Decision Trees

This can be used for both classification and regression tasks

Decision tree is called as non metric methods for classification.

Decision tree are non metric classifer because they perform classification through rule-based splitting rather than distance or similarity computation

Metric Based classification Methods

A metric based classification technique that relies on:

→ A distance measure a

→ A similarity measure (dot product)

Non metric Methods for classification

1) Problem with some datasets

→ in many real world Problems, features are not numeric

→ These features are called as nominal features

2) Nominal features:

→ nominal features are descriptive in nature.

→ They describe the characteristics of a class, not quantities

→ Example.

* colour
* shape
* Taste

3) Why Numeric Representation is not meaningful

→ These features do not have a natural order.

→ Converting them into numbers can create wrong assumptions.

→ Example.

* Red = 1, Green = 2, Yellow = 3
* This wrongly suggests yellow > Green > Red.

4) Metric cannot be computed

→ Since features are not numeric, we cannot calculate distance or similarity

→ Therefore, metric based methods cannot be used.

5) Non-Metric models:

→ classification models built using nominal features are called as

NON metric methods based models.

→ Example:

* Decision Tree
* Random Forest.

Nominal feature : Illustration

* Nominal features are non numeric features.

* They describe an object in a natural, human readable way.

Features	Values
color	Green, Yellow, Red
Size	Small, medium, big
Shape,	Round, non- Round
Taste	Sweet, non- Sweet

Classes :

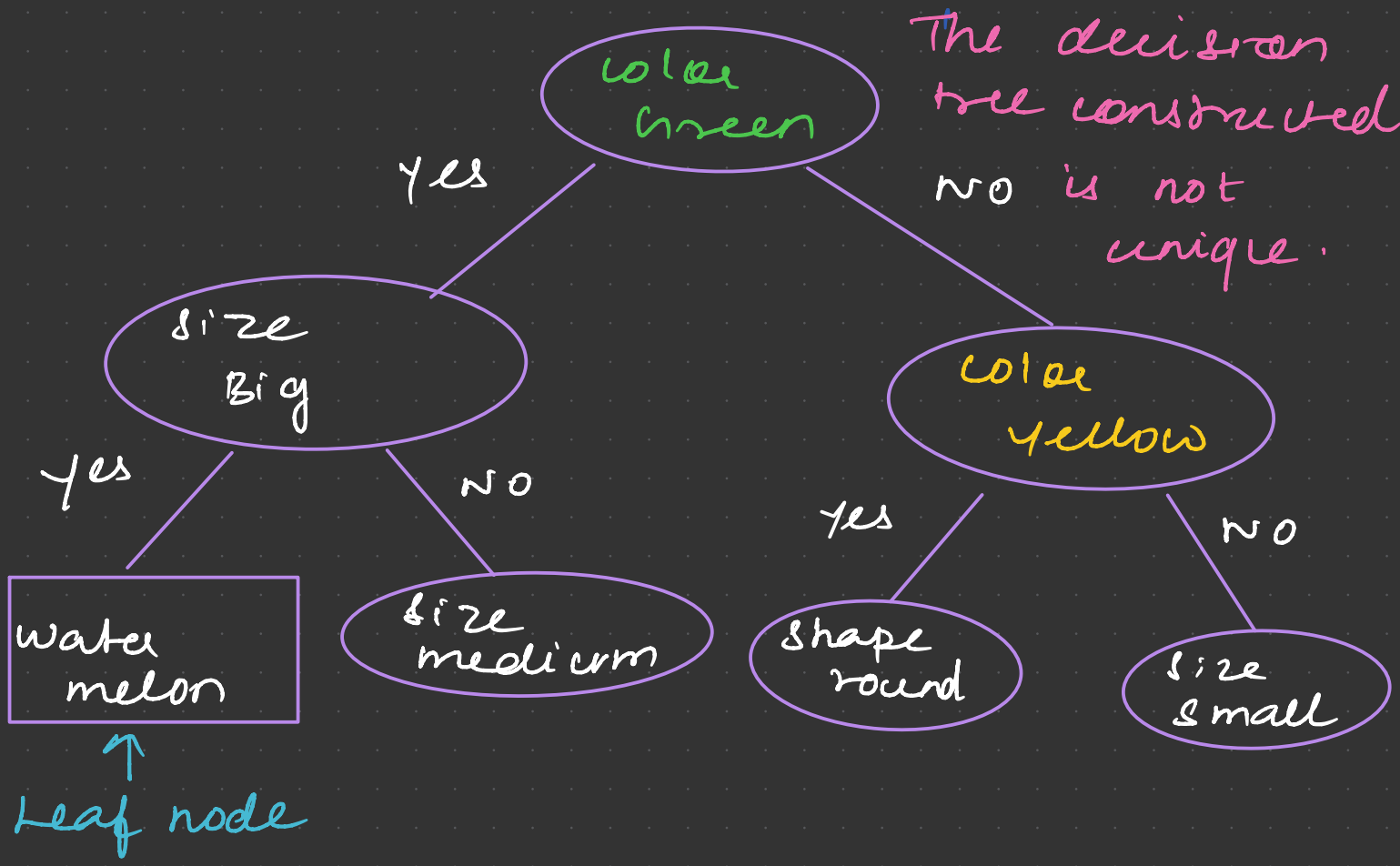
watermelon, Apple, Grape, Banana, Lemon, Cherry, Grapefruit.

