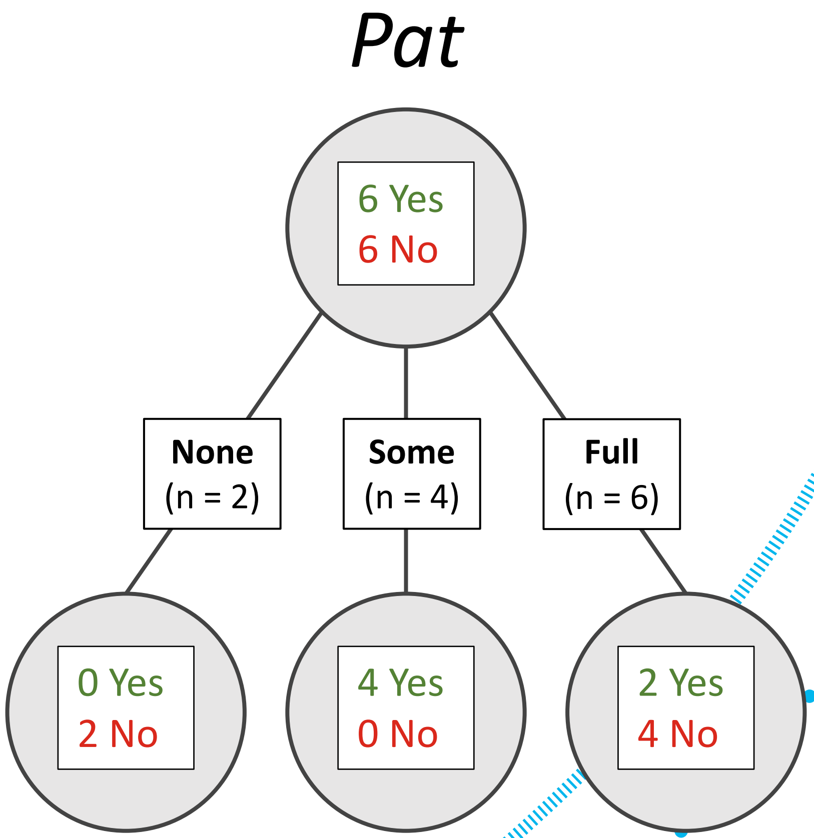
Next, we calculate the weighted average of the Entropy for each node.

$E(\mathit{Pat} = \text{None}) = 0$

$E(\mathit{Pat} = \text{Some}) = 0$

$E(\mathit{Pat} = \text{Full}) \approx 0.92$

$$E(\mathit{Pat}) \approx (2/12 * 0) + (4/12 * 0) + (6/12 * 0.92)$$
$$\approx 0.46$$



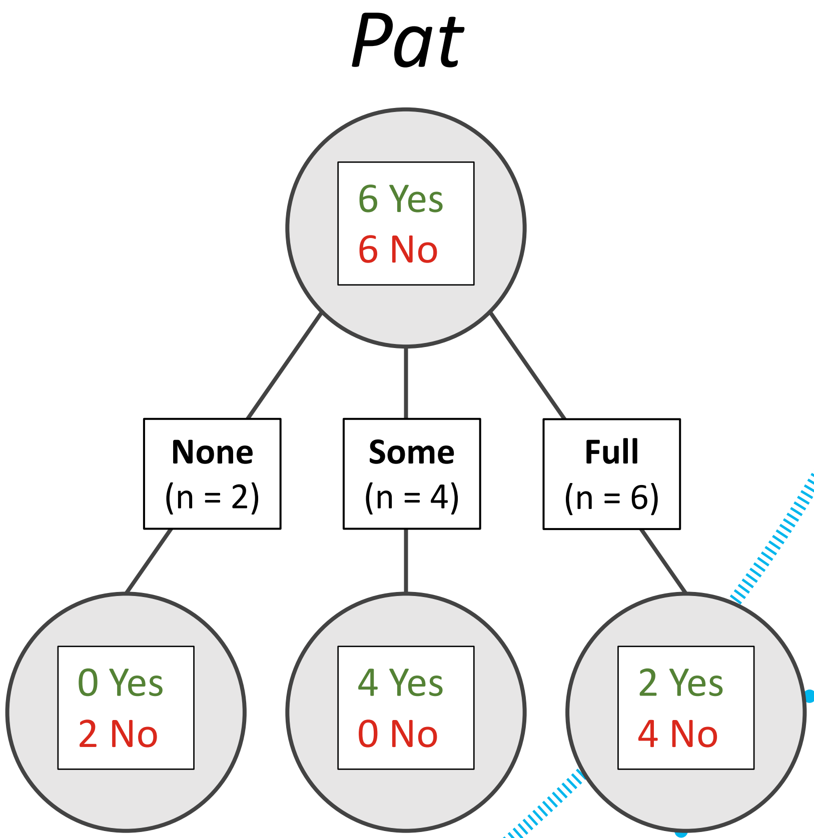
Choosing the Best Attribute

Finally, we calculate the Information Gain.

