DECISION TREES

A Decision Tree is a supervised machine learning algorithm that is used for both classification and regression tasks. It models decisions by splitting the data into branches based on feature values, forming a tree-like structure where each internal node represents a decision on a feature, each branch represents an outcome of the decision, and each leaf node represents a final prediction or outcome.

Important features of decision trees are –

1. Non-Parametric model: A decision tree is considered a non-parametric model because it doesn't assume a specific mathematical form for the relationship between inputs and outputs.

2. White Box model: A decision tree is often called a "white box" model because its decision-making process is easy to understand and interpret.

3. It is a giant if-else based model.

Why Do We Need Entropy, Information Gain & Gini Index?

A decision tree works by splitting data into smaller and purer subsets.

At every node, the tree must answer:

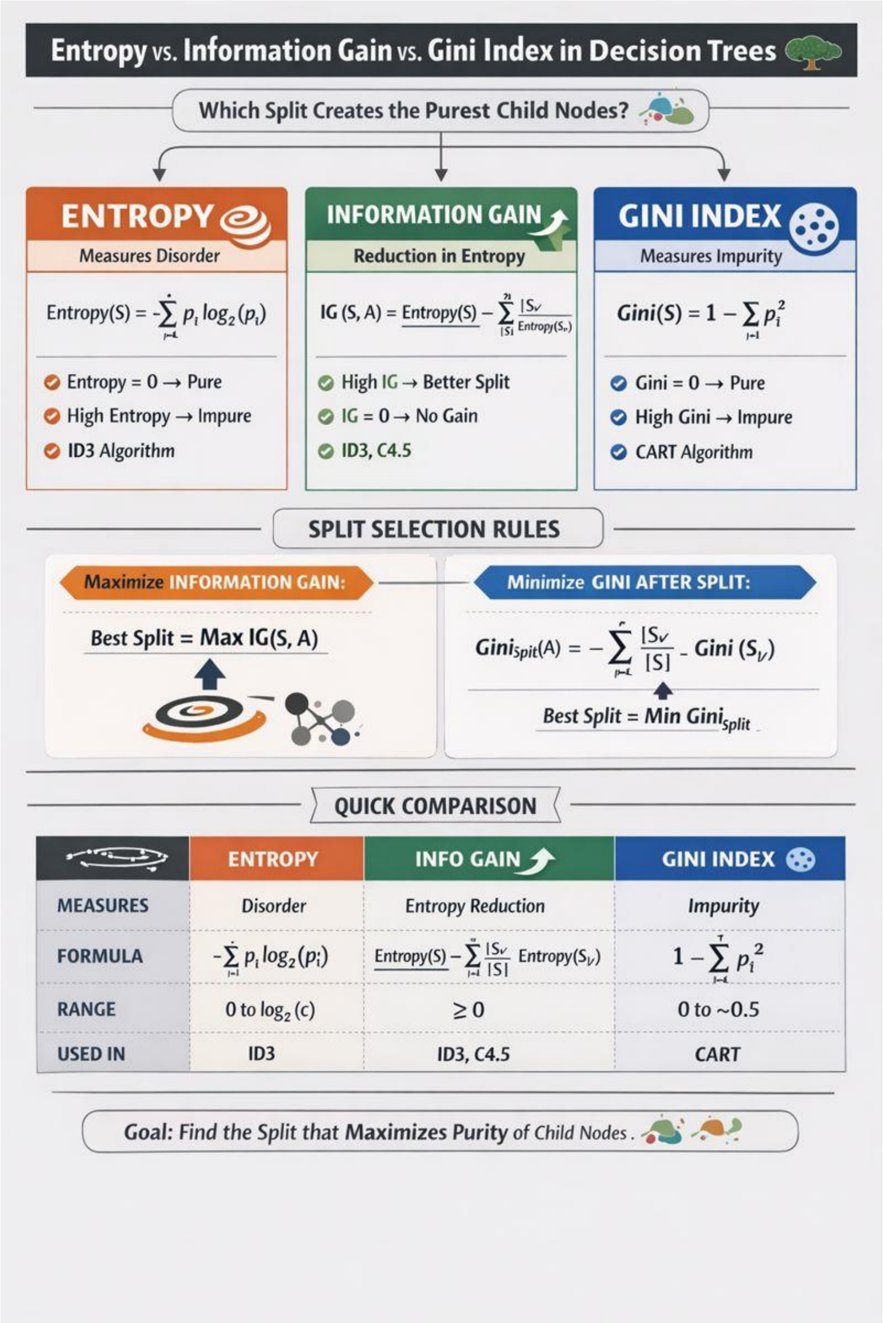
"Which feature and which split produces the purest child nodes?"