* each fruit is described using words, not numbers
* Not all the features are used for every fruit
* The same fruit can have multiple description

Example:

* watermelon → Green, Big
* Banana → Yellow, non-round
* Lemon → Yellow, round, not big (small or medium)
* Grapefruit → Yellow, round, big
* Apple → Green, red, medium
* Cherry → Small, red, sweet
* Grape → Small, black, not-sweet.

* This creates vagueness and variation in data.



The decision tree constructed No is not unique.

** This tree grows until its depth.

In decision tree

→ At every node we ask a query or question

→ Based on the answer tree is traversed down to reach a leaf node.

→ leaf node is associated with a class.

→ Decision Tree can also be built for numerical feature.

* No limit to ask question on one feature node.

→ can combine more than one feature.

* Number of questions asked is very large in number

* out of several questions, which questions to be asked at a particular node is the issue to be addressed.

* This issue to be addressed is constructing a decision tree.

Classification and Regression Tree (CART)

* CART provides framework to construct a decision tree.

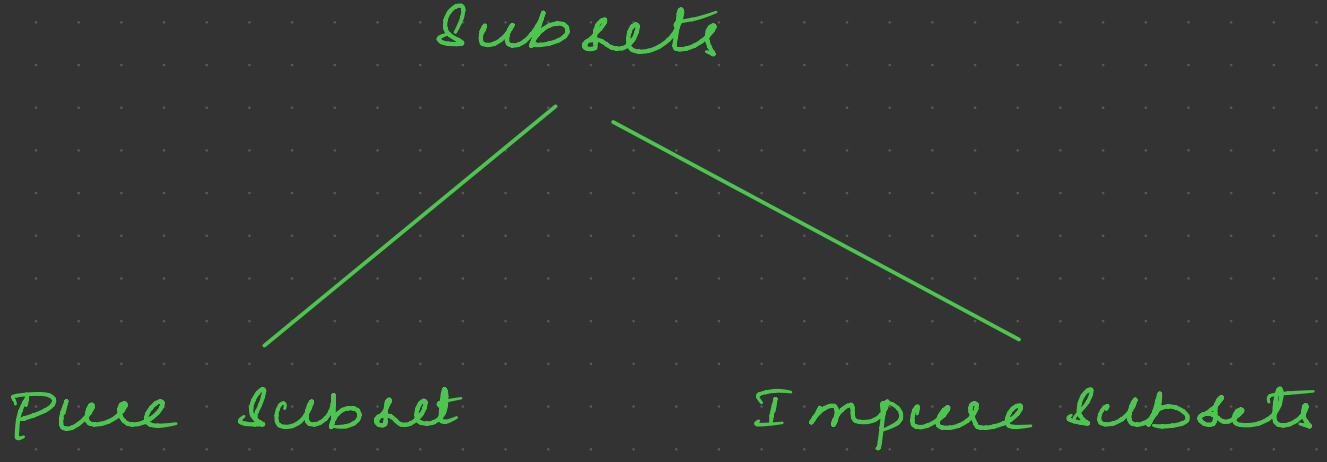
* works for
  - → classification (class labels)
  - → Regression (numerical output)

How CART Builds the Tree

* The decision tree starts with all training data at the root node.

* At each node:
  - → A question (query) is asked
  - → The data is splited into smaller subsets

* Each subset sent to a child node.



Pure subsets:

All examples in the subsets have the same class label.

Pure subsets are called as pure nodes.

Impure subset

It includes examples of more than one classes

A node with impure subsets are called as impure nodes.

