DecisionTree

-MUKESH KUMAR

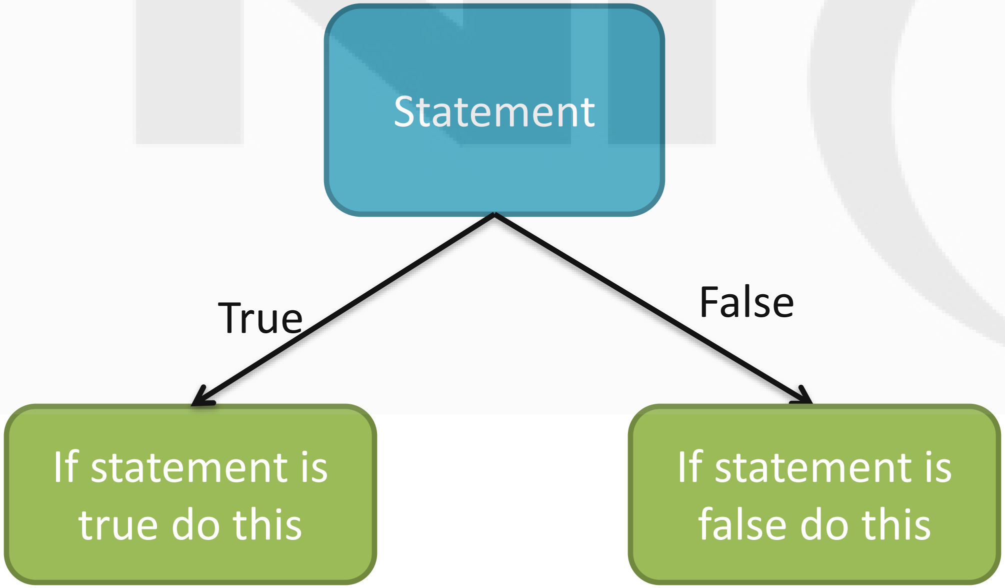
- A **Decision Tree** is a supervised machine learning algorithm used for both classification and regression tasks.

- It splits the data into subsets based on the value of input features, creating a tree-like model of decisions.

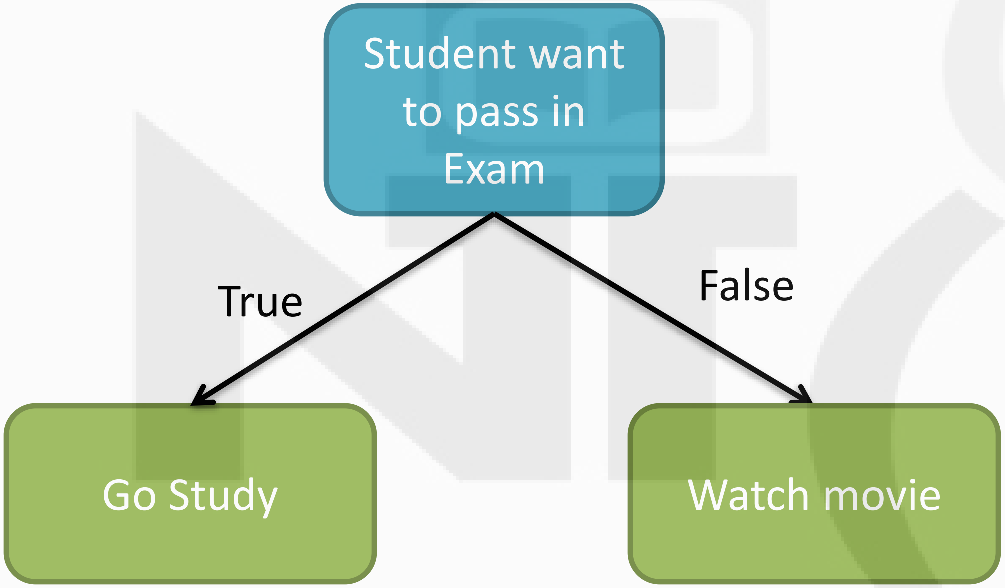
Simplest Decision Tree

- In general a Decision Tree makes a statement and then makes a decision based on whether the statement is true or false

- By default node on the **left** represents **True** condition and the one on the **right** is for **False** condition



Simplest Decision Tree



Student want to pass in Exam

True

False

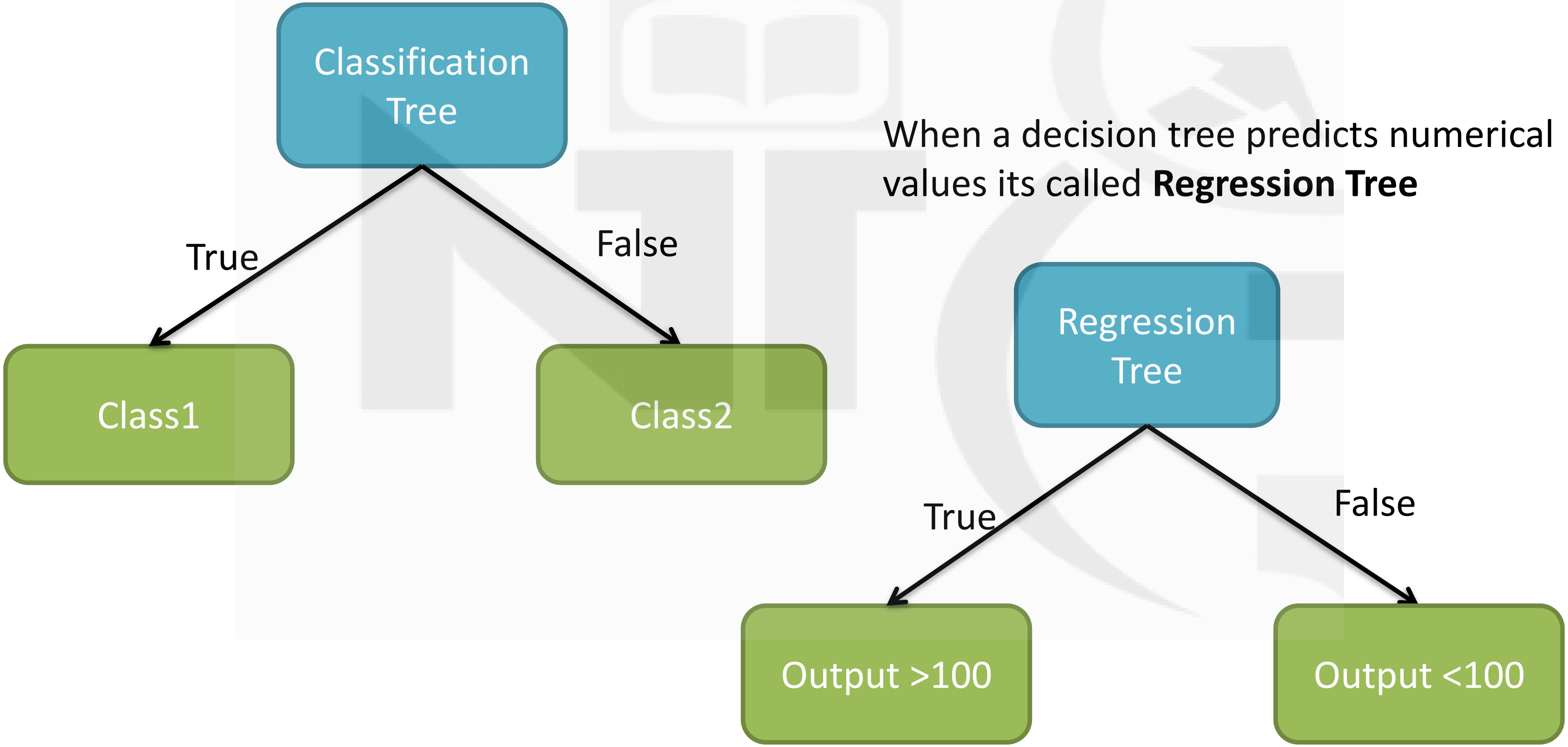
Go Study

Watch movie

Classification Vs Regression Tree

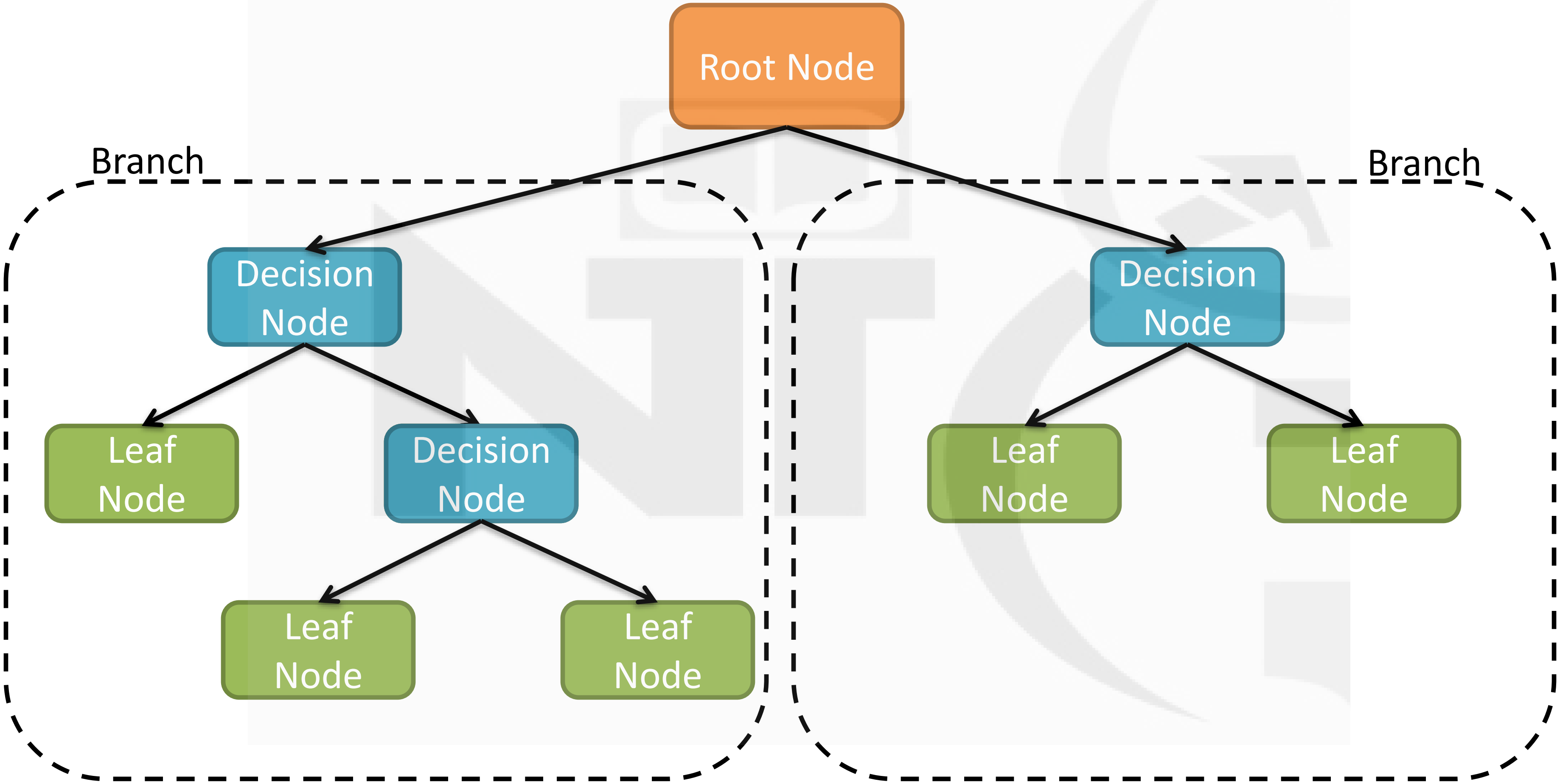
When a decision tree classifies something into categories it **Classification Tree**

When a decision tree predicts numerical values its called **Regression Tree**



Tree Structure

- **Root Node**: The top node where the first decision is made.
- **Internal Nodes**: Nodes where decisions based on features are made to split data into subsets.
- **Branches**: Connects nodes and represents decision paths based on feature values.
- **Leaf Nodes**: Terminal nodes where no further splits are made, and a prediction is given.



