DECISION TREES

INNOMATİCS
RESEARCH LABS

WHAT IS CLASSIFICATION?

> “Classification is a form of data analysis that extracts models describing important data classes. Such models, called classifiers, predict categorical (discrete, unordered) class labels.”

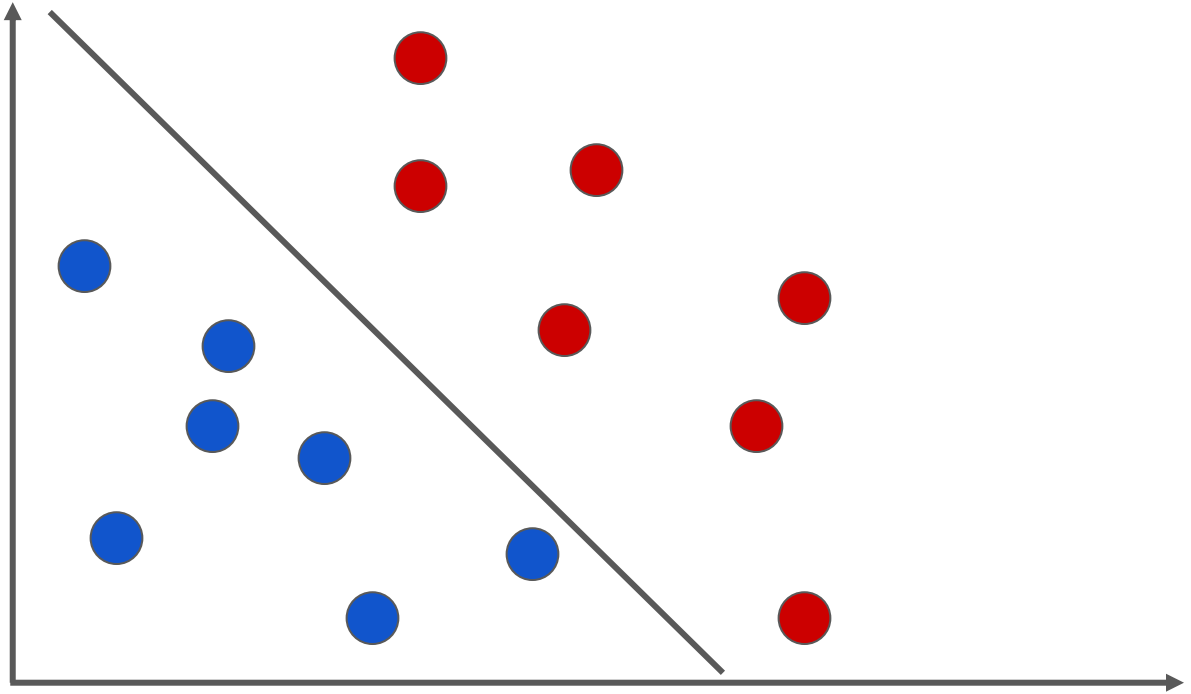
- Jiawei Han

DECISION BOUNDARY

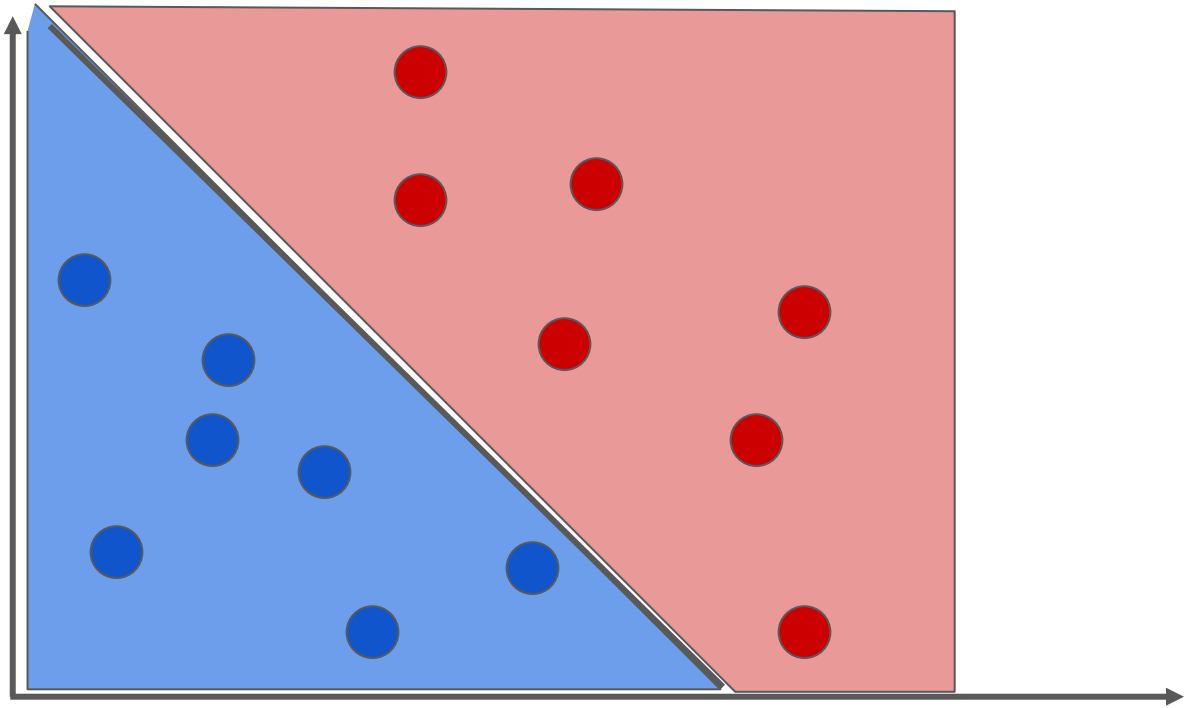
What is meant by decision boundary ?

- A decision boundary is **the region of a problem space in which the output label of a classifier is ambiguous**. If the decision surface is a hyperplane, then the classification problem is linear, and the classes are linearly separable. Decision boundaries are not always clear cut.