IG = (Entropy of parent node) - (Entropy of child node)

$\approx$ 1 – 0.46

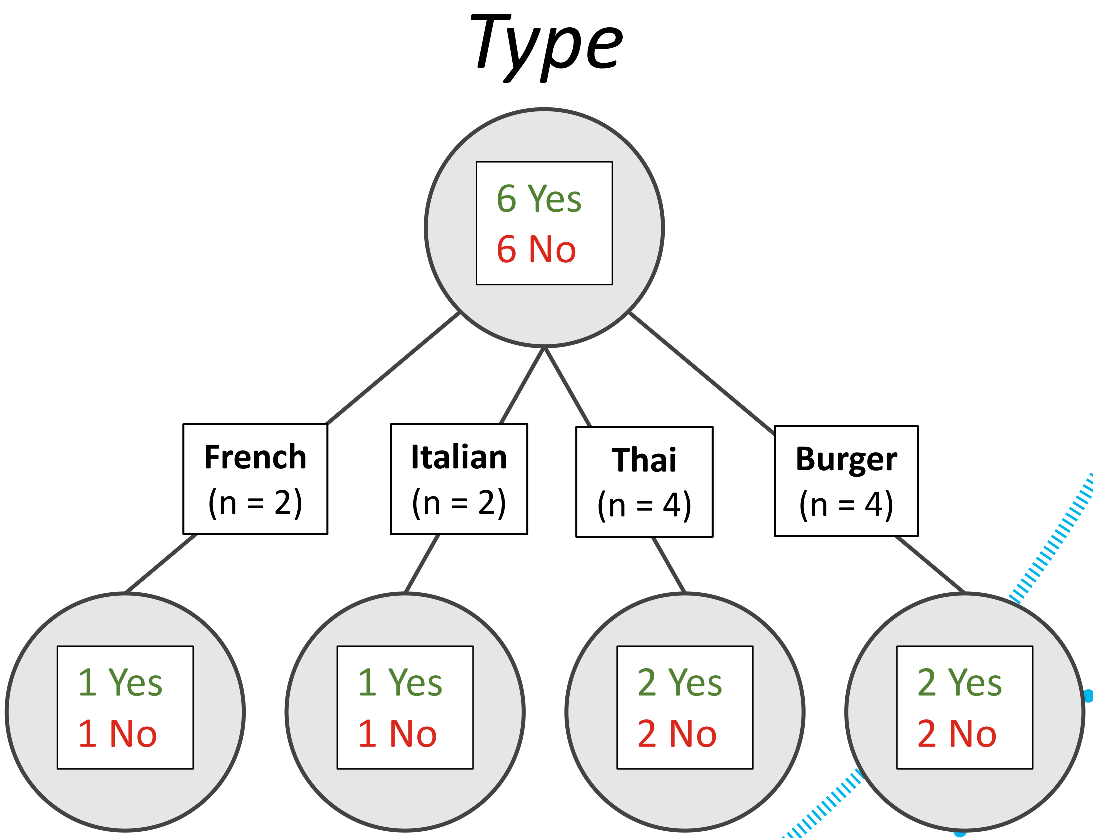
IG(*Pat*) $\approx$ 0.54



Choosing the Best Attribute

Let's say we repeat that process with all other attributes (but we'll just show one as an example):

The *Type* of restaurant.