Root Node

Branch

Branch

Decision Node

Decision Node

Leaf Node

Decision Node

Leaf Node

Leaf Node

Leaf Node

Leaf Node

HOW DECISION TREE WORKS

1.Start with the Entire Dataset (Root Node)

- The tree-building process begins with the entire dataset. This dataset is represented as the root node, which contains all the training samples.

2.Choose the Best Split (Based on a Criterion)

- The algorithm evaluates all possible splits of the data based on features.

- It uses a criterion (e.g., Gini Impurity, Information Gain, Entropy) to determine the best feature and threshold to split the data.

- The goal is to select the split that best separates the data into homogeneous groups (i.e., groups with similar target values).

3.Create Child Nodes (Recursive Splitting)

- Once a split is selected, the dataset is divided into two subsets. These subsets become the child nodes of the original (parent) node.
- The splitting process is recursively repeated for each child node, with the goal of further separating the data in each subset.

4.Stop Splitting (When a Stopping Condition is Met)

- The tree continues to split until a stopping condition is reached. Common stopping criteria include:
  - All data in the node belongs to the same class (purity).
  - A predefined depth of the tree is reached.
  - The node contains too few samples to split further.
  - There is no significant gain from further splitting.
- When splitting stops, the node becomes a **leaf node**.

5.Assign Labels or Values to Leaf Nodes

- In classification tasks, the leaf node is assigned the most common class among the samples in that node.
- In regression tasks, the leaf node is assigned the mean (or median) of the target values in that node.

6.Prediction with the Decision Tree

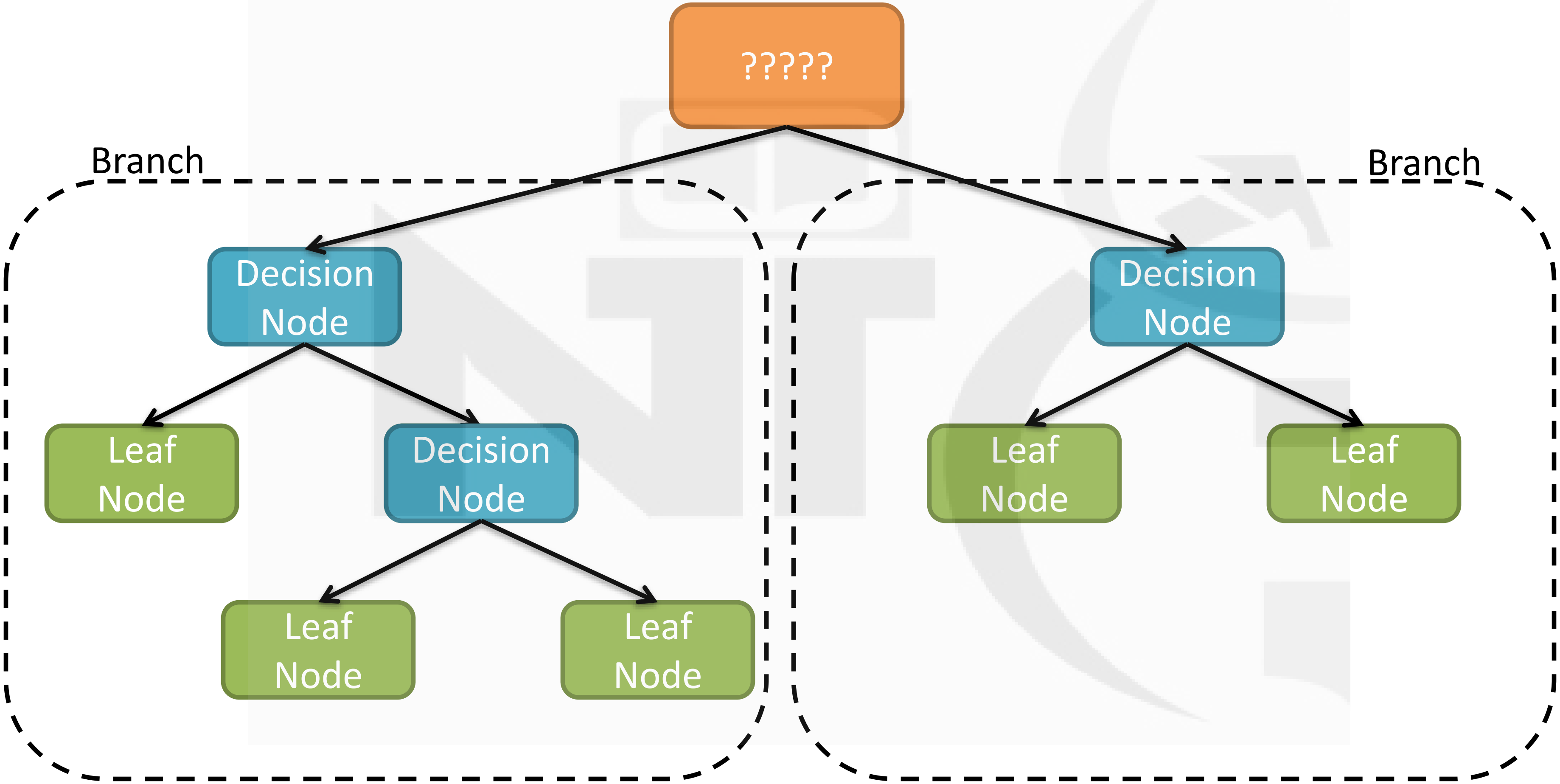
- To make predictions, the decision tree traverses from the root to a leaf node based on the input features.
- Each internal node represents a decision (based on a feature), and the model follows the path based on the input values.
- Once a leaf node is reached, the model outputs the assigned class (for classification) or value (for regression).

Example

Loves Action Genre	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

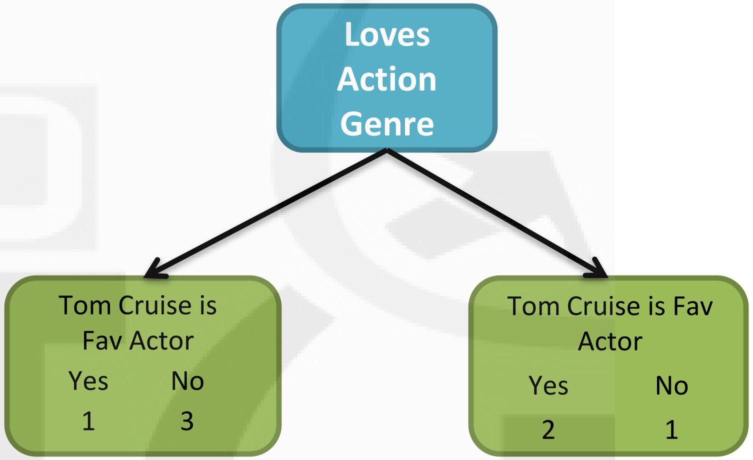
Step1: Decide the feature at the root node

(**Loves Action Gerne Has Watched Top Gun Age**)



- Lets see how well “**Loves Action Genre**“ predicts the output(**Tom Cruise is Fav Actor**).
- Lets build a simple tree with only this feature.

Loves Action Genre	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Loves Action Genre

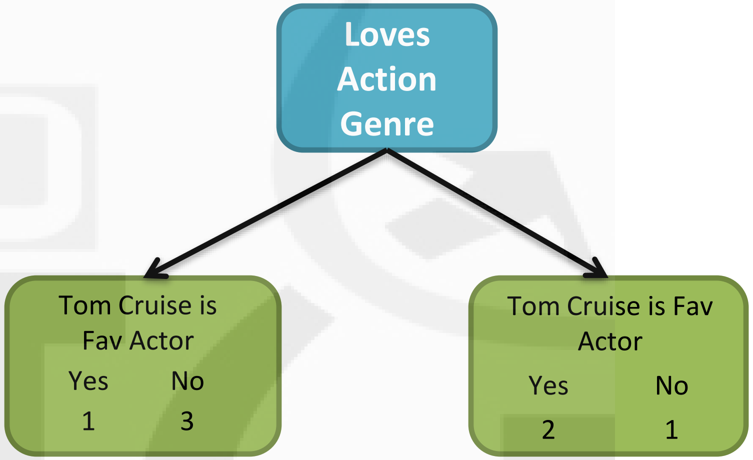
Tom Cruise is Fav Actor

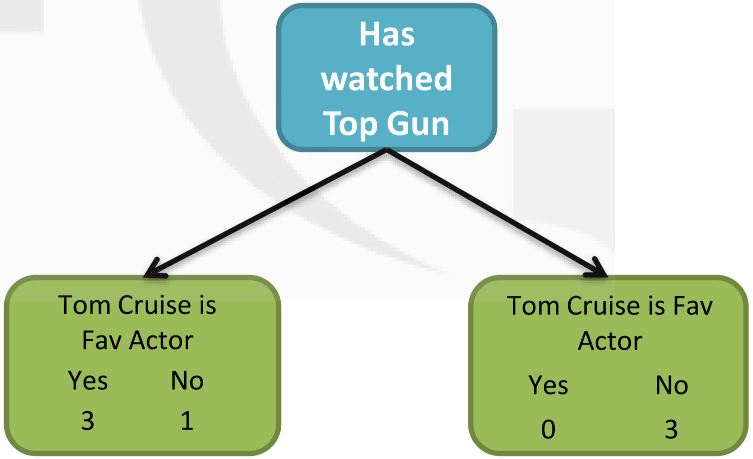
Yes	No
1	3

Tom Cruise is Fav Actor

Yes	No
2	1

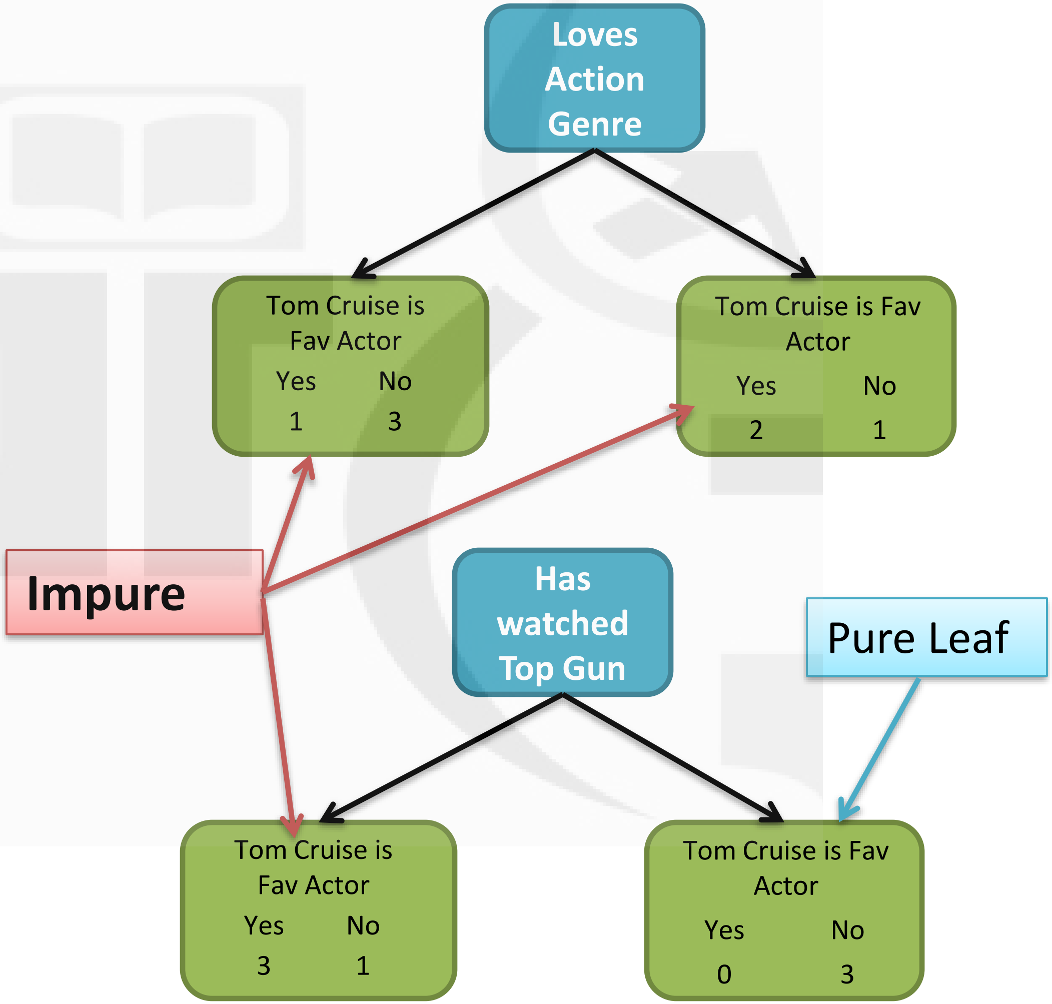
Loves Action Genre	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No