To answer this we use -

- Entropy → measures randomness
- Gini Index → measures impurity
- Information Gain → measures impurity reduction after a split

Symbol Meaning

- S - Original dataset
- N - Number of classes
- Pi - Probability of class i in dataset S
- A – Attribute used for splitting



Entropy vs. Information Gain vs. Gini Index in Decision Trees

Which Split Creates the Purest Child Nodes?

ENTROPY

Measures Disorder

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

- Entropy = 0 → Pure
- High Entropy → Impure
- ID3 Algorithm

INFORMATION GAIN

Reduction in Entropy

$IG(S, A) = \text{Entropy}(S) - \sum \frac{|S_v}{|S|} \text{Entropy}(S_v)$

- High IG → Better Split
- IG = 0 → No Gain
- ID3, C4.5

GINI INDEX

Measures Impurity

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

- Gini = 0 → Pure
- High Gini → Impure
- CART Algorithm

SPLIT SELECTION RULES

Maximize INFORMATION GAIN:

Best Split = Max IG(S, A)

Minimize GINI AFTER SPLIT:

$Gini_{spit}(A) = -\sum_{i=1} \frac{[S_v}{[S]} - Gini(S_v)$

Best Split = Min $Gini_{split}$

QUICK COMPARISON

	ENTROPY	INFO GAIN	GINI INDEX
MEASURES	Disorder	Entropy Reduction	Impurity
FORMULA	$-\sum_{i=1} p_i \log_2(p_i)$	$\text{Entropy}(S) - \sum_{i=1} \frac{|S_v}{|S|} \text{Entropy}(S_v)$	$1 - \sum_{i=1} p_i^2$
RANGE	0 to $\log_2(c)$	≥ 0	0 to ~0.5
USED IN	ID3	ID3, C4.5	CART

Goal: Find the Split that Maximizes Purity of Child Nodes.

Gini Index After a Split

Suppose a split divides node $t$into:

- Left child t_L
- Right child t_R

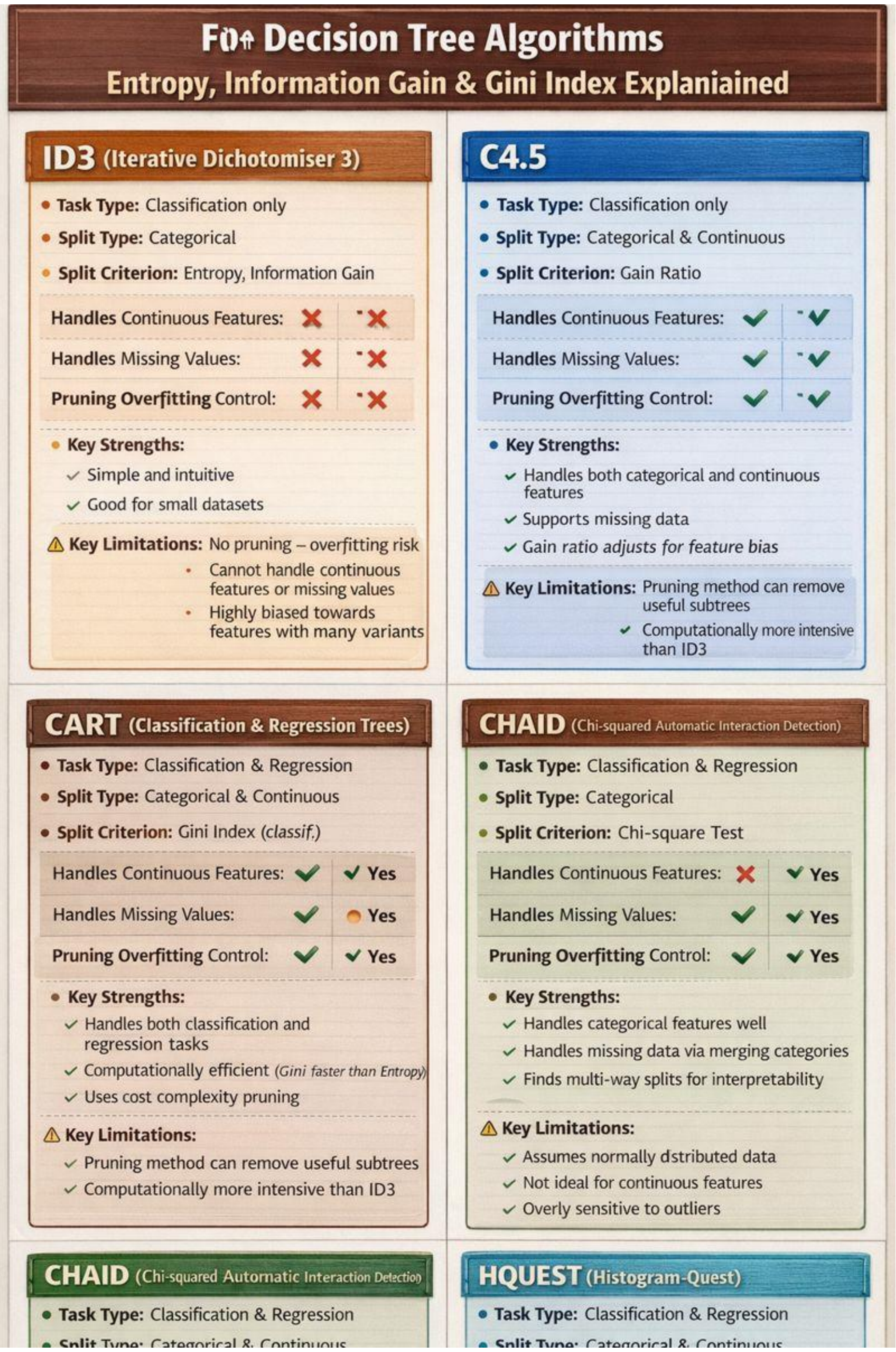
Then the Gini after split is:

$$\boxed{\text{Gini}_{\text{split}} = \frac{N_L}{N} \text{Gini}(t_L) + \frac{N_R}{N} \text{Gini}(t_R)}$$

where:

- N= total samples in parent node
- N_L= samples in left child
- N_R= samples in right child

Various algorithms to create Decision Trees are –



F0↑ Decision Tree Algorithms

Entropy, Information Gain & Gini Index Explaniained

ID3 (Iterative Dichotomiser 3)

- **Task Type:** Classification only
- **Split Type:** Categorical
- **Split Criterion:** Entropy, Information Gain

Handles Continuous Features:	✗	✗
Handles Missing Values:	✗	✗
Pruning Overfitting Control:	✗	✗

- **Key Strengths:**
  - ✓ Simple and intuitive
  - ✓ Good for small datasets

⚠ **Key Limitations:** No pruning – overfitting risk

- Cannot handle continuous features or missing values
- Highly biased towards features with many variants

C4.5

- **Task Type:** Classification only
- **Split Type:** Categorical & Continuous
- **Split Criterion:** Gain Ratio

Handles Continuous Features:	✓	✓
Handles Missing Values:	✓	✓
Pruning Overfitting Control:	✓	✓

- **Key Strengths:**
  - ✓ Handles both categorical and continuous features
  - ✓ Supports missing data
  - ✓ Gain *ratio adjusts for feature bias*

⚠ **Key Limitations:** Pruning method can remove useful subtrees

✓ Computationally more intensive than ID3

CART (Classification & Regression Trees)

- **Task Type:** Classification & Regression
- **Split Type:** Categorical & Continuous
- **Split Criterion:** Gini Index (*classif.*)

Handles Continuous Features:	✓	✓ Yes
Handles Missing Values:	✓	● Yes
Pruning Overfitting Control:	✓	✓ Yes

- **Key Strengths:**
  - ✓ Handles both classification and regression tasks
  - ✓ Computationally efficient *(Gini faster than Entropy)*
  - ✓ Uses cost complexity pruning