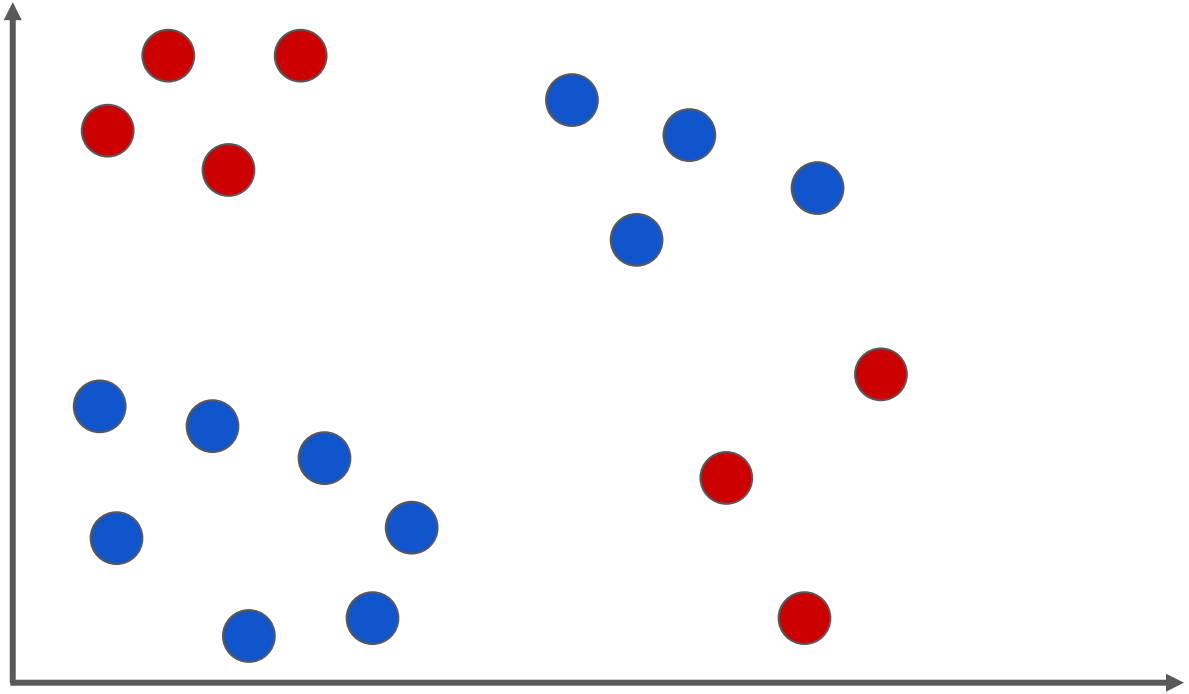
DECISION BOUNDARY



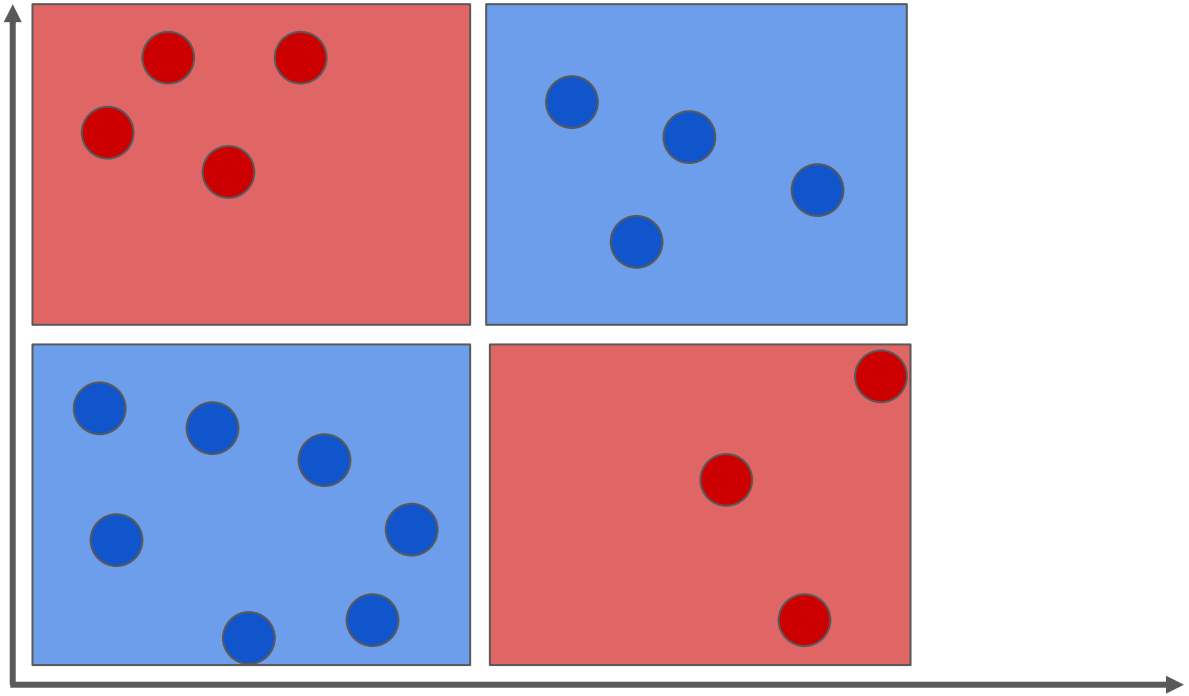
DECISION BOUNDARY



COMPLEX REGIONS



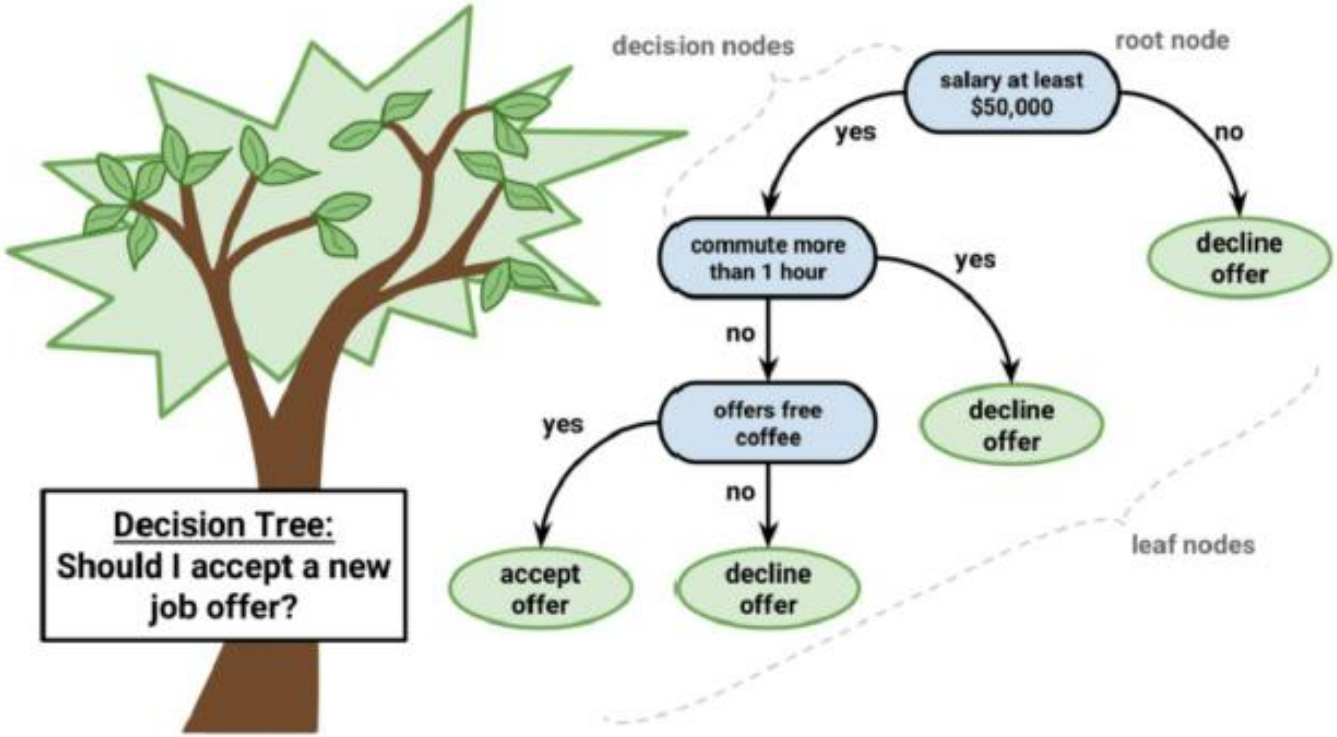
COMPLEX REGIONS



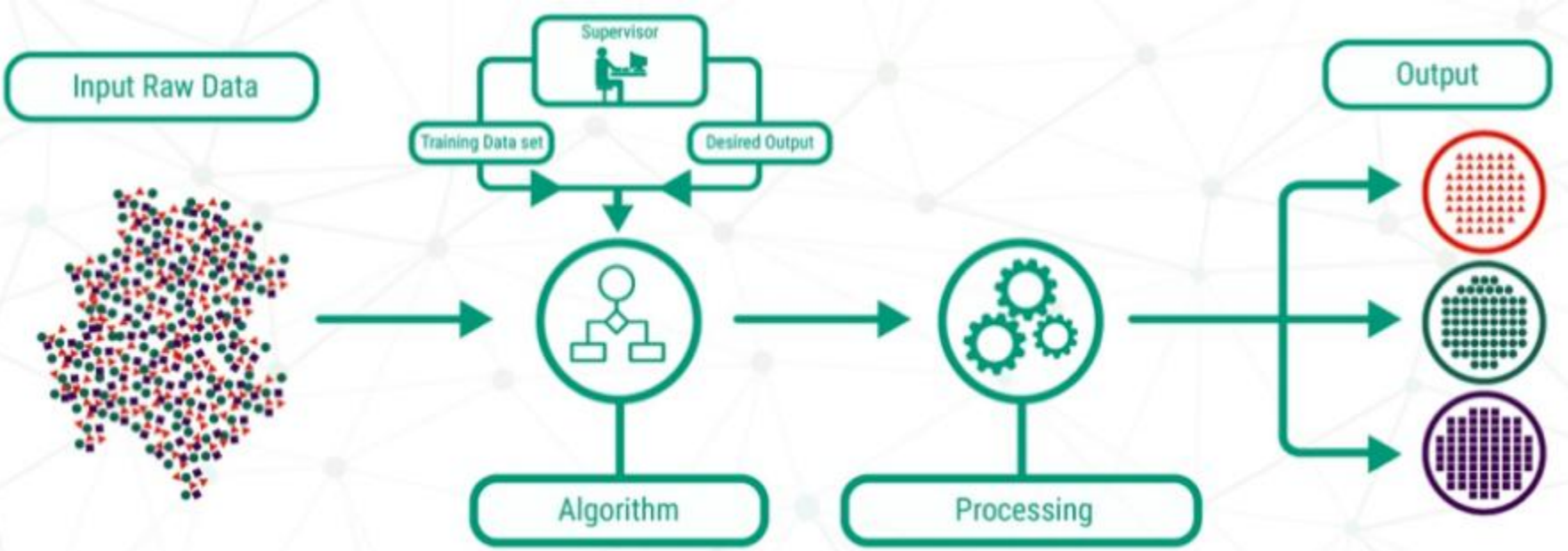
DECISION TREE

A decision tree is a flowchart-like structure in which each internal node represents a "test" or a condition on an attribute (e.g. monthly salary is greater than Rs. 30,000 or not), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

DECISION TREE



SUPERVISED LEARNING



DECISION TREE

Decision Tree is a highly popular technique for Supervised Learning. The model thus created is easy to understand and interpret.