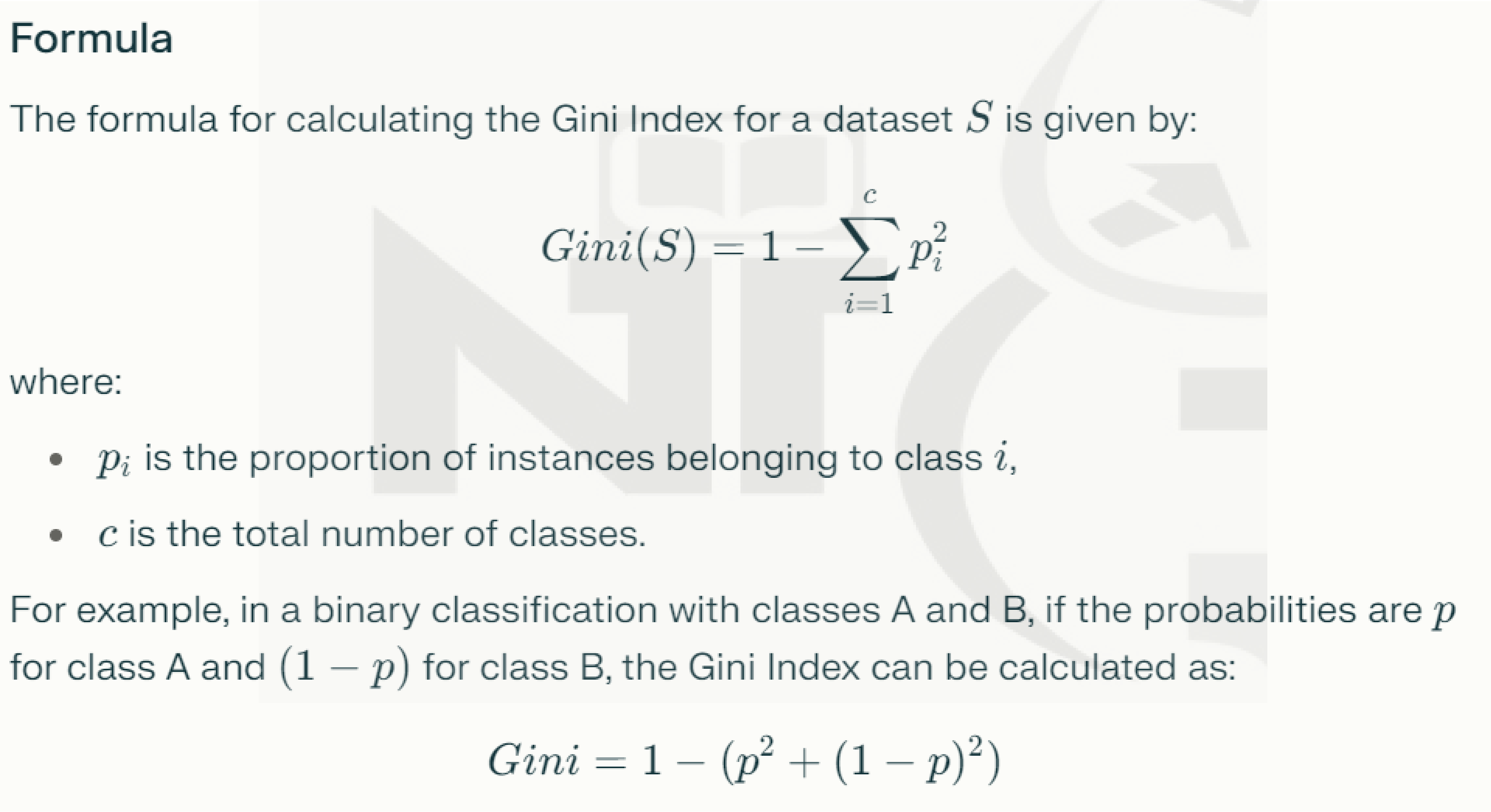
- Both the trees fail to perfectly classify the output.
- Impure nodes contain mixture of output
- Pure leaf contains only one type of output

- **Because there is a pure leaf in second tree it does better job of classifying output**



There are many ways to measure the impurity of leaves:

- **Entropy**
- **Gini Impurity**
- **Information Gain**
- All these method are similar and Gini is the most popularly used method



Formula

The formula for calculating the Gini Index for a dataset S is given by:

$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2$$

where:

- p_i is the proportion of instances belonging to class i,
- c is the total number of classes.

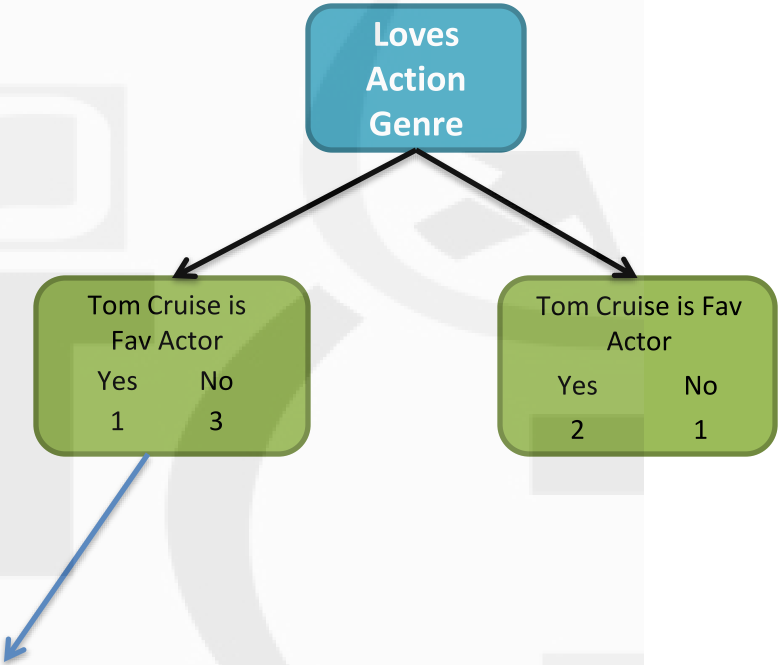
For example, in a binary classification with classes A and B, if the probabilities are p for class A and $(1 - p)$ for class B, the Gini Index can be calculated as:

$$Gini = 1 - (p^2 + (1 - p)^2)$$

- Now lets start with gini index calculation of each of the features one by one.

Gini index for “Loves Action Genre” = YES

Loves Action Genre	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



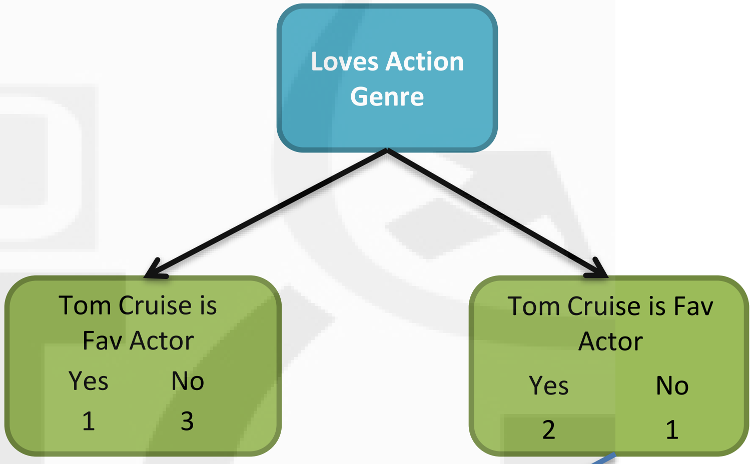
Gini Impurity for a Leaf = 1 - (the probability of “Yes”)^2 - (the probability of “No”)^2

$$= 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2$$

= 0.375

Gini index for “Loves Action Genre” = No

Loves Action Genre	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



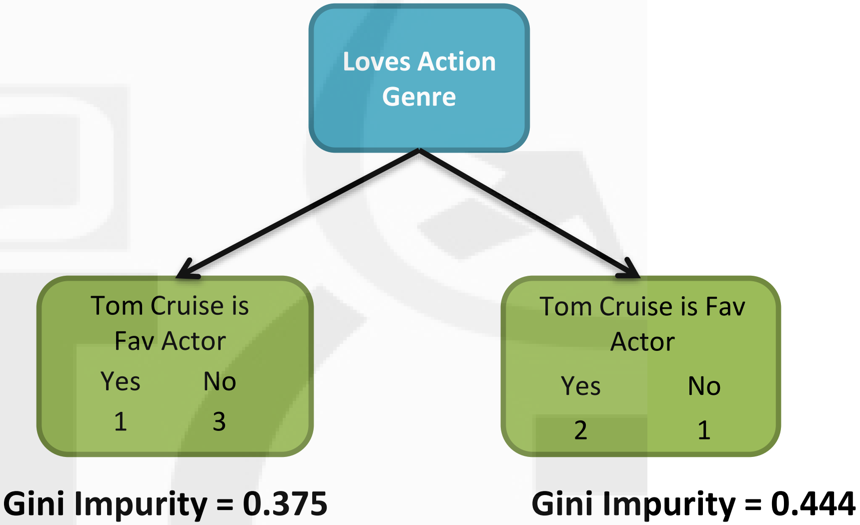
Gini Impurity = 0.375

Gini Impurity for a Leaf = 1 - (the probability of “Yes”)^2 - (the probability of “No”)^2

$$= 1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2$$

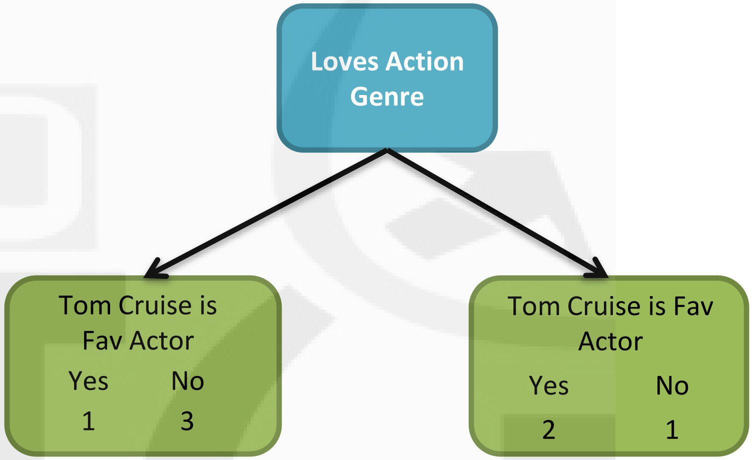
= 0.444

- As the number of ppl are different in both leaves :
  - Left leave has 4
  - Right leave has 3
- So we need to find the weighted average Gini index.
- We need to calculate the weight of both the leaves