⚠ **Key Limitations:**

- ✓ Pruning method can remove useful subtrees
- ✓ Computationally more intensive than ID3

CHAID (Chi-squared Automatic Interaction Detection)

- **Task Type:** Classification & Regression
- **Split Type:** Categorical
- **Split Criterion:** Chi-square Test

Handles Continuous Features:	✗	✓ Yes
Handles Missing Values:	✓	✓ Yes
Pruning Overfitting Control:	✓	✓ Yes

- **Key Strengths:**
  - ✓ Handles categorical features well
  - ✓ Handles missing data via merging categories
  - ✓ Finds multi-way splits for interpretability

⚠ **Key Limitations:**

- ✓ Assumes normally dstributed data
- ✓ Not ideal for continuous features
- ✓ Overly sensitive to outliers

CHAID (Chi-squared Automatic Interaction Detectioŋ

- **Task Type:** Classification & Regression
- **Split Type:** Categorical & Continuous

HQUEST (Histogram-Quest)

- **Task Type:** Classification & Regression
- **Split Type:** Categorical & Continuous

ID3 ALGORITHM

ID3 (Iterative Dichotomiser 3) is a greedy, top-down decision tree learning algorithm used for classification, which constructs a tree by recursively selecting the attribute that maximizes Information Gain (based on entropy) at each node.

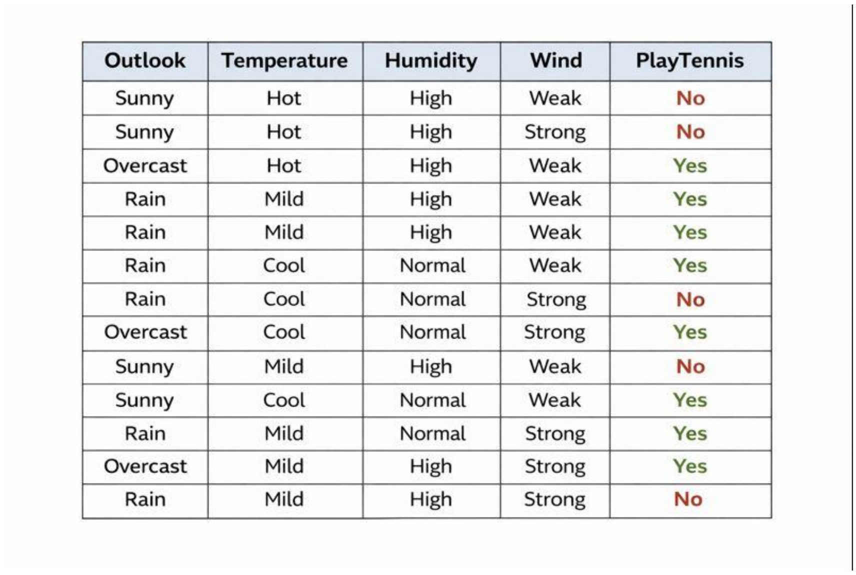
Steps of the ID3 Algorithm (Working) are -

1. Start with the full training dataset
2. Calculate entropy of the target class
3. For each attribute:
   - Partition the dataset based on attribute values
   - Compute entropy for each partition
   - Compute Information Gain
4. Select attribute with highest Information Gain
5. Create a decision node for that attribute
6. Repeat recursively for each branch
7. Stop when:
   - All samples belong to one class
   - No attributes remain
   - Dataset is empty (use majority class)

Problem Statement

Predict whether a person will Play Tennis based on weather conditions.