These can be used for both classification as well as regression tasks. Decision Trees can handle high dimension data with good accuracy.

Decision Trees can be constructed using algorithms like the following:

- ID3 (Iterative Dichotomiser 3 by Ross Quinlan
- C4.5 (Iterative Dichotomiser 3 by Ross Quinlan
- CART (Classification And Regression Tree by Breiman et al)

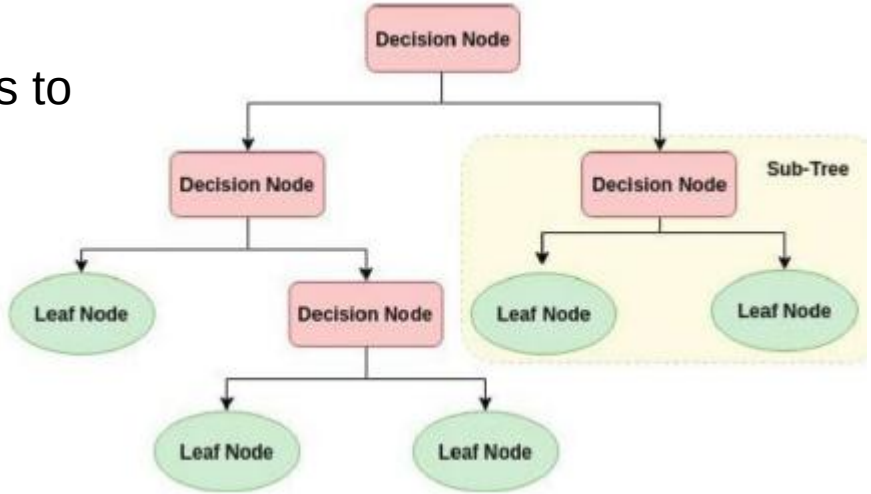
DECISION TREE - WORKING

A Decision Tree is a flowchart-like structure.

Each internal node, or decision node learns to partition the data based on attribute value. Each branch indicates the decision rule.

The path thus followed leads to the Outcome specified by the Leaf Node.

The tree is partitioned and built recursively starting from the root node.