* Quantification of impurity :

Impurity measure.

Impurity measure, will be used to construct the decision tree to decide which query need to be processed at each of the node.

Splitting at one node should decrease the impurity from one level to the next level.

Goal of CART

* Keep splitting the data such that
  * Nodes becomes more pure
  * Tree Becomes simple and compact

To summarize, CART is a framework used to construct decision trees by recursively splitting the training dataset into smaller subsets to form a pure nodes for classification or regression.

Main issues in CART approach:

1) Number of split at a node.

→ A node can split into

* Two Branches (Binary split)
  ↳ B = 2
* Three Branches (Ternary split)
  ↳ B = 3
* or more (Multi-way split)

→ Branching Factor (B)

* Number of branches coming out of a node.

→ In practice:

* Binary split is most commonly used

* Makes the tree simpler and easier to control

2) Query selection at a Node

→ Many possible questions can be asked at a node.

→ The challenge is

> which question should be selected.

→ Rule used:

* Occam's Razor

> The simplest tree that explains the data is preffered.

→ Goal

* Choose the question that
  > makes child nodes less impure
  > leads to a small and compact tree.

3) When should we stop splitting

* If we keep splitting
  > Tree becomes very large
  > model may overfit
  > classification becomes slower
* So we need a stopping condition

4) Assigning class label to impure leaf nodes

* Sometimes a leaf nodes still has
  > samples from multiple class
* Then we must decide :
  > which class label to assign to that leaf itself

CART must decide

* How many branches to create
* which question to ask
* when to stop growing the tree
* what label to give at the end.

CART : Measure of Impurity

* Impurity measure how mixed the classes are at a node
* It helps to decide
  → which questions (query) should be asked at a node
* the goal is to build a simple and compact decision tree.

Notation

* Let a node be called N_n (N_d)
* The impurity at the node is written as

$$i(N_d)$$

Meaning of impurity values

i(Nd) = 0

* Node is pure
* All samples belong to the same class

i(Nd) is large

* Node is impure
* samples are evenly distributed among multiple classes.

Why Impurity is important?

When a node is split

* The impurity of child nodes should be lower
* This means split is good

CART always tries to:

* Reduce impurity from parent node to child nodes.

Entropy (Information Impurity)

Entropy is the most commonly used impurity measure

It measures the uncertainity or randomness in the distribution

Formula

$$i(Nd) = -\sum_{j=1}^{M} P(C_j) \log_2 P(C_j)$$

Terms,

$M \rightarrow$ Number of classes

$P(C_j) \rightarrow$ Fraction of samples at node Nd that belongs to C_j

Interpretation

* More class mixing $\rightarrow$ higher entropy
* more class purity $\rightarrow$ lower entropy.

Summary

* If all data at a node belongs to one class → Entropy = 0
* If data is evenly split across classes → Entropy is maximum
* CART chooses splits that reduce entropy the most.

Gini Impurity

* Gini Impurity is a measure of how impure the classes are at a node
* It is an alternative to Entropy
* commonly used in CART and Practical implementation

Gini impurity measures the probability of incorrect classification of randomly choosen samples from the node, if its labeled according to the class distribution at the node

Formula

$$i(N_a) = 1 - \sum_{j=1}^{M} P(C_j)^2$$

Entropy v Gini

* Both measures Impurity
* Both aims to
  → select the split that makes child nodes more pure
* Gini is :
  → faster to compute
  → often used in real-world tree implementation

Summary

* If most samples belongs to one class → Gini is low
* If samples are evenly spread across classes → Gini is high
* CART chooses the split that reduces Gini impurity the most.

CART: Query selection at a Node and splitting - construction of Decision tree

* At each node, many possible questions (queries) can be asked
* CART evaluates all possible queries
* Goal is to select the query that:

  Reduces the impurity the most.

* Generally, impurity level of the child nodes should be smaller.
* The difference between the impurity level at a node (Nd) and its child node is important.

Basic Idea

* A node contains a subset of training data
* when a query is applied
  → data is split into child nodes
* The child nodes should be more pure than the parent node.

Binary split

This is a most common case

* Let the current node be Nd
* After applying the query T, it splits into:
  → Nd L → Left child
  → Nd R → Right child.