Dataset –



Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Rain	Mild	High	Strong	No

Step 1□Calculate Entropy of Target Variable

- Yes = 9

- No = 5

$$\begin{aligned}&\text{Entropy}(S) = -\left(\frac{9}{14}\log_2\frac{9}{14} + \frac{5}{14}\log_2\frac{5}{14}\right)\\&= -(0.643 \times -0.64 + 0.357 \times -1.485)\\&= 0.94\end{aligned}$$

Step 2⃣ Compute Information Gain for Each Attribute

◈ Attribute: Outlook

Value	Yes	No	Entropy
Sunny	2	3	0.971
Overcast	4	0	0
Rain	3	2	0.971

$$\begin{aligned}&\text{Entropy after split} = \frac{5}{14}(0.971) + \frac{4}{14}(0) + \frac{5}{14}(0.971) = 0.693\\&\text{IG}(Outlook) = 0.94 - 0.693 = 0.247\end{aligned}$$

Attribute: Temperature

$$\text{IG}(Temperature) = 0.029$$

◈ Attribute: Humidity

$$\text{IG}(Humidity) = 0.151$$

◈ Attribute: Wind

$$\text{IG}(Wind) = 0.048$$

Step 3⃣ Select Best Attribute

Attribute	Information Gain
Outlook	0.247 (Highest)
Humidity	0.151
Wind	0.048
Temperature	0.029

✅ Root Node = Outlook

Step 4️⃣ Recursive Splitting

🌿 Outlook = Overcast

- All examples = Yes
- Leaf node = Yes

🌿 Outlook = Sunny

Subset:

Humidity	PlayTennis
High	No
High	No
High	No
Normal	Yes
Normal	Yes

Split on Humidity → pure nodes

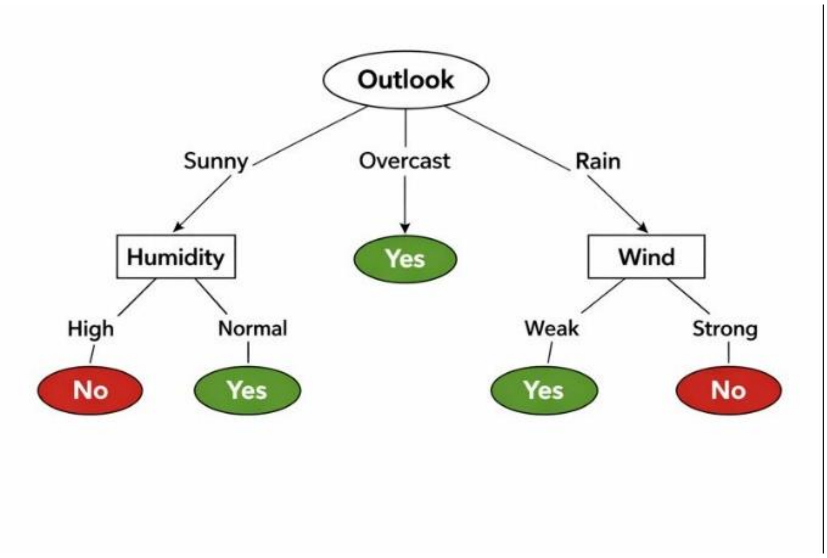
🌿 Outlook = Rain

Subset:

Wind	PlayTennis
Weak	Yes
Weak	Yes
Weak	Yes
Strong	No
Strong	No

Split on Wind → pure nodes

Final Decision Tree -



CART ALGORITHM

CART (Classification and Regression Trees) is a binary decision tree learning algorithm that builds trees by recursively selecting the feature and threshold that minimize node impurity, using Gini Index for classification and Mean Squared Error (MSE) for regression.

4⃣CART Algorithm — Step-by-Step Working

General Procedure

1. Start with the full dataset at the root
2. For each feature:
   - Try all possible split thresholds
   - Compute impurity of resulting binary splits
3. Select the split with lowest impurity
4. Create left and right child nodes
5. Recursively repeat for each child
6. Stop when:
   - Node becomes pure
   - Minimum samples reached
   - Maximum depth reached
7. Apply cost-complexity pruning to remove weak branches

Problem Statement

Predict whether a person will Buy a Computer.

📊 Dataset

Age	Income	Student	Credit	Buy
Youth	High	No	Fair	No
Youth	High	No	Excellent	No
Middle	High	No	Fair	Yes

Age	Income	Student	Credit	Buy
Senior	Medium	No	Fair	Yes
Senior	Low	Yes	Fair	Yes
Senior	Low	Yes	Excellent	No
Middle	Low	Yes	Excellent	Yes
Youth	Medium	No	Fair	No
Youth	Low	Yes	Fair	Yes
Senior	Medium	Yes	Fair	Yes
Youth	Medium	Yes	Excellent	Yes
Middle	Medium	No	Excellent	Yes
Middle	High	Yes	Fair	Yes
Senior	Medium	No	Excellent	No

Step 1️⃣ Gini Impurity of Root Node

- Yes = 9
- No = 5

$$\text{Gini}(S) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2$$
$$= 1 - (0.413 + 0.127) = 0.46$$

Step 2️⃣ Evaluate Possible Splits

💠 Split on Age

CART forces binary splits, so multi-category features are split as subsets.