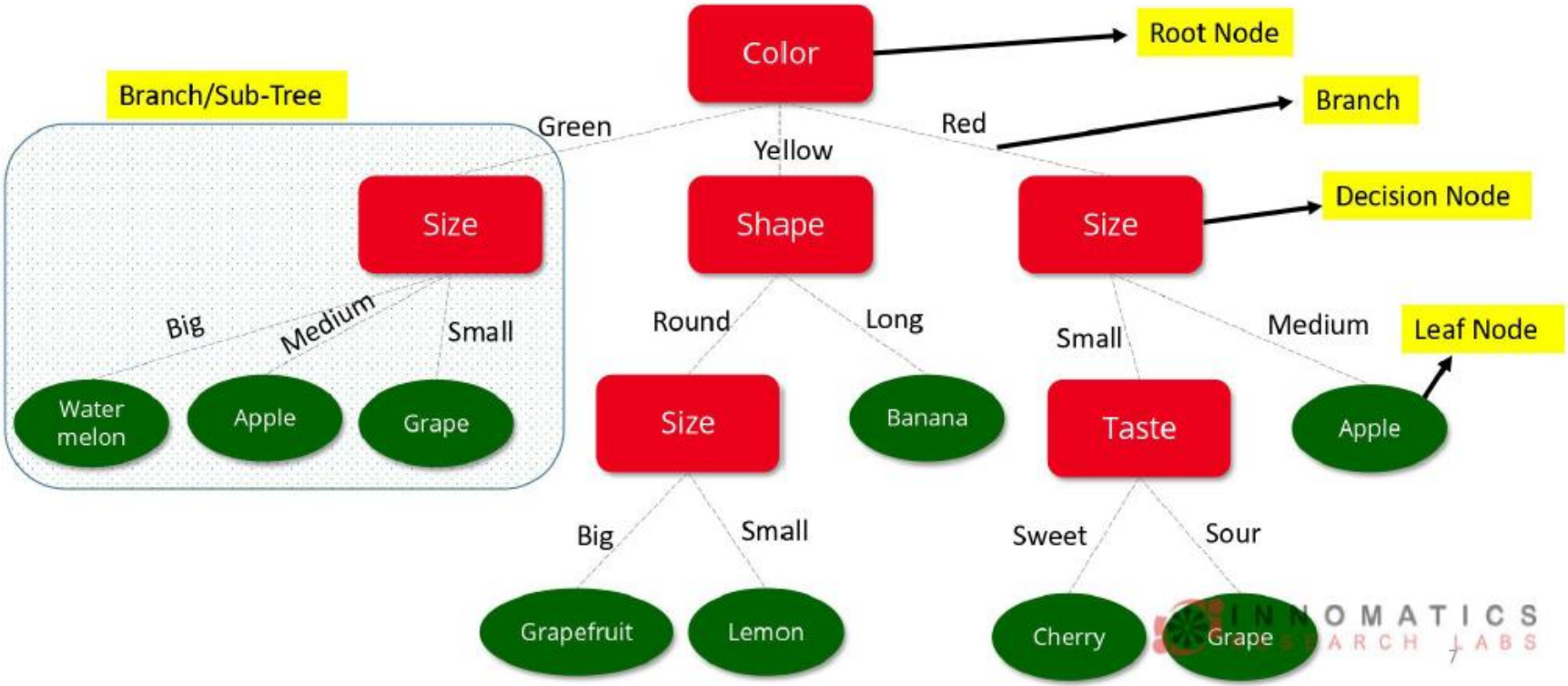
DECISION TREE - EXAMPLE

Color	Size	Shape	Taste	Fruit
Green	Big			Watermelon
Green	Medium			Apple
Green	Small			Grape
Yellow	Big	Round		Grapefruit
Yellow	Small	Round		Lemon
Yellow	Medium	Long		Banana
Red	Small		Sweet	Cherry
Red	Small		Sour	Grape
Red	Medium			Apple

DECISION TREE - EXAMPLE

Color	Size	Shape	Taste	Fruit
Green	Big			Watermelon
Green	Medium			Apple
Green	Small			Grape
Yellow	Big	Round		Grapefruit
Yellow	Small	Round		Lemon
Yellow	Medium	Long		Banana
Red	Small		Sweet	Cherry
Red	Small		Sour	Grape
Red	Medium			Apple

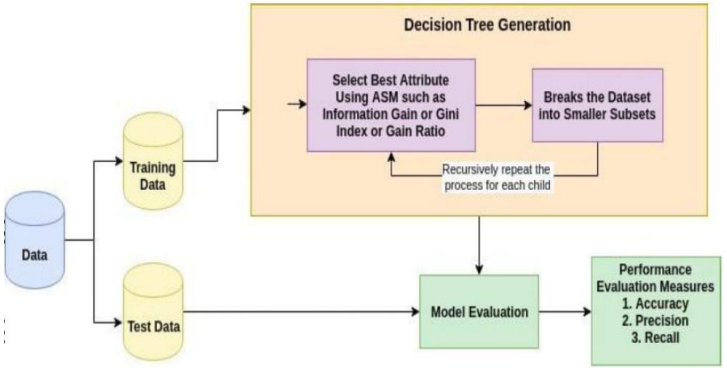
DECISION TREE - EXAMPLE



DECISION TREE INDUCTION

Steps to construct a Decision Tree:

- Select the best attribute using **Attribute Selection Measures(ASM)** to split the records.
- Make that attribute a root/decision node and break the dataset into smaller subsets.
- Start tree building by repeating this process recursively for each child leaf node is reached.



DECISION TREE INDUCTION

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, D.

Input:

- Data Partition, D, set of training tuples and their associated class labels;
- Attribute_list, the set of candidate attribute;
- Attribute_selection_method, a procedure to determine the splitting criterion that best partitions the dat atuples into individual classes.

Output: A decision Tree

Method:

(1) Create a node N;
(2) If tuples in D are all of the same class, C, then
(3) Return N as a leaf node labelled with the class C
(4) If attribute_list is empty then
(5) Return N as a leaf node labelled with the majority class in D // majority voting
(6) Apply Attribute_selection_method (D, attribute_list) to find the best splitting_criterion;
(7) Label Node N with splitting_criterion;
(8) If splitting_attribute is discrete-valued and multiway splits allowed then
(9) Attribute_list = attribute_list - splitting attribute;
(10) For each outcome j of splitting criterion
// partition the tuples and grow subtrees for each partition
(1) Let D_j be the set of data tuples in D satisfying outcome j (partition)
(2) If D_j is empty then
(3) Attach a leaf labelled with the majority class in D to node N
(4) Else Attach a node returned by **Generate_decision_tree(D_j , attribute_list)** to node N;
Endfor
(1) Return N;

ATTRIBUTE SELECTION MEASURES

- Used for selection of the splitting criterion that partitions the data into best possible manner
- Provides rank to each attribute or feature by explaining the given dataset
- Best score attribute will be selected as a splitting attribute
- Most popular selection measures are:
  - Information Gain, Entropy
  - Gini Index