- Total Gini Index = weighted Avg Gini Impurities of leaves

- **Gini index for “Loves Action Genre is 0.405**



Loves Action Genre

Tom Cruise is Fav Actor

Yes	No
1	3

Tom Cruise is Fav Actor

Yes	No
2	1

Gini Impurity = 0.375

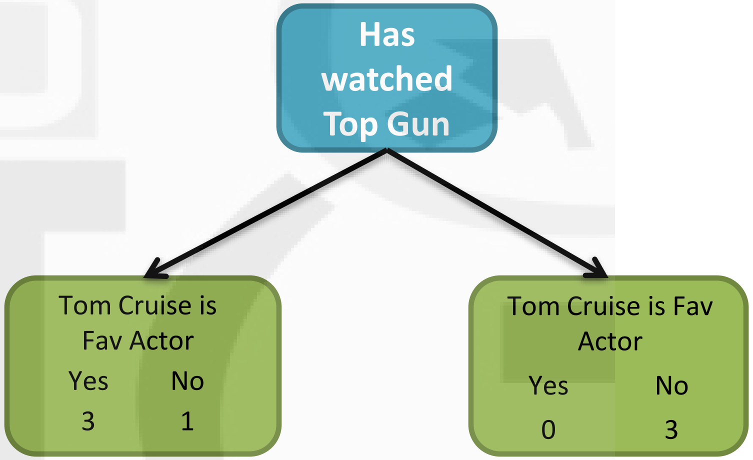
Gini Impurity = 0.444

$$= \left(\frac{4}{4+3}\right)0.375 + \left(\frac{3}{4+3}\right)0.444$$

=0.405

Gini Index for “Has Watched Top Gun”

- **Gini Impurity for “Has Watched Top Gun = 0.214**



Gini Index for Age

Loves Action Genre	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

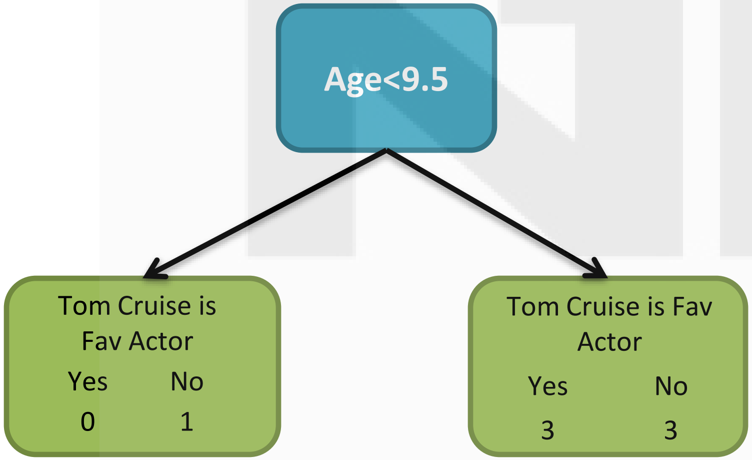
Gini Index for Age

- For numeric value calculation of Gini is different
- Sort in ascending
- Find the avg of adjacent values

Age	Tom Cruise is Fav Actor
7	No
9.5	
12	No
15	
18	Yes
26.5	
35	Yes
36.5	
38	Yes
44	
50	No
66.5	
83	No

Gini Index for Age

- We calculate the Gini Index for each of these avg ages one by one



$$= 1 - \left(\frac{0}{0+1}\right)^2 - \left(\frac{1}{0+1}\right)^2$$

=0

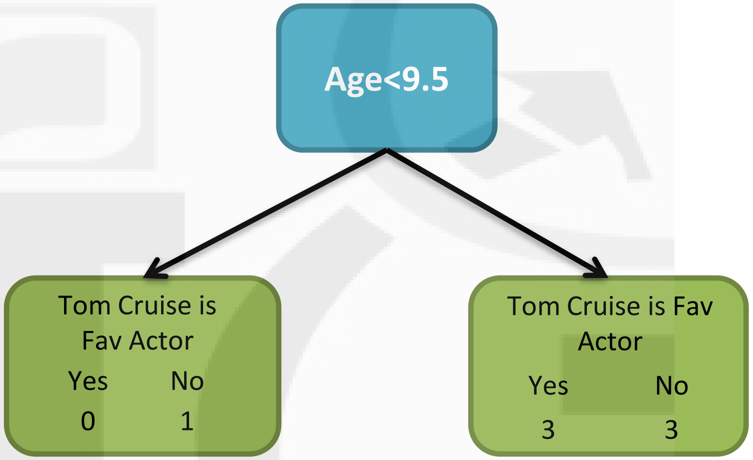
$$= 1 - \left(\frac{3}{3+3}\right)^2 - \left(\frac{3}{3+3}\right)^2$$

=0.5

Age	Tom Cruise is Fav Actor
7	No
9.5	
12	No
15	
18	Yes
26.5	
35	Yes
36.5	
38	Yes
44	
50	No
66.5	
83	No

Gini Index of Age<9.5

- Total Gini index = weighted avg index = 0.429



$$= 1 - \left(\frac{0}{0+1}\right)^2 - \left(\frac{1}{0+1}\right)^2$$

=0

$$= 1 - \left(\frac{3}{3+3}\right)^2 - \left(\frac{3}{3+3}\right)^2$$

=0.5

- Similarly we calculate Gini index for each of the avg age as shown below:

- Age 15 and 44 has lowest GI
- We pick anyone of them

Age	Tom Cruise is Fav Actor
7	No
12	No
18	Yes
35	Yes
38	Yes
50	No
83	No

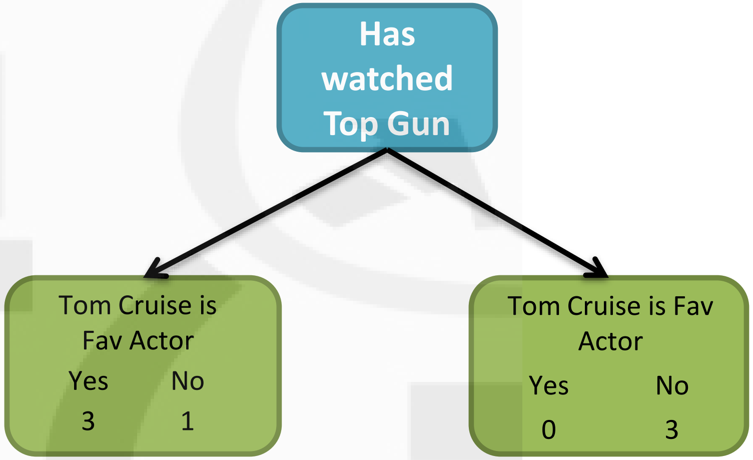
GI = 0.429	9.5
GI = 0.343	15
GI = 0.476	26.5
GI = 0.476	36.5
GI = 0.343	44
GI = 0.429	66.5

Final values of Gini Indices for all features

Since “Has Watched Top Gun” feature has the lowest Gini index,
It becomes the root node

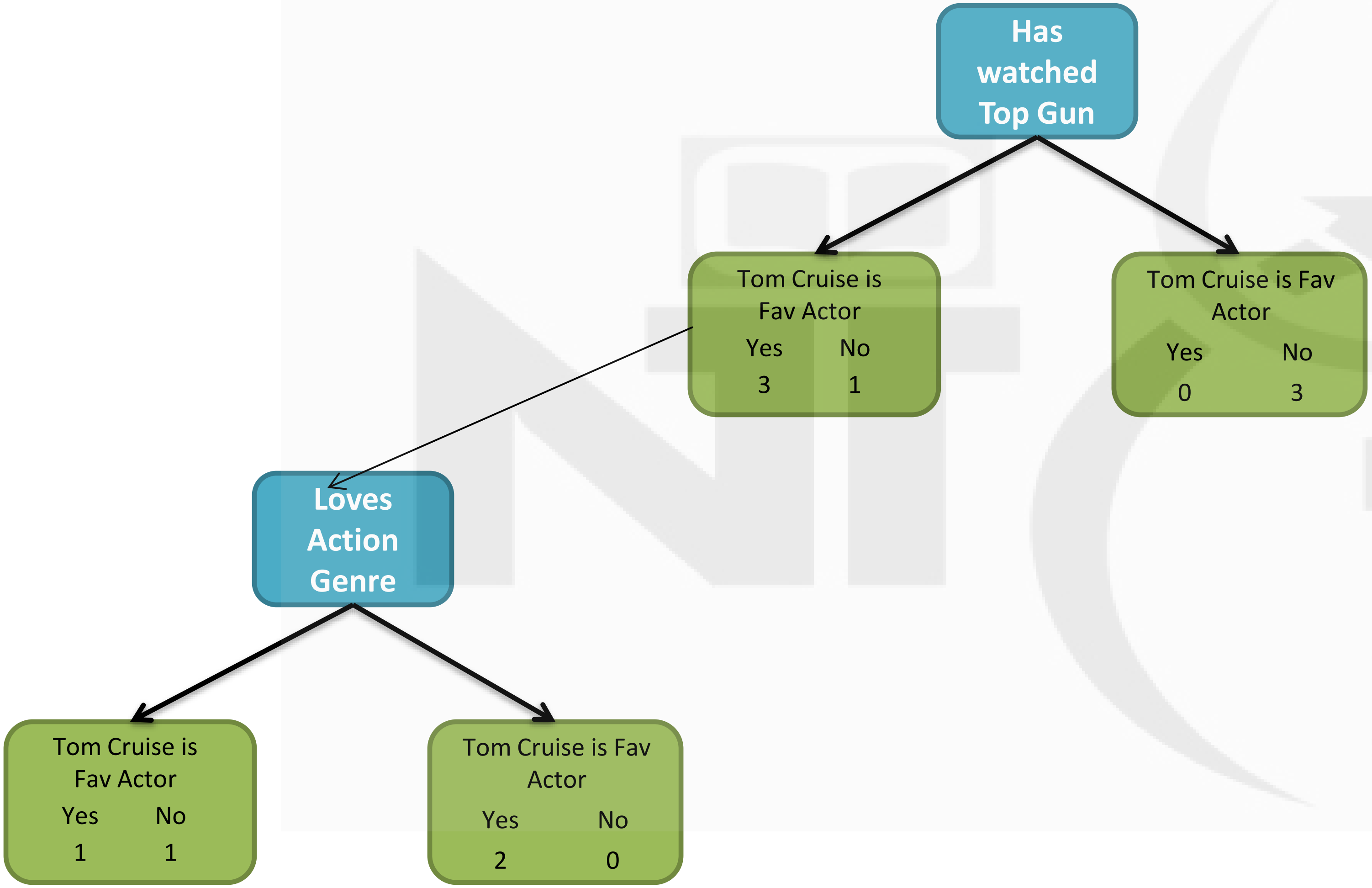
Loves Action Gerne	Has Watched Top Gun	Age
0.405	0.214	0.343