Example split:

- Left: {Youth}
- Right: {Middle, Senior}

Left (Youth)

- Yes = 2, No = 3

$$\text{Gini}_L = 1 - (0.4^2 + 0.6^2) = 0.48$$

Right (Middle + Senior)

- Yes = 7, No = 2

$$\text{Gini}_R = 1 - (0.78^2 + 0.22^2) = 0.35$$

Weighted Gini

$$\text{Gini}_{split} = \frac{5}{14}(0.48) + \frac{9}{14}(0.35) = 0.40$$

💠 Split on Student (Yes / No)

Student = Yes

- Yes = 6, No = 1

$$\text{Gini}_L = 1 - (0.86^2 + 0.14^2) = 0.24$$

Student = No

- Yes = 3, No = 4

$$\text{Gini}_R = 1 - (0.43^2 + 0.57^2) = 0.49$$

Weighted Gini

$$\text{Gini}_{split} = \frac{7}{14}(0.24) + \frac{7}{14}(0.49) = 0.36$$

💠 Split on Credit Rating

Weighted Gini ≈ 0.43

Step 3⃣ Choose Best Split

Attribute	Weighted Gini
Age	0.40
Student	0.36 (Lowest)
Credit	0.43

✅ Root Node = Student

Step 4⃣ Recursive Splitting

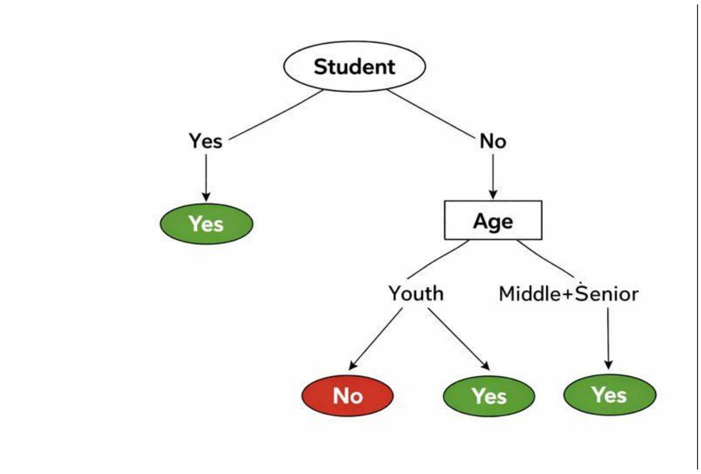
🌿 Student = Yes

Mostly Yes, stop or split further if needed.

🌿 Student = No

Split further on Age or Credit using same procedure.

Final CART Tree



Pruning in CART (Cost-Complexity)

CART uses:

$$R_{\alpha}(T) = R(T) + \alpha \mid T \mid$$

Where:

- $R(T)$= misclassification error
- $\mid T \mid$= number of leaf nodes
- α= complexity parameter

Prunes subtrees that do not sufficiently reduce error.