ENTROPY

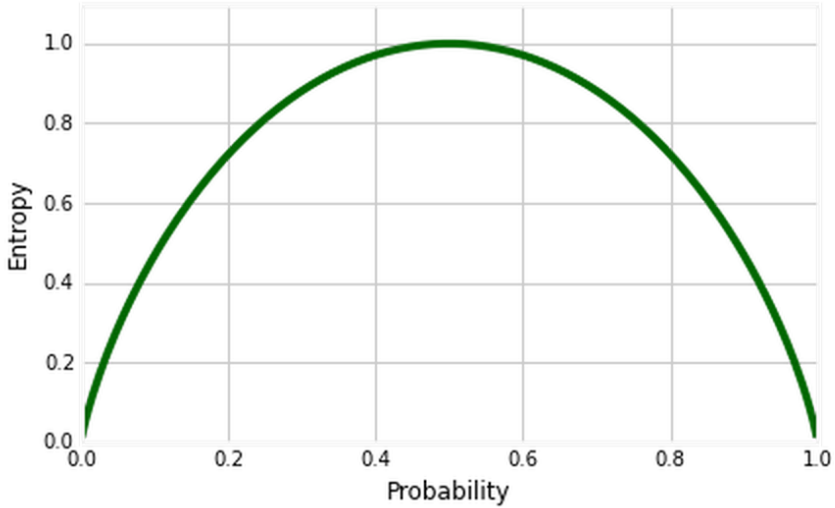
- **Entropy** is the amount of information disorder or simply said is the amount of randomness in the data or uncertainty.
- The entropy of a dataset depends on how much randomness is in the node.
- It should be noted that the **lower the entropy** the less uniform the distribution and the **purer the node**.
- If a sample is completely homogenous then the entropy is completely zero and if a sample is equally divided it has an entropy of 1.
- Entropy is calculated as:

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

- Where 'c' is the number of classes.

ENTROPY

- If the sample is completely homogeneous, the entropy is 0 (prob= 0 or 1), and if the sample is evenly distributed across classes, it has an entropy of 1 (prob =0.5).
- The next step is to make splits that minimize entropy. We use ***information gain*** to determine the best split.



INFORMATION GAIN

- Information gain (IG) measures how much “information” a feature gives us about the class.
- While using Information Gain as a criterion, we assume attributes to be categorical.
- The ID3/C4.5 algorithms uses entropy to calculate the homogeneity of a sample.
- The choice of the best tree depends on the node with the highest Information Gain after splitting.
- Information Gain is calculated as:

IG = Entropy(S) – {Weighted Avg. x Entropy(each future)}

EXAMPLE

Day	Temperature	Outlook	Humidity	Windy	Play Golf?
07-05	hot	sunny	high	false	no
07-06	hot	sunny	high	true	no
07-07	hot	overcast	high	false	yes
07-09	cool	rain	normal	false	yes
07-10	cool	overcast	normal	true	yes
07-12	mild	sunny	high	false	no
07-14	cool	sunny	normal	false	yes
07-15	mild	rain	normal	false	yes
07-20	mild	sunny	normal	true	yes
07-21	mild	overcast	high	true	yes
07-22	hot	overcast	normal	false	yes
07-23	mild	rain	high	true	no
07-26	cool	rain	normal	true	no
07-30	mild	rain	high	false	yes

today	cool	sunny	normal	false	?
tomorrow	mild	sunny	normal	false	?

Root Node Calculation

Outlook Variable

	Yes	No	Total
Sunny	2	3	5
Rain	3	2	5
Overcast	4	0	4
Total	**9**	**5**	14

Play Golf Variable

Yes	No
9	5

Temperature Variable

	Yes	No	**Total**
Hot	2	2	**4**
Mild	4	2	**6**
Cool	3	1	**4**
Total	9	5	**14**

Humidity Variable

	Yes	No	Total
High	3	4	7
Normal	6	1	7
Total	**9**	**5**	14

Wind Variable

	Yes	No	Total
Weak	6	2	8
Strong	3	3	6
Total	**9**	**5**	14

Entropy Calculation for all cases

Entropy = $-\sum_{i=1}^{J} p_i \log_2 p_i$

Step 1: Calculate entropy for all cases:

Yes = 9 No = 5 Total = 14

E(Play Golf) = $-\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$ = **0.940**

Entropy Calculation for Outlook

Step 2: Loop over all attributes, calculate gain: –

“Attribute = Outlook”

Outlook = Sunny

Yes = 2, No = 3, Total = 5

E(Sunny) = $-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$ = **0.971**

Outlook = Overcast

Yes = 4, No = 0, Total = 4

E(Sunny) = $-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}$ = **0.00**

Outlook = Rain

Yes = 3, No = 2, Total = 5

E(Sunny) = $-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$ = **0.971**

Calculate Information Gain for attribute Outlook

Gain(Play Golf, Outlook) = E(Play Golf) – (#Sunny/Total) * E(Sunny)
- (#Over/Total) * E(Overcast)
- (#Rain/Total) * E(Rainy)

Gain(Play Golf, Outlook) = **0.94 - (5/14)*0.971 - (4/14)*0 - (5/14)*0.971**

Gain(Play Golf, Outlook) = **0.25**

Entropy Calculation for Temperature

“Attribute = Temperature”

(Repeat process looping over {Hot, Mild, Cool})

Temperature = Hot

Yes = 2, No = 2, Total = 4

E(Sunny) = $-\frac{2}{4}\log_2\frac{2}{4}-\frac{2}{4}\log_2\frac{2}{4}$ = **1**

Temperature = Mild

Yes = 4, No = 2, Total = 6

E(Sunny) = $-\frac{4}{6}\log_2\frac{4}{6}-\frac{2}{6}\log_2\frac{2}{6}$ = **0.918**

Temperature = Cool

Yes = 3, No = 1, Total = 4

E(Sunny) = $-\frac{3}{4}\log_2\frac{3}{4}-\frac{1}{4}\log_2\frac{1}{4}$ = **0.811**

Calculate Information Gain for attribute Temperature

Gain(Play Golf, Temperature) = E(Play Golf) – #Hot/Total*E(Hot)
- #Mild/Total*E(Mild)
- #Cool/Total*E(Cool)

Gain(Play Golf, Temperature) = **0.94 - (4/14)*1 - (6/14)* 0.918 - (4/14)* 0.811**

Gain(Play Golf, Temperature) = **0.03**

Information Gain for Humidity & Wind

– Attribute = Humidity

(Repeat process looping over {High, Normal})

Gain(Play Golf, Humidity) = **0.15**

– Attribute = Wind

(Repeat process looping over {Weak, Strong})