Loves Action Genre	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



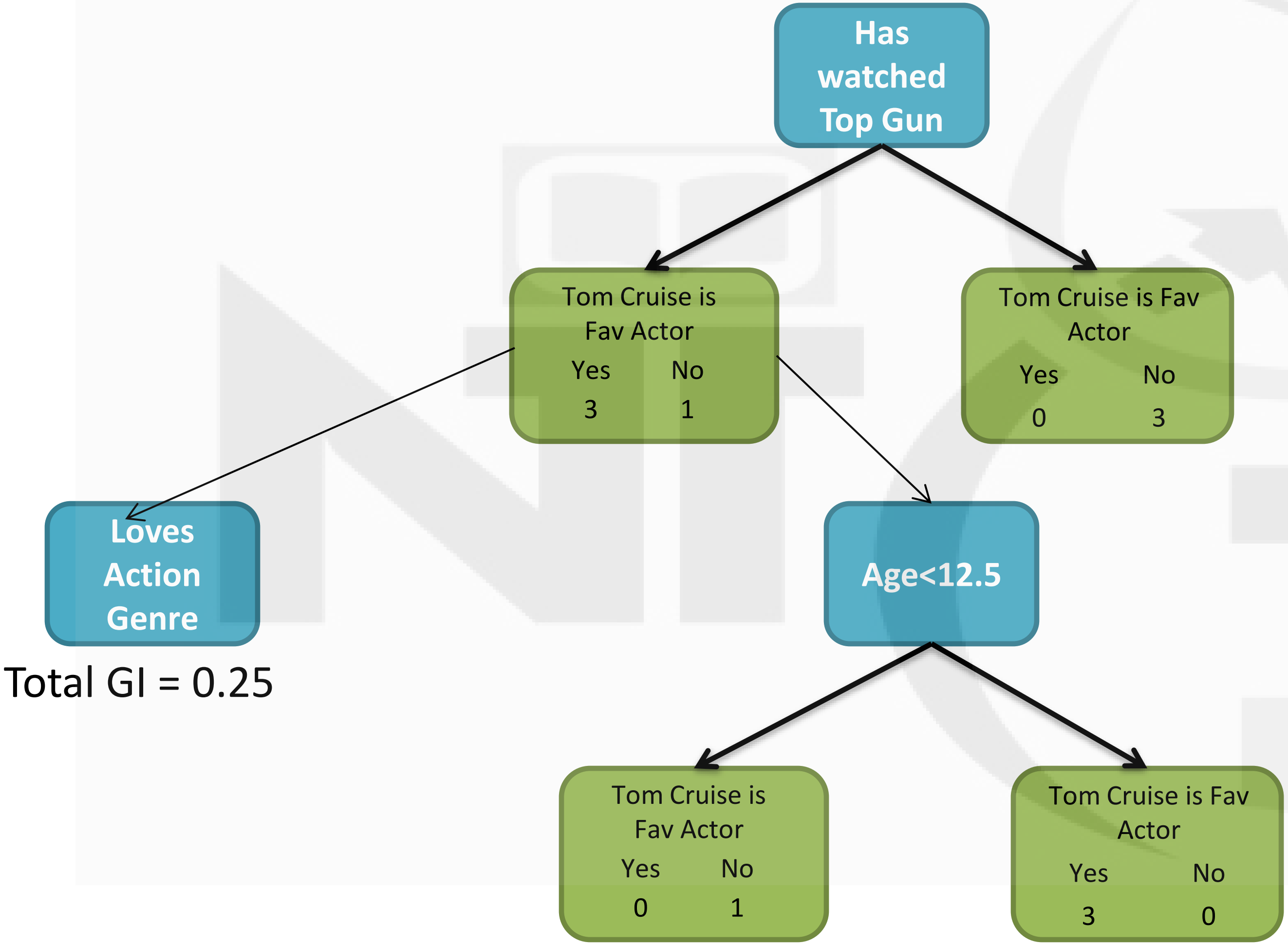
- Left node is impure , so we try to reduce the impurity by further spliting it

Left node can be splited with Age or LoveActionGenre feature, we again calculate the Gini index to decide the feature for split



Total GI = 0.25

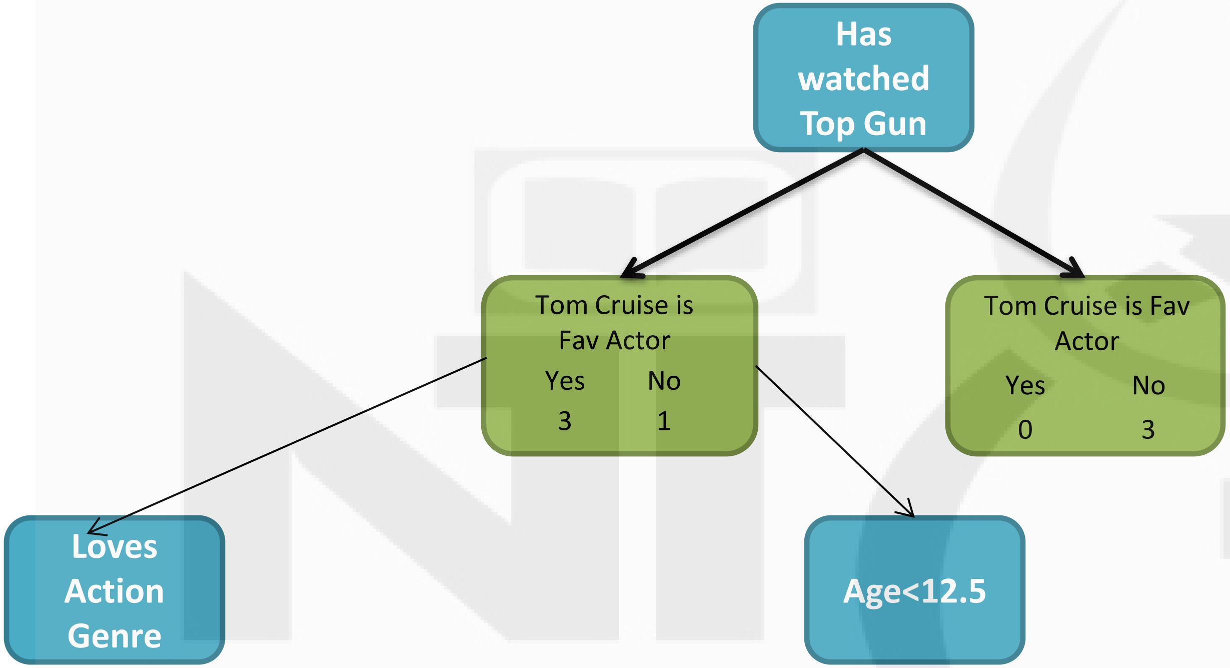
Lets try spliting on Age, Age<12.5 give the lowest gini index



Total GI = 0.25

Here in case of Age both the leaves are pure hence GI =0

GI =0 hence Age is selected for spliting

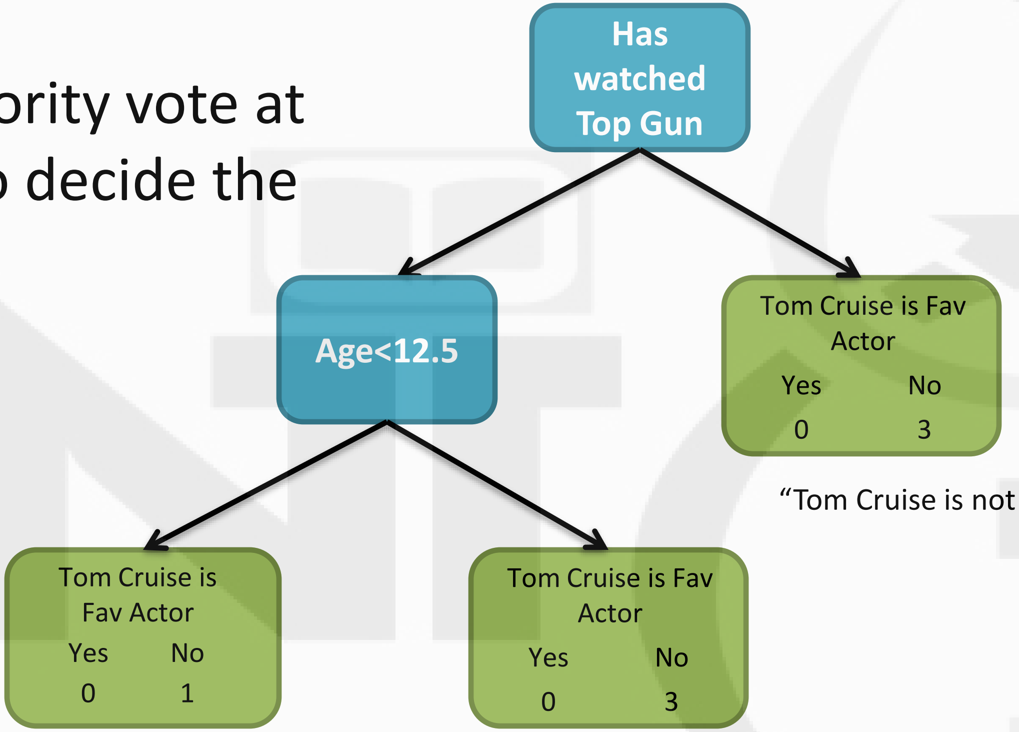


Total GI = 0.25

GI =0

Assigning output to the leaves

- We take majority vote at each leave to decide the output label.

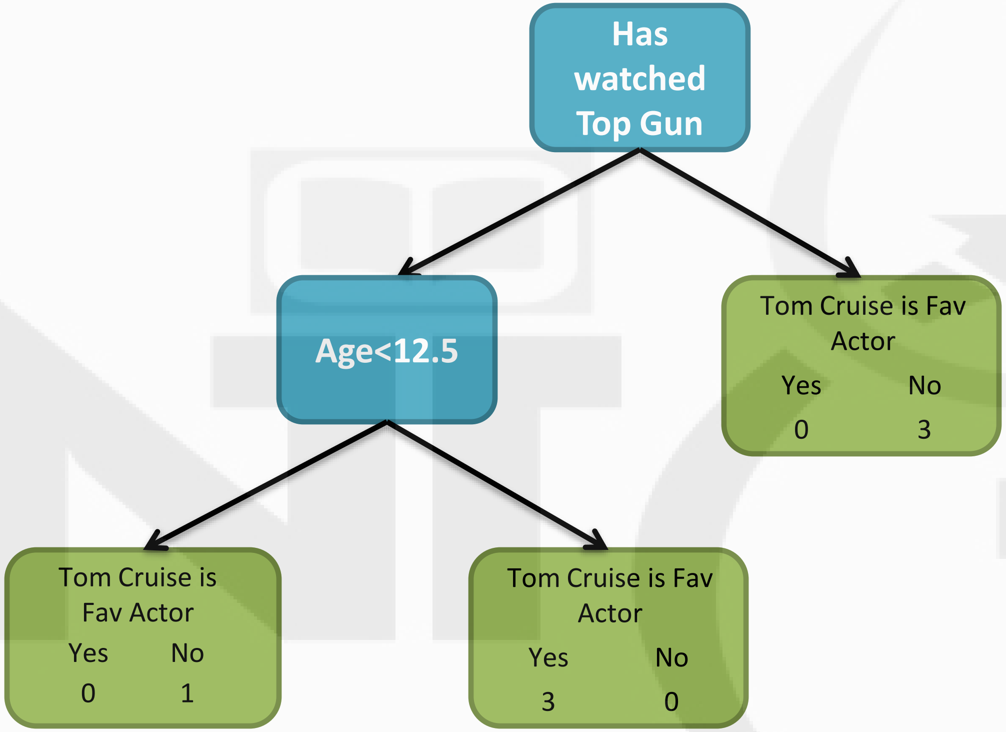


"Tom Cruise is not Fav Actor"