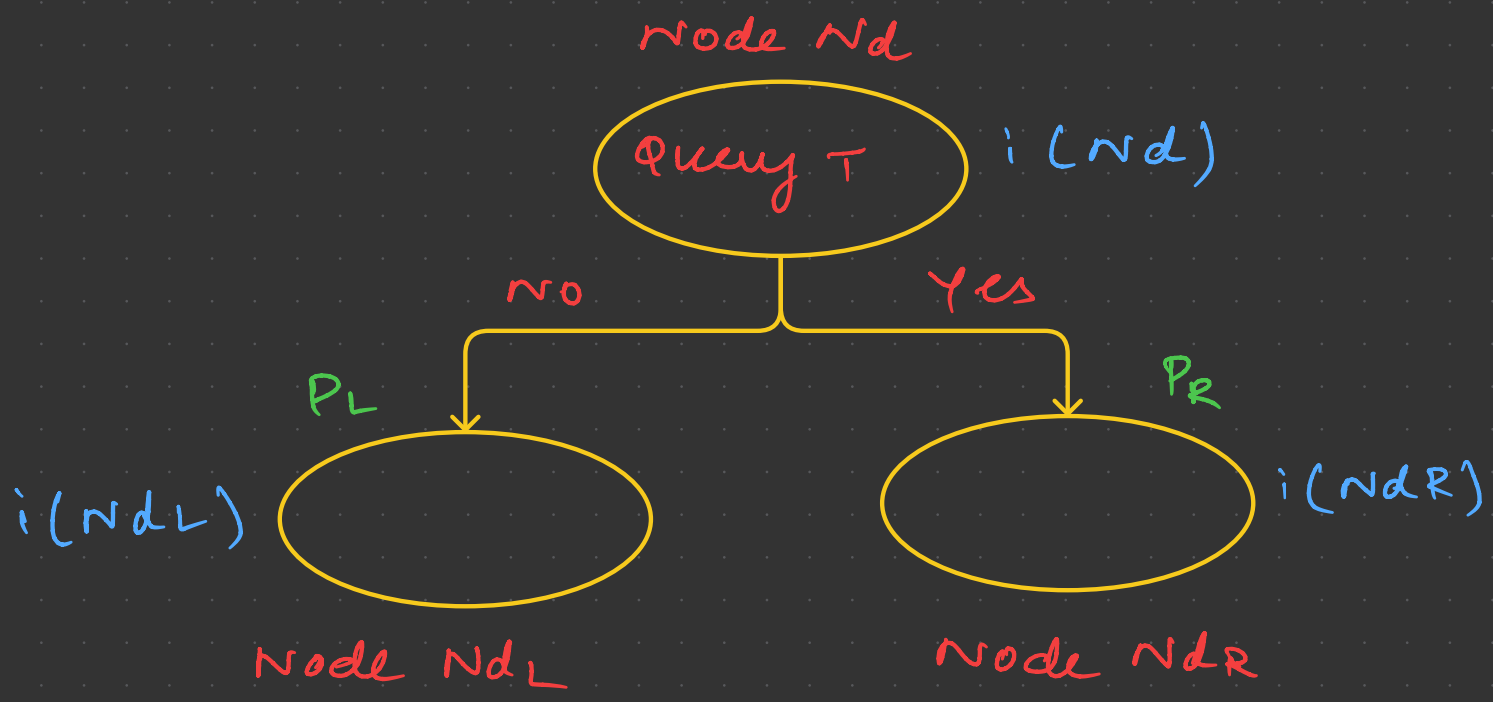
Fraction of Data

* P_L → Fraction of samples that goes to Nd_L

* P_R → Fraction of samples that goes to Nd_R

* Relationship of P_R → Since we have only two nodes.

$$P_R = 1 - P_L$$



Impurity terms

* $i(Nd)$ → Impurity of Parent node
* $i(NdL)$ → Impurity of left child
* $i(NdR)$ → Impurity of right child

Change in Impurity (Information Gain)

The decrease in impurity caused by query T is

$$\Delta i(Nd) = i(Nd) - (P_L \cdot i(NdL) + P_R \cdot i(NdR))$$

Decision Rule

* compute $\Delta i(Nd)$ for all possible queries
* Select the query which has maximum $\Delta i(Nd)$
* This query gives the best split.