Gain(Play Golf, Wind) = **0.05**

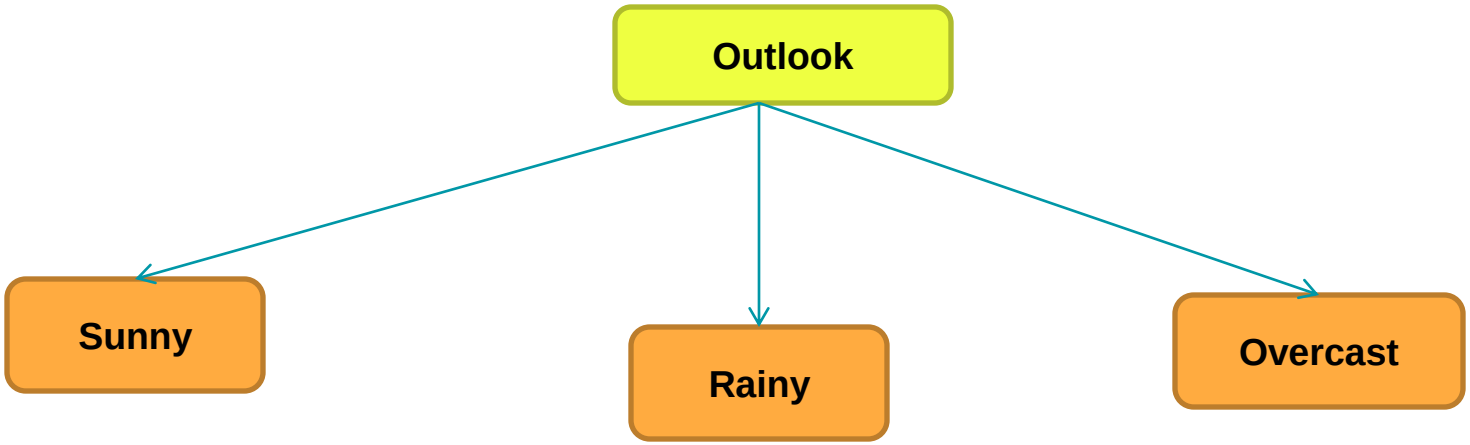
Find attribute with greatest information

information gain:

Gain(Play Golf, Outlook) = 0.25
Gain(Play Golf, Temperature) = 0.03
Gain(Play Golf, Humidity) = 0.15
Gain(Play Golf, Wind) = 0.05

Outlook is root node of tree

Root Node Construction



Outlook - Sunny

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Decision Node Calculation

Temp

	Yes	No	Total
Hot	0	2	2
Mild	1	1	2
Cool	1	0	1
Total	2	3	5

Humidity

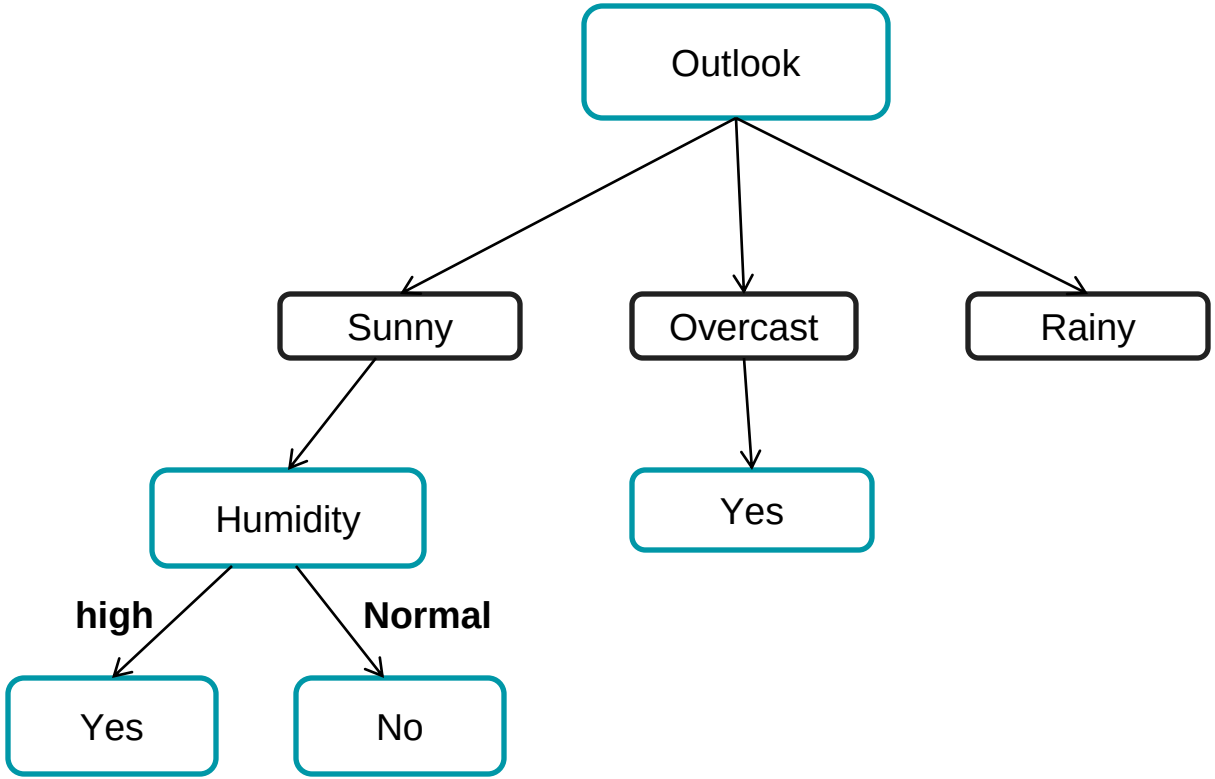
	Yes	No	Total
High	0	3	3
Normal	2	0	2
Total	2	3	5

Wind

	Yes	No	Total
Weak	1	2	3
Strong	1	1	2
Total	2	3	5

Gain(Play Golf, Temperature)
Gain(Play Golf, Humidity)----------
→ High, randomness is low
Gain(Play Golf, Wind)

Decision Tree Construction



Outlook - Rainy

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Leaf Node Calculations

Temperature

	Yes	No	Total
Hot	0	0	0
Mild	2	1	3
Cool	1	1	2
Total	**3**	**2**	**5**

Humidity

	Yes	No	Total
High	1	1	2
Normal	2	1	3
Total	**3**	**2**	**5**

Wind

	Yes	No	Total
Weak	3	0	3
Strong	0	2	2
Total	**3**	**2**	**5**

Gain(Play Golf, Temperature)

Gain(Play Golf, Humidity)