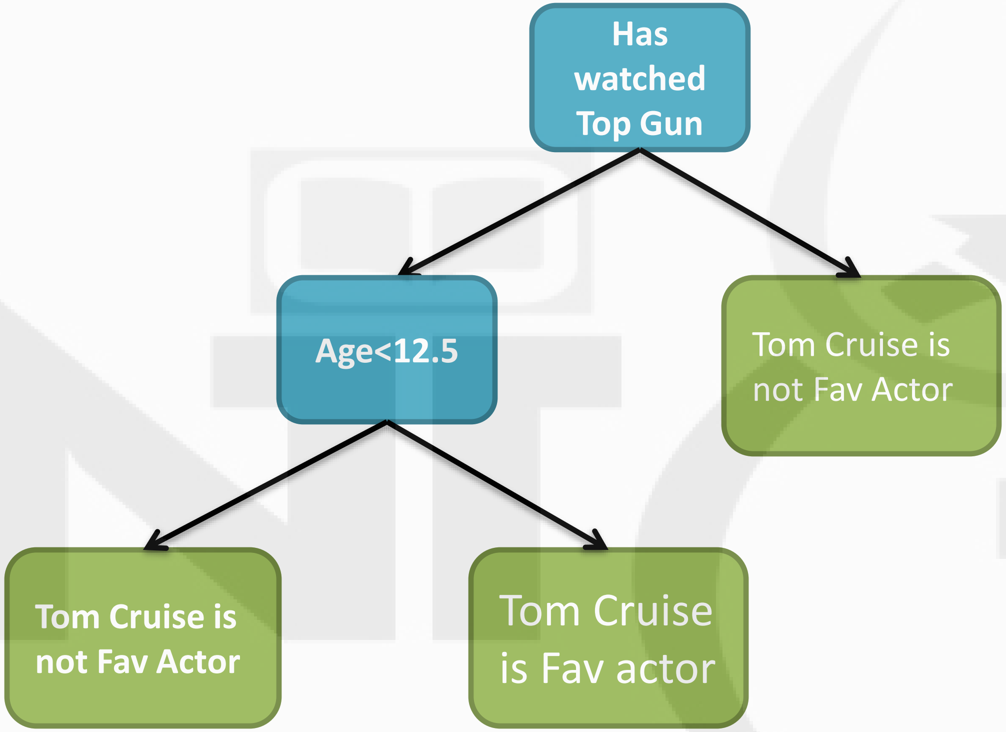
As majority of ppl(1) are in No category so we label it as **"Tom Cruise is not Fav Actor"**

Tom Cruise is Fav actor

All the leaves are pure, as they perfectly classify the ouput

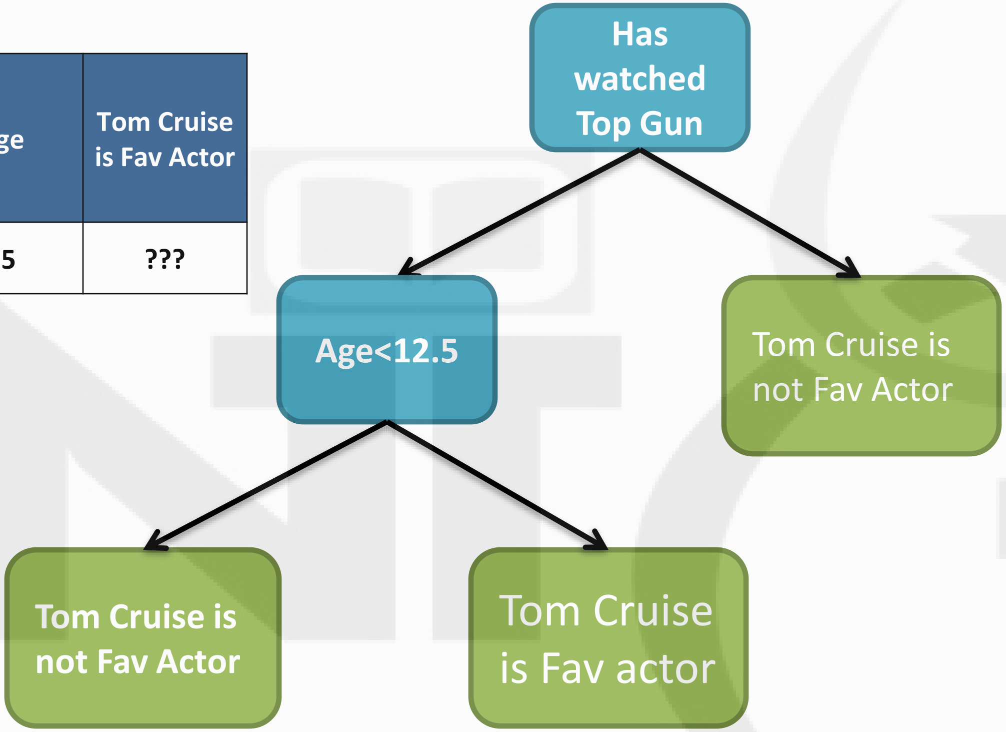


Final Decision tree looks like this



New Data Prediction

Loves Action Gerne	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
Yes	Yes	15	???



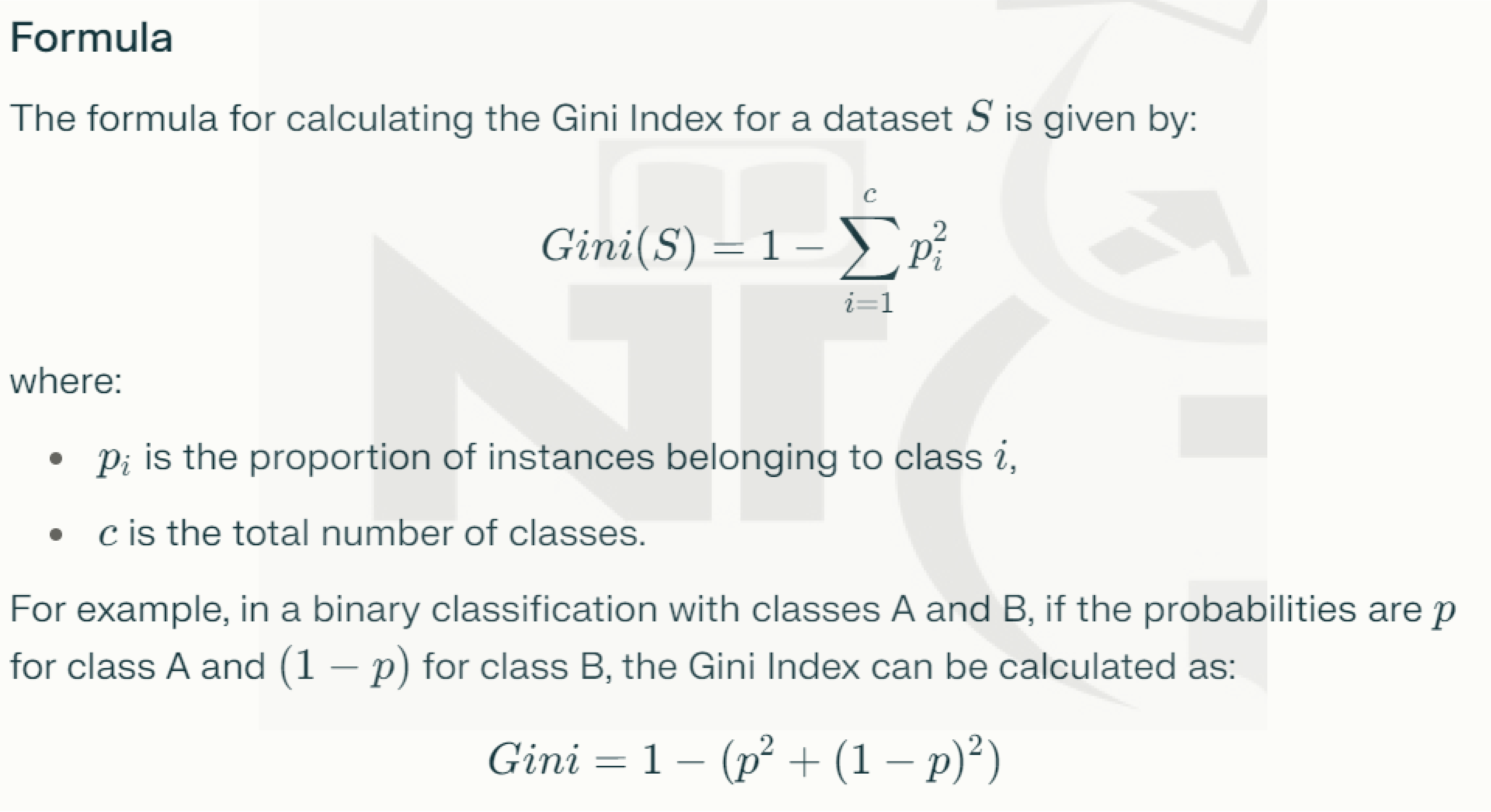
Answer: Yes

There are many ways to measure the impurity of leaves:

- **Entropy**
- **Gini Impurity**
- **Information Gain**
- All these method are similar and Gini is the most popularly used method

Gini Impurity

- **Gini Impurity** measures how impure a dataset is.
- The goal of a decision tree is to find splits that reduce impurity (lower Gini Impurity).
- At each node, the tree tries different splits and chooses the one with the lowest weighted Gini Impurity.



Formula

The formula for calculating the Gini Index for a dataset S is given by:

$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2$$

where:

- p_i is the proportion of instances belonging to class i,
- c is the total number of classes.

For example, in a binary classification with classes A and B, if the probabilities are p for class A and $(1 - p)$ for class B, the Gini Index can be calculated as:

$$Gini = 1 - (p^2 + (1 - p)^2)$$

Definition and Range

- The Gini Index ranges from 0 to 1:0 indicates perfect purity, meaning all instances belong to a single class.

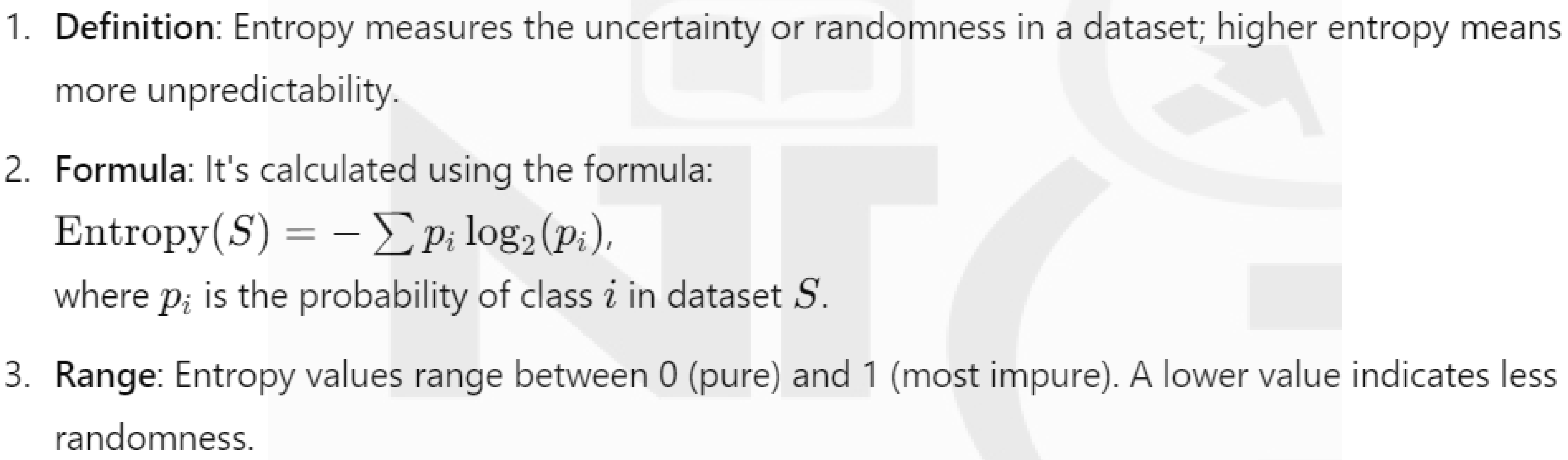
- 1 indicates maximum impurity, where instances are uniformly distributed across multiple classes.

