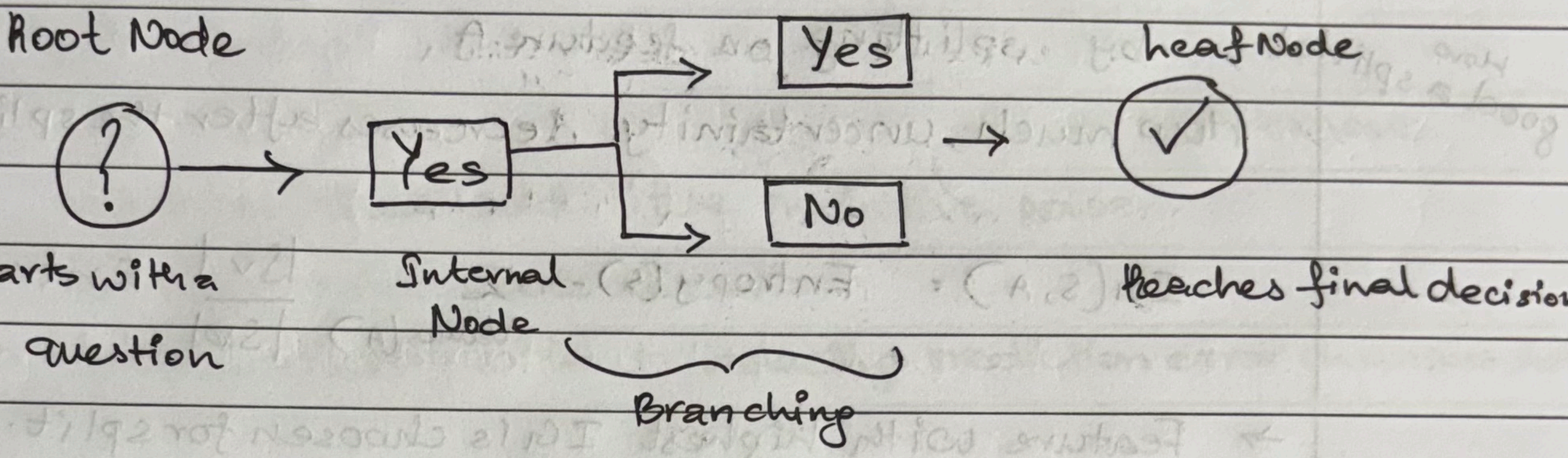
__/__/____

(19) Decision Trees

A Decision Tree is a flowchart-like model that splits the dataset into branches based on feature values to predict a target variable.



Why?

✓ Handles both numerical & categorical data

✓ No feature scaling required

✓ Non-linear boundaries.

It recursively splits the data using a crieterion that measures purity (homogeneity) of that target variable in each nod

→ When building decisiontree - which feature and split make the child nodes as pure as possible?

* The splitting criterian is the rule or formula that decides which split to choose - ie, which one reduces the impurity the most

Splitting Criterion - Reduction in Impurity

__/__/____

→ Impurity Measures

ID3 Algos

[Small dataset]

(a) Entropy , when entropy = 0 ? perfectly pure node.

↳ disorder / randomness / impurity

$$\text{Entropy}(S) = \sum_{i=1}^{k} p_i \log_2 p_i \qquad [0, 1]$$

How good a split is?

(b) Information Gain : How much impurity is reduced by splitting on feature A

How much uncertainity decreases after the split

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

→ Feature with highest IG is choosen for split.

CART Algos

[Large dataset]

(c) Gini Impurity : How often a randomly chosen element would be incorrectly identified.

An attribute with lower Gini Index is preferred.

$$G.I = 1 - \sum_{i=1}^{k} (p)^2 \qquad [0, 0.5]$$

Decision Trees:

1. Start with root node
2. Based on impurity find best feature to split
3. Split the nodes into child nodes
4. Repeat recursively for each child.