Gain(Play Golf, Wind) ----------→ High, randomness is Low

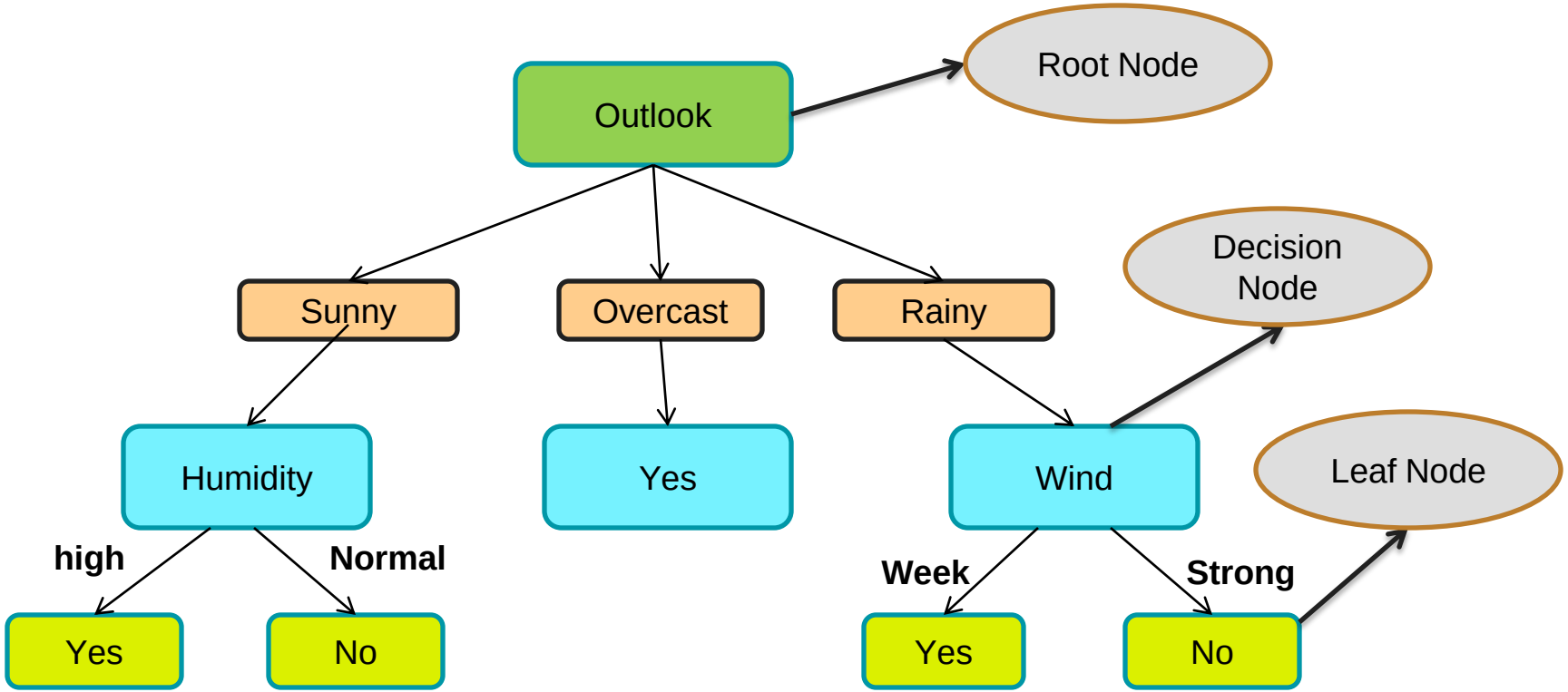
Rules:

- If **Outlook** is Sunny. Then we check Humidity
- If **Humidity** is High, then decision id No
- If **Humidity** is Normal, then decision is Yes

- If **Outlook** is Rainy, then we check Wind
- If **Wind** is Strong, then decision is No
- If **Wind** is Yes, then decision is Yes

- If **Outlook** is Overcast, then decision is Yes

Decision Tree Construction



GINI INDEX

- **Gini –** the most commonly used measurement of impurity.

$$\text{Gini} = 1 - \sum_{i=1}^{classes} p(i \mid t)^2$$

- Works with the categorical target variable “Success” or “Failure”. It performs only binary splits.
- Higher the value of Gini, higher the homogeneity. CART (Classification and Regression Tree) uses the Gini method to create binary splits.
- For gini index, attributes are assumed to be continuous.

Gini Index for Outlook

$$\text{Gini} = 1 - \sum_{i=1}^{classes} p(i \mid t)^2$$

Gini (Play Golf, Outlook = Sunny)
= 1 – (2/5)^2 – (3/5)^2 = 0.48
Gini (Play Golf, Outlook = Overcast)
= 1 – (4/4)^2 – (0/4)^2 = 0
Gini (Play Golf, Outlook = Rainy)
= 1 – (3/5)^2 – (2/5)^2 = 0.48

The Gini Index of Outlook
= 5/14 * 0.48 + 4/14 * 0 + 4/14 * 0.48 = **0.3429**

Gini Gain = Gini (Play Golf) – Gini (Outlook)
= [1 – (10/14)^2 – (4/14)^2] – 0.3429
= 0.4082 – 0.3429
= 0.065

Outlook Variable

	Yes	No	**Total**
Sunny	3	2	**5**
Overcast	4	0	**4**
Rainy	3	2	**5**
Total	**10**	**4**	

Gini Index for Humidity

Gini (Play Golf, Humidity = High)

= 1 – (3/7)^2 – (4/7)^2 = 0.4898

Gini (Play Golf, Humidity = Normal)

= 1 – (6/7)^2 – (1/7)^2 = 0.2449

The Gini Index of Humidity

= 7/14 * 0.4898 + 7/14 * 0.2449 = **0.3674**

Gini Gain = Gini (Play Golf) – Gini (Humidity)

= [1 – (9/14)^2 – (5/14)^2] – 0.3674

= 0.4591 – 0.3674

= 0.0917

Humidity Variable

	Yes	No	Total
High	3	4	7
Normal	6	1	7
Total	**9**	**5**	14

Gini Index for Temperature

Gini (Play Golf, Temperature = Hot)

= 1 – (2/4)^2 – (2/4)^2 = 0.5

Gini (Play Golf, Temperature = Mild)

= 1 – (4/6)^2 – (2/6)^2 = 0.444

Gini (Play Golf, Temperature = Cool)

= 1 – (3/4)^2 – (1/4)^2 = 0.375

The Gini Index of Humidity

= 4/14 * 0.5 + 6/14 * 0.444 + 4/14 * 0.375 = **0.44**

Temperature Variable

	Yes	No	**Total**
Hot	2	2	**4**
Mild	4	2	**6**
Cool	3	1	**4**
Total	**9**	**5**	**14**