- In binary classification problems, the Gini Index can also reach a maximum value of 0.5 when instances are evenly split between the two classes

Usage in Decision Trees

- In decision trees, the Gini Index is used as a criterion to evaluate potential splits at each node:
  - Calculate Gini for Each Split: For each possible split based on an attribute, compute the Gini Index for the resulting child nodes.
  - Weighted Gini Impurity: The weighted Gini impurity for a split is calculated by averaging the Gini indices of the child nodes, weighted by their sizes.
  - Select Optimal Split: The attribute that results in the lowest weighted Gini impurity is chosen for splitting at that node. This process continues recursively until stopping criteria are met (e.g., reaching a maximum depth or achieving pure leaf nodes).

ENTROPY



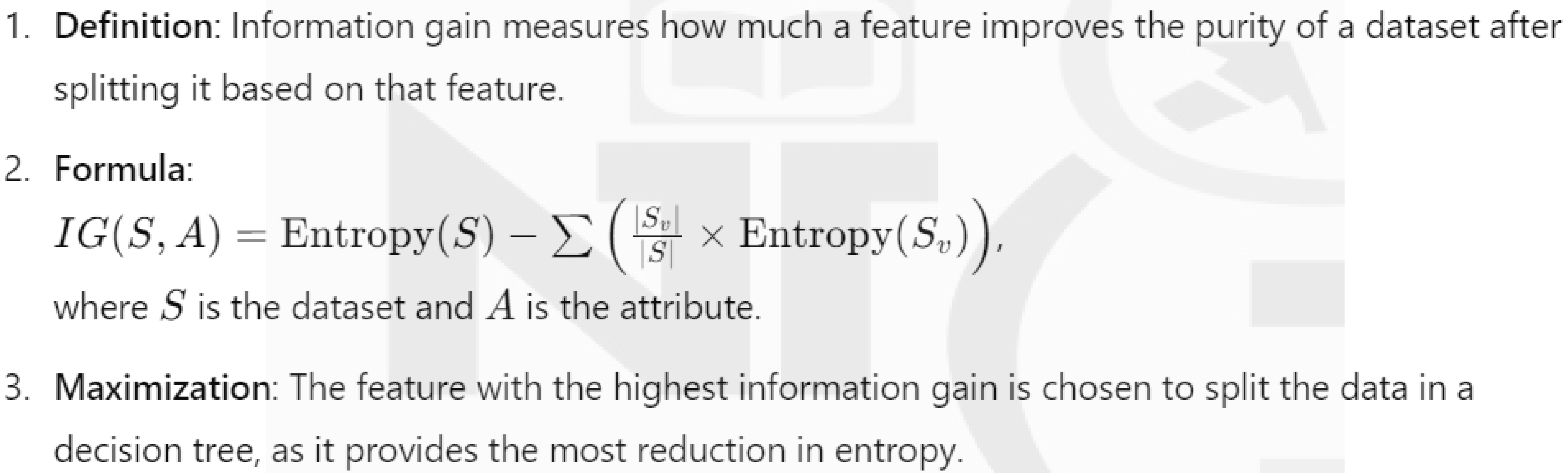
1. **Definition**: Entropy measures the uncertainty or randomness in a dataset; higher entropy means more unpredictability.
2. **Formula**: It's calculated using the formula:
   $\text{Entropy}(S) = -\sum p_i \log_2(p_i)$,
   where p_i is the probability of class i in dataset S.
3. **Range**: Entropy values range between 0 (pure) and 1 (most impure). A lower value indicates less randomness.

4. **Pure Set**: When all data points belong to one class, entropy is 0 (completely pure).

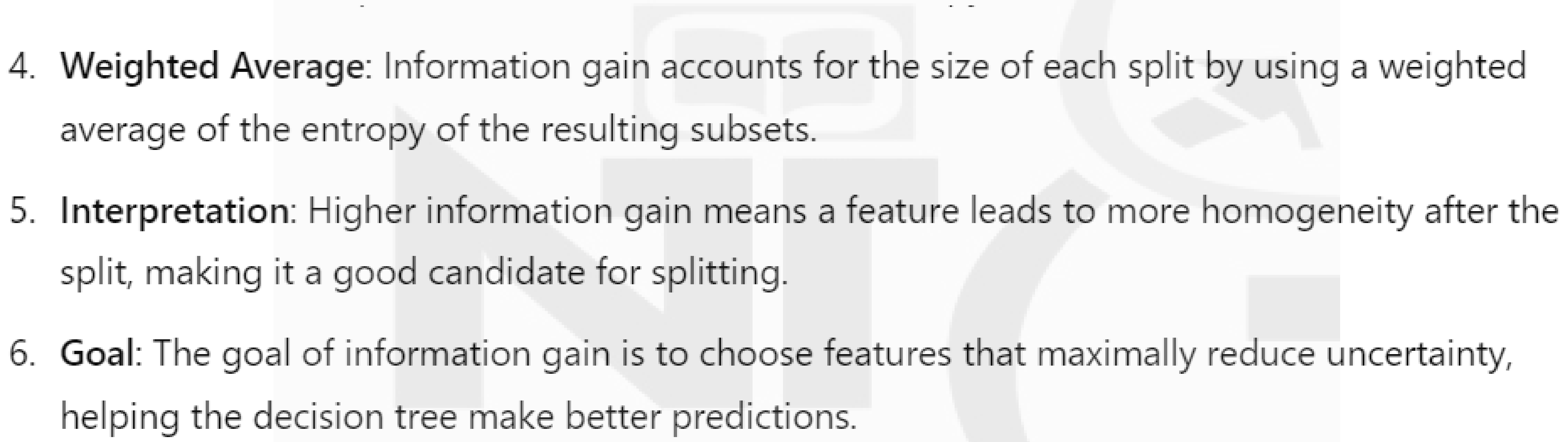
5. **Max Entropy**: When data points are evenly split between classes (e.g., 50% of each class in binary classification), entropy is highest (1 in binary classification).

6. **Goal**: Entropy helps determine how informative a split is, with the goal of reducing entropy at each decision point in a tree.

INFORMATION GAIN

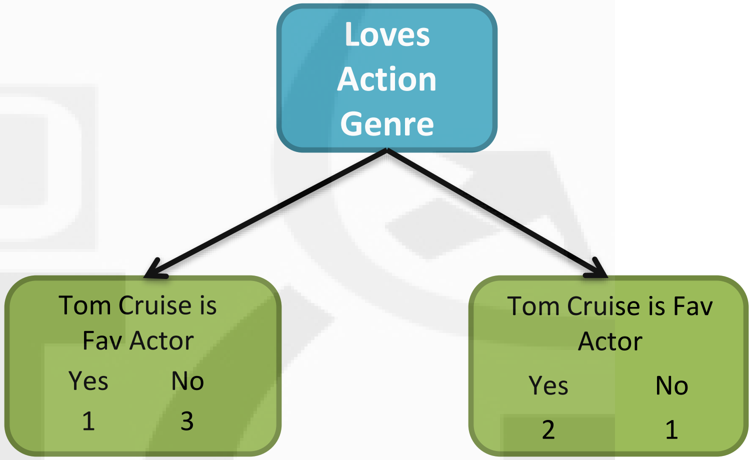


1. **Definition**: Information gain measures how much a feature improves the purity of a dataset after splitting it based on that feature.
2. **Formula**:
$IG(S, A) = \text{Entropy}(S) - \sum \left(\frac{|S_v|}{|S|} \times \text{Entropy}(S_v)\right),$
where S is the dataset and A is the attribute.
3. **Maximization**: The feature with the highest information gain is chosen to split the data in a decision tree, as it provides the most reduction in entropy.



4. **Weighted Average**: Information gain accounts for the size of each split by using a weighted average of the entropy of the resulting subsets.
5. **Interpretation**: Higher information gain means a feature leads to more homogeneity after the split, making it a good candidate for splitting.
6. **Goal**: The goal of information gain is to choose features that maximally reduce uncertainty, helping the decision tree make better predictions.

Loves Action Genre	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Entropy Calculations

$$\boxed{\text{Entropy}(S) = -\sum p_i \log_2(p_i),}$$

a. Left Node (Loves Action Genre = "Yes"):

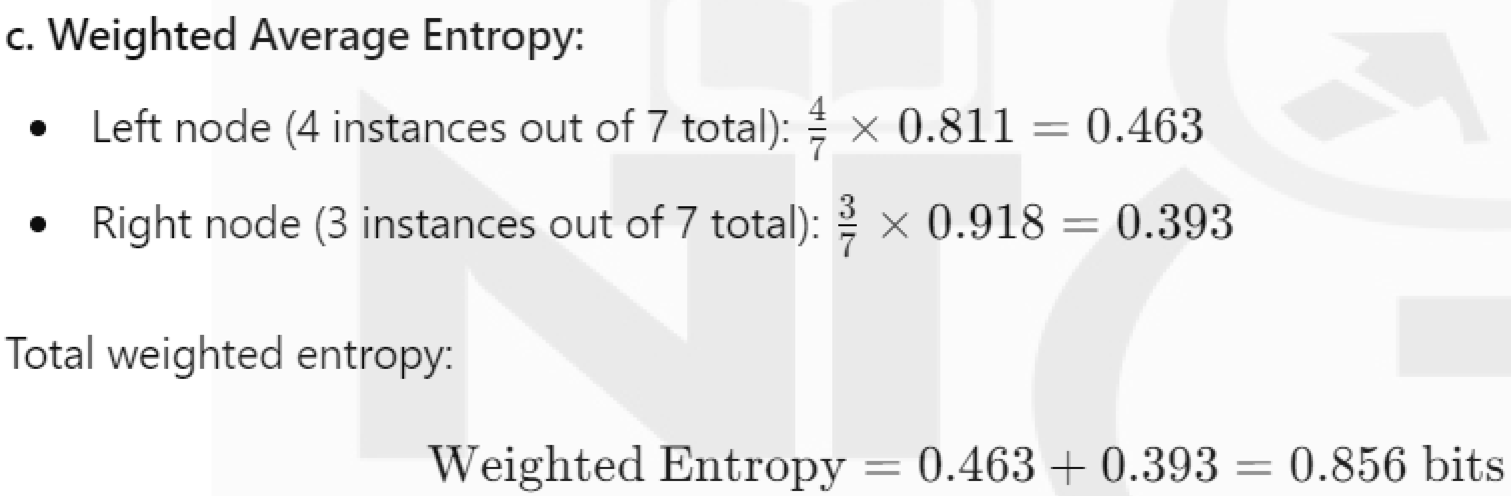
$$\text{Entropy} = -\left(\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right)$$

$$= -\left(\frac{1}{4} \times (-2) + \frac{3}{4} \times (-0.415)\right) = 0.5 + 0.311 = 0.811 \text{ bits}$$

b. Right Node (Loves Action Genre = "No"):

$$\text{Entropy} = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right)$$

$$= -\left(\frac{2}{3} \times (-0.585) + \frac{1}{3} \times (-1.585)\right) = 0.39 + 0.528 = 0.918 \text{ bits}$$



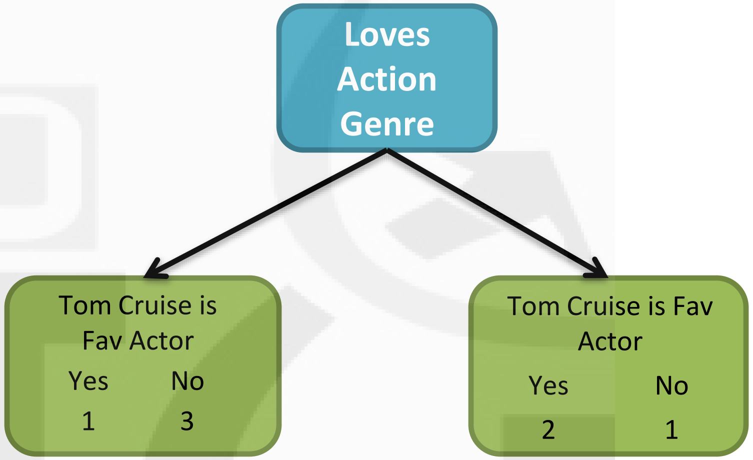
c. Weighted Average Entropy:

- Left node (4 instances out of 7 total): $\frac{4}{7} \times 0.811 = 0.463$
- Right node (3 instances out of 7 total): $\frac{3}{7} \times 0.918 = 0.393$

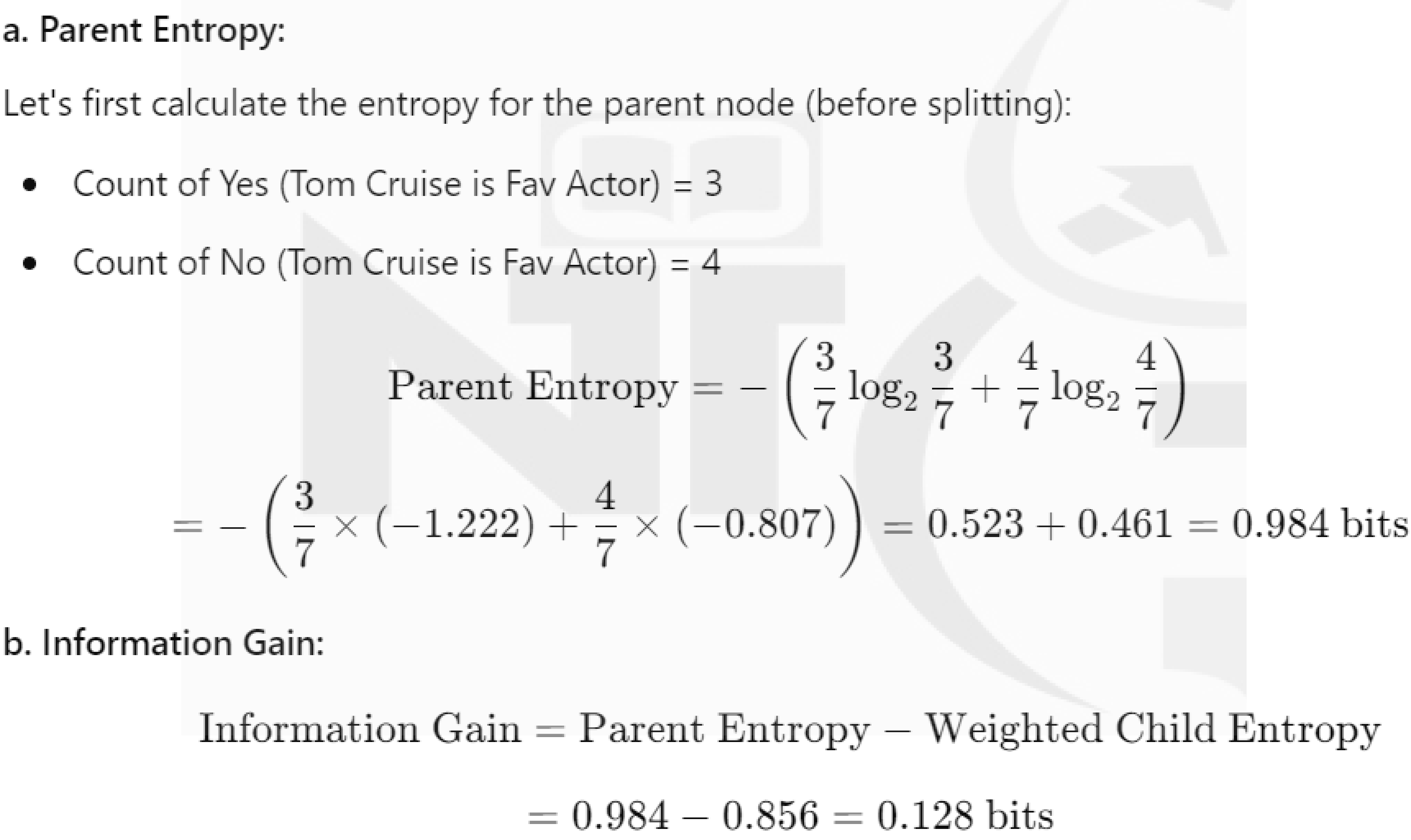
Total weighted entropy:

$$\text{Weighted Entropy} = 0.463 + 0.393 = 0.856 \text{ bits}$$

Loves Action Genre	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Information Gain Calculations



a. Parent Entropy:

Let's first calculate the entropy for the parent node (before splitting):

- Count of Yes (Tom Cruise is Fav Actor) = 3
- Count of No (Tom Cruise is Fav Actor) = 4

$$\text{Parent Entropy} = -\left(\frac{3}{7}\log_2\frac{3}{7} + \frac{4}{7}\log_2\frac{4}{7}\right)$$

$$= -\left(\frac{3}{7} \times (-1.222) + \frac{4}{7} \times (-0.807)\right) = 0.523 + 0.461 = 0.984 \text{ bits}$$

b. Information Gain:

$$\text{Information Gain} = \text{Parent Entropy} - \text{Weighted Child Entropy}$$

$$= 0.984 - 0.856 = 0.128 \text{ bits}$$

To get the best feature :

- **Gini impurity should be low**: A feature with a low Gini impurity results in purer splits, meaning the subsets are more homogenous after splitting.
- **Information Gain should be high**: A high information gain indicates that a feature greatly reduces uncertainty or randomness (entropy) in the dataset, making it a good choice for splitting.
- **Entropy should reduce after the split**: When a dataset is split using a feature, the goal is to reduce the entropy in the resulting subsets. A good feature creates subsets with less disorder or uncertainty, which means entropy is lower than before the split.

Key Concepts

- **Entropy**: A measure of impurity or randomness. It quantifies the uncertainty in the dataset.
- **Gini Impurity**: Another metric to measure the impurity, commonly used in classification.
- **Information Gain**: The reduction in entropy or impurity after splitting the data on a feature. The higher the information gain, the better the feature is for making decisions.
- **Recursive Binary Splitting**: At each step, the algorithm splits the dataset based on the feature that provides the maximum information gain.

Stopping Criteria

- **Max Depth**: The maximum number of levels in the tree.
- **Min Samples per Leaf**: Minimum number of samples required to make a split.
- **No Information Gain**: When splitting no longer reduces impurity.

Advantages of Decision Trees

- Easy to interpret and visualize.
- No need for feature scaling (standardization).
- Handles both numerical and categorical data.
- Non-parametric, meaning it doesn't assume a fixed form for the underlying data distribution.

Disadvantages

- **Overfitting**: Trees can become too complex and fit the noise in the training data.
- **Unstable**: Small changes in the data can lead to completely different trees.
- **Bias towards Features with More Levels**: Features with many possible values can dominate the splits.

Avoiding Overfitting

- **Pruning**: Trimming the branches of the tree to remove nodes that add little predictive power.
- **Setting a max depth**: Limiting the depth of the tree to avoid overly specific patterns.
- **Cross-validation**: Evaluating the performance of the tree on unseen data to ensure it generalizes well.

NODE SPLIT CRITERIA

Advantages & Disadvantages

Advantages:

- Simplicity: The Gini Index is straightforward to compute and interpret.
- Efficiency: It tends to favor larger partitions, which can lead to more balanced splits.

Disadvantages:

- Bias: It may be biased towards attributes with many levels or categories.
- Overfitting Potential: Like other decision tree methods, it can overfit to noisy data if not properly managed through techniques like pruning

HYPER-PARAMETERS

Hyper-Parameters

- Decision tree hyperparameters control how the tree grows, splits data, and prevents overfitting or underfitting. Here's a breakdown of key hyperparameters and their role in shaping the model:
  - ✓ max_depth
  - ✓ min_samples_split
  - ✓ min_samples_leaf
  - ✓ max_features
  - ✓ max_leaf_nodes
  - ✓ min_impurity_decrease
  - ✓ Criterion
  - ✓ ccp_alpha

max_depth

- **Description**: Limits the maximum depth of the tree.
- **Effect**: Controls how many splits the tree can make from the root to a leaf.
- **Purpose**: A deeper tree can model more complex relationships but is also more prone to overfitting.
- **Typical values**: Integer values (e.g., 3, 5, 10). Lower values simplify the model.

min_samples_split

- **Description**: The minimum number of samples required to split an internal node.
- **Effect**: Controls when to stop splitting a node. If a node has fewer than this number of samples, it won’t be split.
- **Purpose**: Higher values prevent the model from creating nodes based on small amounts of data, which helps combat overfitting.
- **Typical values**: Integer (e.g., 2, 10) or float (as a fraction of total samples, e.g., 0.05 for 5%).

min_samples_leaf

- **Description**: The minimum number of samples required to be at a leaf node.
- **Effect**: Prevents nodes from being too small. A node with fewer than this number of samples will not be a leaf.
- **Purpose**: Like min_samples_split, but applies specifically to leaves. Helps prevent overfitting by enforcing larger leaves.
- **Typical values**: Integer or float (e.g., 1, 10, or 0.1 for 10% of samples).

max_features

- **Description**: The number of features to consider when looking for the best split.
- **Effect**: If set to a subset, the model randomly selects features to test at each split.
- **Purpose**: Reduces model complexity by limiting the feature space, which can improve generalization and reduce overfitting.
- **Typical values**: "auto", "sqrt", "log2", or a fraction (e.g., 0.5 for half the features

max_leaf_nodes

- **Description**: Limits the number of leaf nodes in the tree.
- **Effect**: Forces the tree to stop growing after reaching this number of leaves.
- **Purpose**: Controls the complexity of the tree. Fewer leaf nodes reduce overfitting.
- **Typical values**: Integer (e.g., 10, 100).

min_impurity_decrease

- **Description**: A node will be split only if it decreases the impurity by at least this value.
- **Effect**: Forces the model to create splits only when they result in a significant improvement.
- **Purpose**: Helps avoid creating splits that add very little value, reducing the likelihood of overfitting.
- **Typical values**: Float (e.g., 0.0 for no constraint, 0.01 for stricter splitting).

criterion

- **Description**: The function to measure the quality of a split.
- **Options**:
  - **For classification**: gini (Gini impurity) or entropy (information gain).
  - **For regression**: mse (Mean Squared Error), mae (Mean Absolute Error), or poisson.
- **Effect**: Defines how the tree decides which feature to split on and where to make the cut.
- **Purpose**: Determines how the decision tree evaluates splits.
- **Typical values**: For classification, "gini" or "entropy"; for regression, "mse".

ccp_alpha (Cost Complexity Pruning)

- **Description**: A pruning parameter that penalizes the complexity of the tree.
- **Effect**: Higher values of ccp_alpha will result in smaller trees as it prunes nodes that don't add significant value.
- **Purpose**: Used to prevent overfitting by pruning less important branches.
- **Typical values**: Float (e.g., 0.01 for more aggressive pruning).

Practical Tips for Tuning Hyperparameters

- **Overfitting**: A model with too many splits or deep trees will learn the noise in your data.
  - To avoid this, use ***max_depth, min_samples_split, or min_samples_leaf***.
- **Underfitting**: Too shallow or constrained trees may miss important patterns.
  - To avoid underfitting, allow deeper trees or reduce the minimum samples per leaf or split.

ASSIGNMENTS

Assignments

- **Wine Dataset which was already solved in class**
- **Perfrom a gridsearch for all these params and compare accuracy:**
  - max_depth
  - min_samples_split
  - min_samples_leaf
  - max_features
  - max_leaf_nodes
  - min_impurity_decrease
  - Criterion
  - ccp_alpha

Assignments

- **Build a Decision tree classifier for Parkinsons dataset:**
  - https://archive.ics.uci.edu/dataset/174/parkinsons
  - Build basic DT classifier
  - Perform Hyper parameter tuning
  - Compare accuracies