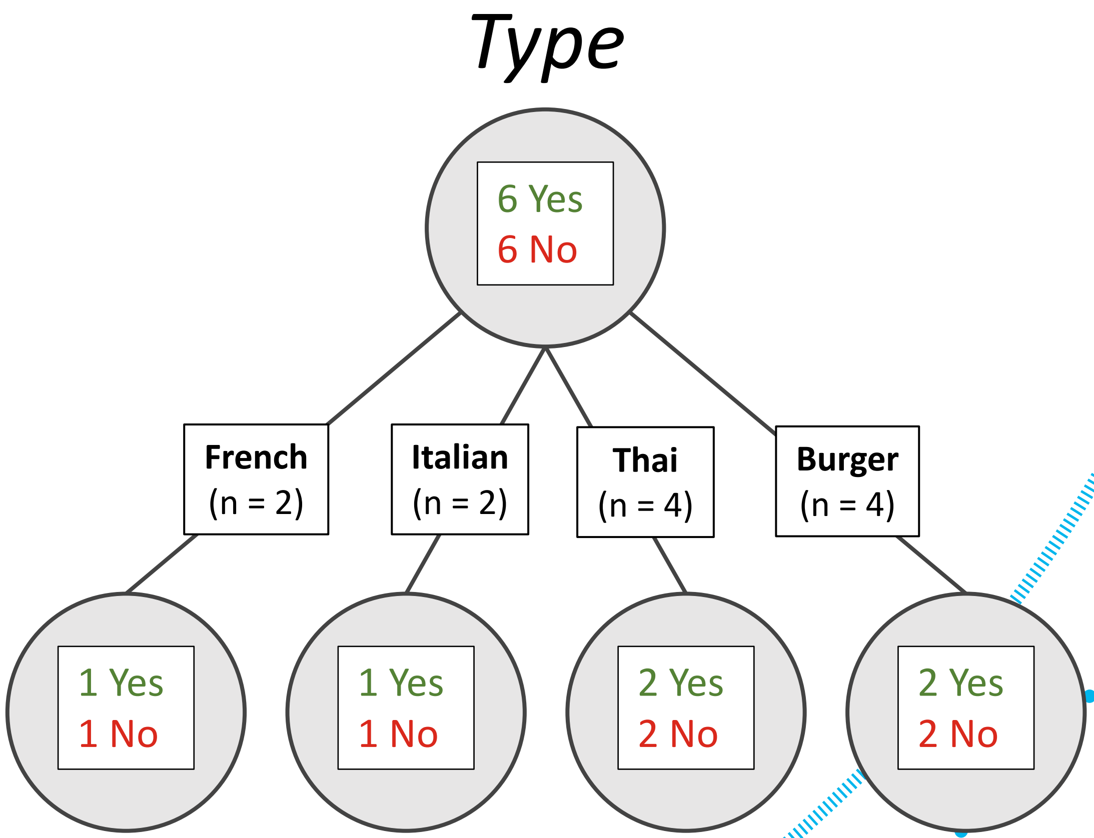
Choosing the Best Attribute

E(*Type* = French) = 1

E(*Type* = Italian) = 1

E(*Type* = Thai) = 1

E(*Type* = Burger) = 1



Choosing the Best Attribute

Weighted Entropy:

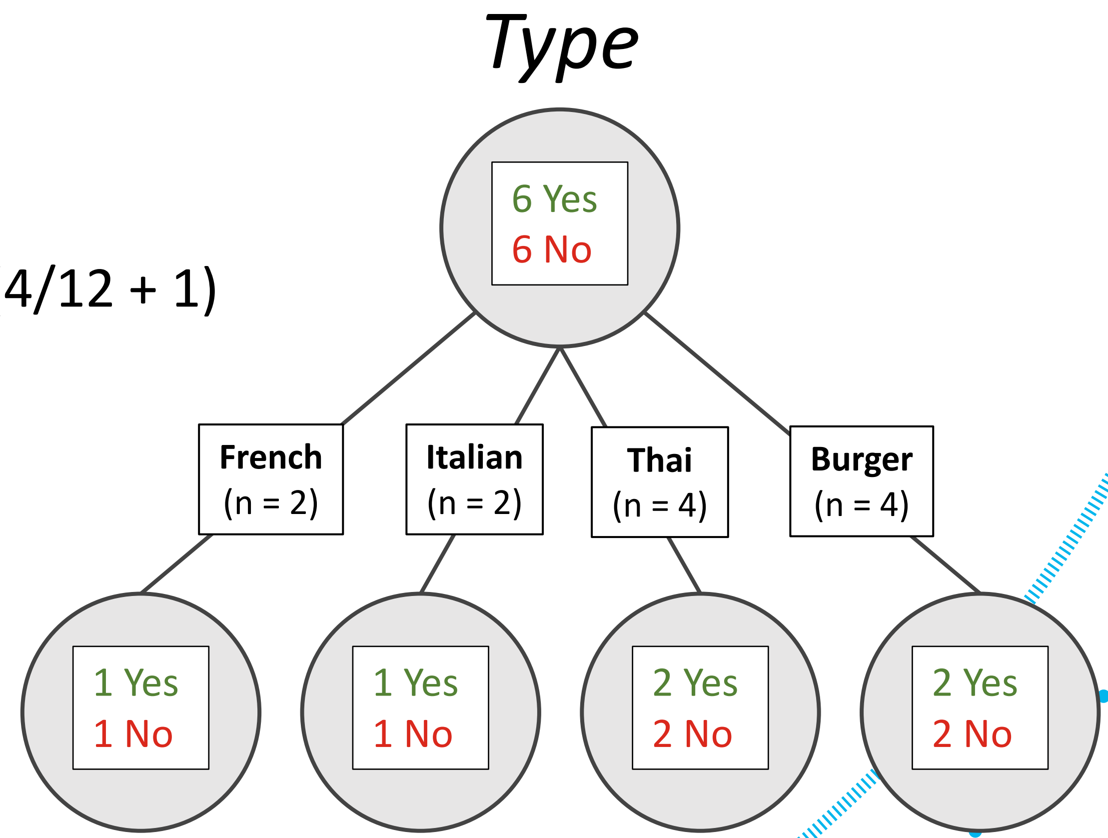
$E(Pat) = (2/12 * 1) + (2/12 * 1) + (4/12 * 1) + (4/12 + 1)$

$= 1$

Information Gain:

$IG(Type) = 1 - 1$

$= 0$



Choosing the Best Attribute

In other words, <u>we gain more information by splitting on *Pat* than by *Type*</u>.