Difference between gini impurity and entropy

Pre – Pruning and Post – Pruning

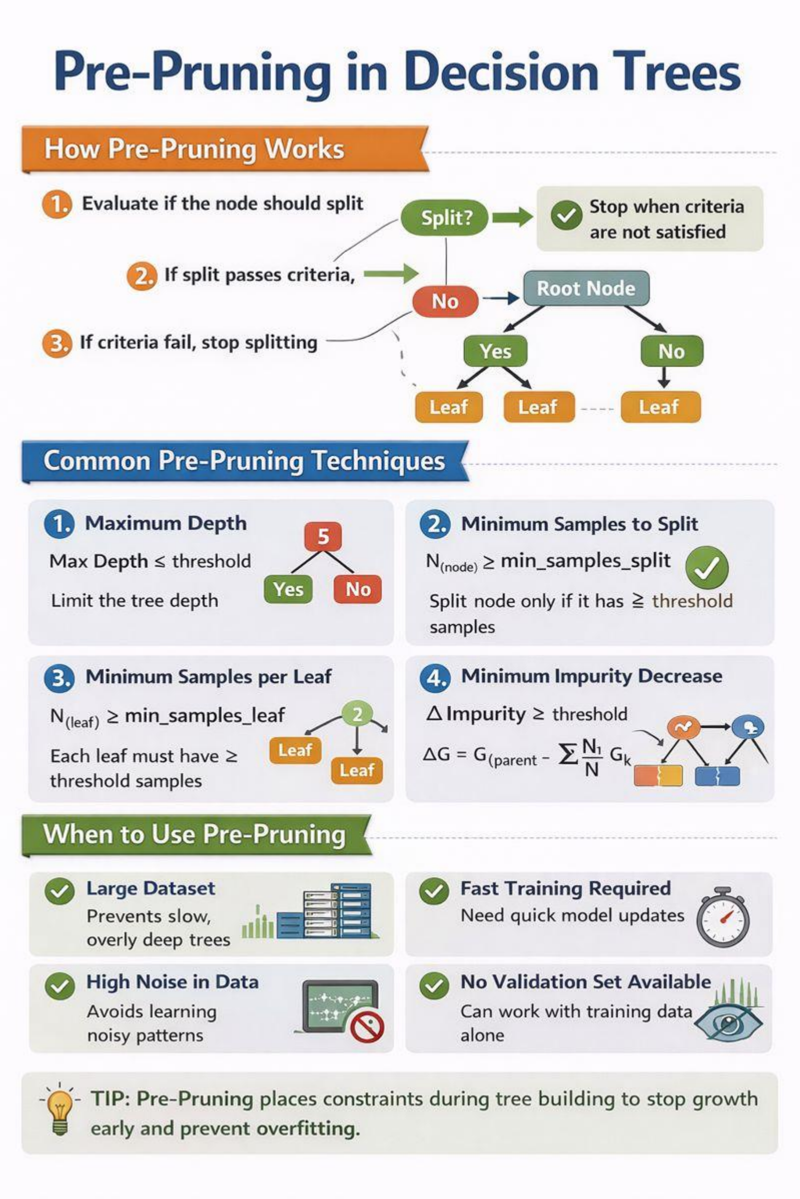
Pruning is the process of removing unnecessary branches to reduce overfitting, improve generalization, and simplify the tree.

Reasons to pruning –

1️⃣Prevents overfitting – Pruning removes overly specific branches so the tree generalizes better to unseen data.

2️⃣Improves interpretability – A pruned tree is smaller and simpler, making decision rules easier to understand and explain.

3️⃣Reduces variance and noise learning – By eliminating splits based on small or noisy samples, pruning stabilizes predictions.



Pre-Pruning in Decision Trees

How Pre-Pruning Works

1. Evaluate if the node should split
2. If split passes criteria,
3. If criteria fail, stop splitting

Common Pre-Pruning Techniques

1. **Maximum Depth**
   Max Depth ≤ threshold
   Limit the tree depth

2. **Minimum Samples to Split**
   $N_{(node)} \geq$ min_samples_split ✓
   Split node only if it has ≥ threshold samples

3. **Minimum Samples per Leaf**
   $N_{(leaf)} \geq$ min_samples_leaf
   Each leaf must have ≥ threshold samples

4. **Minimum Impurity Decrease**
   Δ Impurity ≥ threshold
   $\Delta G = G_{(parent} - \sum \frac{N_1}{N} G_k$