If allowed to grow indefinitely:

A tree will : split till it is pure (impurity = 0)

→ This leads to overfitting

→ We fix this using pre pruning and post pruning

__/__/____

(20) 1. Stopping criteria (Pre-Pruning)

- Prevent the tree from growing too deep or too complex

Some criterias are:

(ID3) min_samples_split, max_depth, max_leaf_nodes...

* Balances tradeoff b/w bias & variance
* Faster, done during training

2. Post Pruning

(CART) - After building a tree fully, we prune it to remove branches that add little predictive power.

* Donot improve generalization
* Add complexity without reducing prediction error on unseen dat

→ Reduced Error Pruning

1. Start with a fully grown tree
2. Remove one node at a time.
3. If removing does not reduce accuracy on validation set
   → keep it pruned.
4. Continue until pruning worsens validation performance.

→ Cost Complexity Pruning (weakest link pruning)

Used in (CART)

Cost function $R_\alpha(T) = R(T) + \alpha \cdot |T|$ (error, penalty, no of leaf nodes)

- Compute $R_\alpha(T)$ for different α values
- Use cross validation and find α that minimizes val error

__/__/____

(21) Regression Trees : Predicts a continuous numeric output

1. Choose Split based on SSE,

Best Split = $\arg\min_{j,s}$ [SSE left + SSE right]

2. - Weare minimizing the variance within each node.

- After every split child nodes should be more homogeneous in target values than the parent

$$\text{loss} = \sum_{m=1}^{M} \sum_{x_i \in R_m} \left(y_i - \bar{y}_{Rm}\right)^2$$

3. Once the best splits are found, the prediction for each terminal node is

$$\hat{y}_{Rm} = \text{mean}(y_i \in R_m)$$

→ Regression minimizes variance

→ Classification minimizes impurity (Maximize IG)

(22) Categorical Attributes and Splits

↳ split the data based on equality rather than threshold

1. Binary Split (two-way): Divide categories into two groups

- Colour ∈ {Red, Green, Blue}
- Splits: {Red, Green} vs {Blue} or {Blue, Green} vs {Red}

Classification: 1. Compute impurity for left & right groups

2. Compute IG

3. choose group with higher / maximum gain

__/__/____

2. Multiway Split : we create one branch per category

(ID3, C4.5)

↳ Each branch leads to a subset of data with that value

$$IG = I(\text{parent}) - \sum_k \frac{|R_k|}{|R|} I(R_k)$$

R_k = subsets of each category

* Overfitting risks - if too many categories

→ Overfitting with high cardinality features (categorical)

- Multiway splits produce thousands of child nodes
- Each node might contain very few samples → overfitting

Solns: 1. Prefer Binary Splits

2. Use target encoding / one-hot encoding

3. Merge categories with similar class distributions.

→ If you allow multiway splits, nodes with many branches may seem better (higher IG).

To fix that C4.5 introduced Gain Ratio

Gain Ratio = IG / SplitInfo

where $SplitInfo = -\sum |R_i|/|R| \log_2 |R_i|/|R|$

This penalizes attributes that create too many splits.

→ Use when categorical variables has many distinct values.

__/__/____

(23) Missing Values in Decision Trees

{ Real world datasets often have missing feature values.
{ Standard Split cannot be applied if values are missing
↘ Decision Trees handle missing values naturally

① Imputation (Filling missing values)

Replace missing values with

Single Imputation
1. Median
2. Mean (Numerical)
3. Mode (categorical)

* Simple & Fast
* Can bias splits if many values are missing.

② Surrogate Splits (D.T Specific)

- Use another feature as backup if/when the primary splitting feature is missing.

(CART)

1. During tree training:
- For each split, find primary feature and threshold that maximises impurity reduction.
- Look for other features that surrogates that mimic the split
↳ Split the data almost the same way as the primary feat
↳ Compute agreement rate : proportion of samples split the same way.

at a
2. If a sample node has missing value in primary feature, follow the surrogate split instead.

* Even though higher computation → no artificial values are generated.