Information Gain for Temperature

Gini Gain = Gini (Play Golf) – Gini (Temperature)

= [1 – (9/14)^2 – (5/14)^2] - 0.3674

= 0.4591 – 0.444

= 0.0151

Gini Index for Wind

Gini (Play Golf, Wind = Week)

= 1 – (6/8)^2 – (2/8)^2 = 0.375

Gini (Play Golf, Wind = Strong)

= 1 – (3/6)^2 – (3/6)^2 = 0.5

The Gini Index of Wind

= 8/14 * 0.375 + 6/14 * 0.5 = **0.4285**

Gini Gain = Gini (Play Golf) – Gini (Wind)

= [1 – (9/14)^2 – (5/14)^2] – 0.3674

= 0.4591 – 0.4285

= 0.0306

Wind Variable

	Yes	No	Total
Weak	6	2	8
Strong	3	3	6
Total	**9**	**5**	14

Conclusion with Gini Index

information gain:

Gain(Play Golf, Outlook) = 0.065

Gain(Play Golf, Temperature) = 0.0151

Gain(Play Golf, Humidity) = 0.0917

Gain(Play Golf, Wind) = 0.0306

Humidity is root node of tree

Issues in Decision Tree Learning

- How deeply to grow the decision tree?
- Handling continuous attributes
- Choosing an appropriate attribute selection measure
- Handling training data with missing attribute values

Overfitting in Decision Trees

- If a decision tree is fully grown, it may lose some generalization capability.
- This is a phenomenon known as overfitting.

Overfitting

> “A hypothesis overfits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances (i.e. including instances beyond the training examples)”

Causes of Overfitting

- **Overfitting Due to Presence of Noise -** Mislabeled instances may contradict the class labels of other similar records.

- **Overfitting Due to Lack of Representative Instances -** Lack of representative instances in the training data can prevent refinement of the learning algorithm.

“A good model must not only fit the training data well but also accurately classify records it has never seen.”

Avoiding overfitting in decision tree

- approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data.
- approaches that allow the tree to overfit the data, and then post-prune the tree.

Approaches to implement

- separate set of examples
- a statistical test: chi-square test
- a heuristic called the Minimum Description Length principle - (explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized.)

TREE PRUNING

A too-large tree increases the risk of over fitting, and a small tree may not capture all the important features of the dataset. Pruning helps us avoid over fitting.

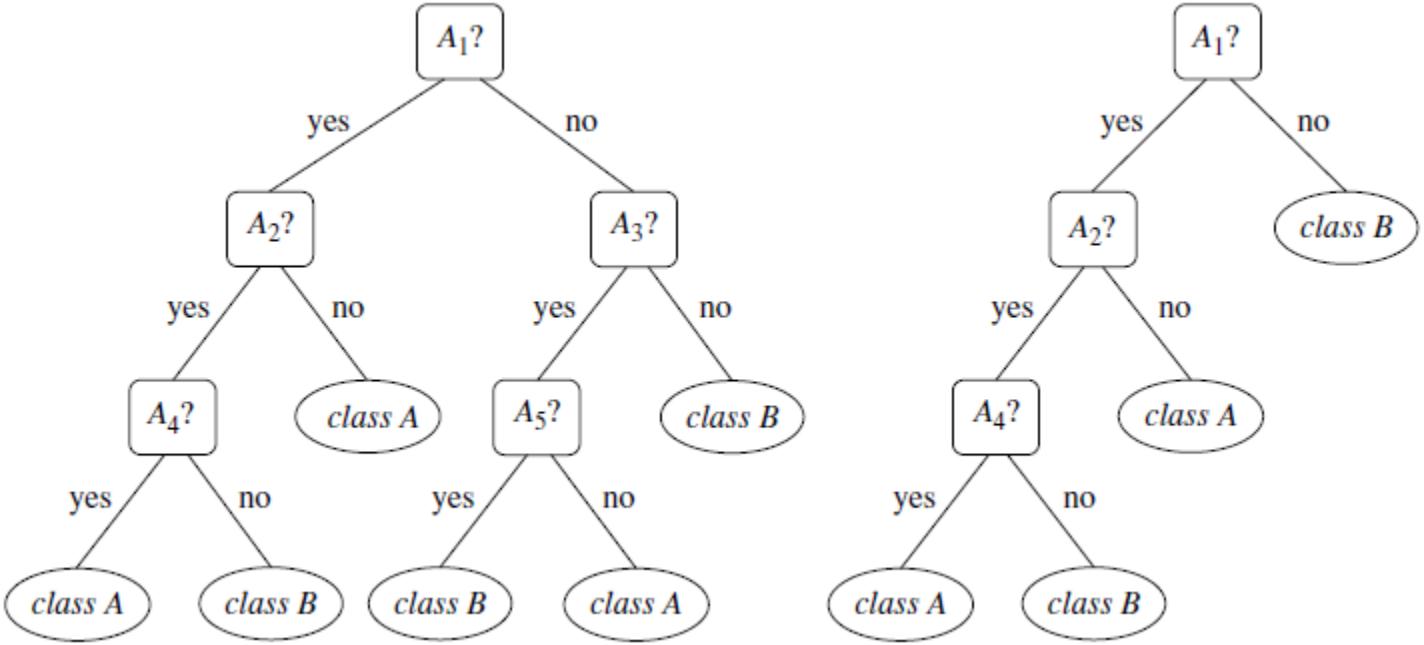
Pre pruning:

Halt the construction of tree early. The current node will become a leaf node with class label of the majority class in it's dataset.

Post pruning:

Remove sub trees from a fully grown tree. e.g. in CART algorithm**, cost complexity** algorithm considers the cost complexity of a tree to be a function of number of leaves in the tree and the error rate (misclassified tuples). It starts from the bottom of the tree. If pruning the sub tree at node N results in a smaller cost complexity, then the sub tree is pruned.

TREE PRUNING



Pruning to Avoid Over fitting

• Growing the tree beyond a certain level of complexity leads to overfitting.
• Pruning helps us to avoid overfitting
• Any additional split that does not add significant value is not required in the model.
• We can avoid overfitting by changing the parameters like -

You can use pruning parameters like:

- **max_leaf_nodes**: reduce the number of leaf nodes
- **min_samples_leaf**: restrict the sample size in the terminal node that supports that decision
- **max_depth**: reduce the depth of the tree to build a generalized tree

Pros & Cons

Pros

- Decision Trees are easy to explain. It is simply a set of rules
- The interpretation can be visualized

Cons

- There is a high probability of over fitting in Decision Tree.
- Information gain in a decision tree with categorical variables gives a biased response for attributes with greater no. of categories.
- Calculations can become too complex when there are too many class labels

THAT'S ALL FOLKS