When to Use Pre-Pruning

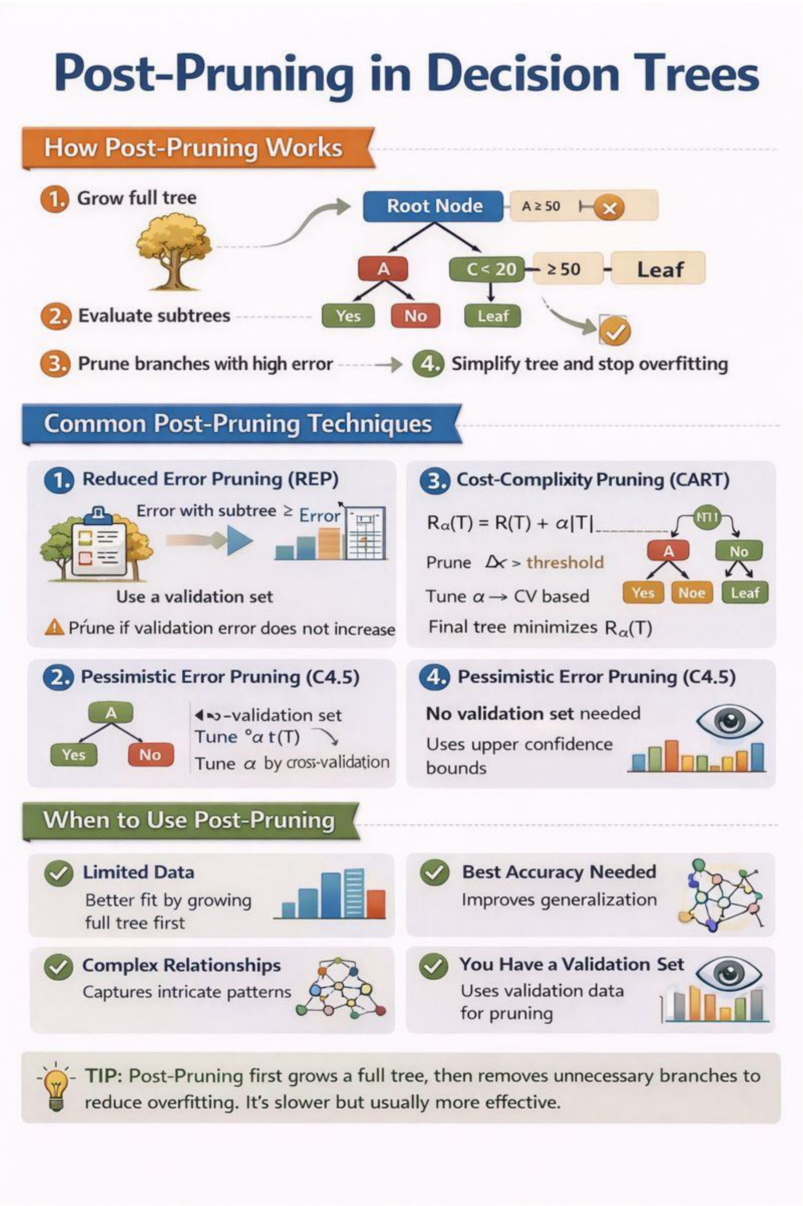
✓ **Large Dataset**
Prevents slow, overly deep trees

✓ **Fast Training Required**
Need quick model updates

✓ **High Noise in Data**
Avoids learning noisy patterns

✓ **No Validation Set Available**
Can work with training data alone

TIP: Pre-Pruning places constraints during tree building to stop growth early and prevent overfitting.



Post-Pruning in Decision Trees

How Post-Pruning Works

1. Grow full tree

2. Evaluate subtrees

3. Prune branches with high error → 4. Simplify tree and stop overfitting

Common Post-Pruning Techniques

1. Reduced Error Pruning (REP)

Use a validation set

⚠ Prúne if validation error does not increase

3. Cost-Complixity Pruning (CART)

$R_\alpha(T) = R(T) + \alpha|T|$

Prune $\Delta\alpha$ > threshold

Tune α → CV based

Final tree minimizes $R_\alpha(T)$

2. Pessimistic Error Pruning (C4.5)

◂∾-validation set

Tune °α t(T)

Tune α by cross-validation

4. Pessimistic Error Pruning (C4.5)

No validation set needed

Uses upper confidence bounds

When to Use Post-Pruning

✓ **Limited Data**
Better fit by growing full tree first

✓ **Best Accuracy Needed**
Improves generalization

✓ **Complex Relationships**
Captures intricate patterns

✓ **You Have a Validation Set**
Uses validation data for pruning

TIP: Post-Pruning first grows a full tree, then removes unnecessary branches to reduce overfitting. It's slower but usually more effective.