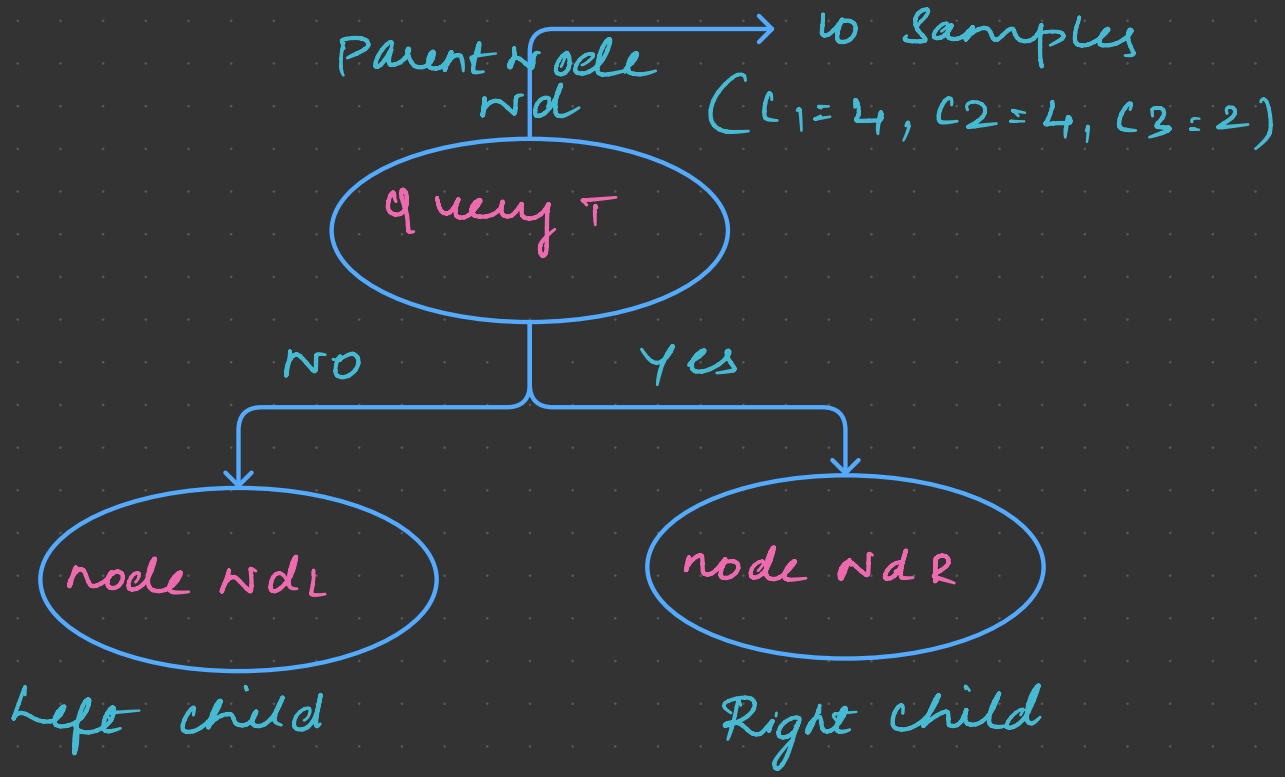
Change in the impurity level is called as Information Gain → when entropy impurity is used.

Illustration

How to calculate impurity and information gain using numbers and how CART chooses the best query.

Given data at node Nd

* The total number of samples is 10
* Class distribution
  * C1 → 4 samples
  * C2 → 4 samples
  * C3 → 2 samples.

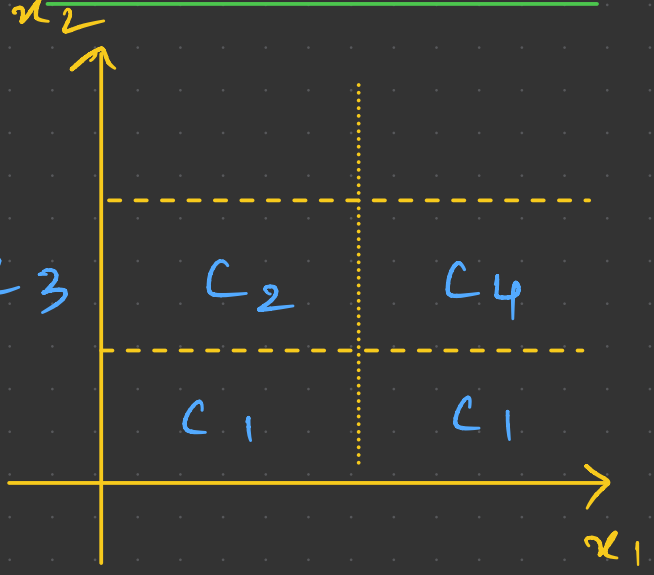


with tree structure

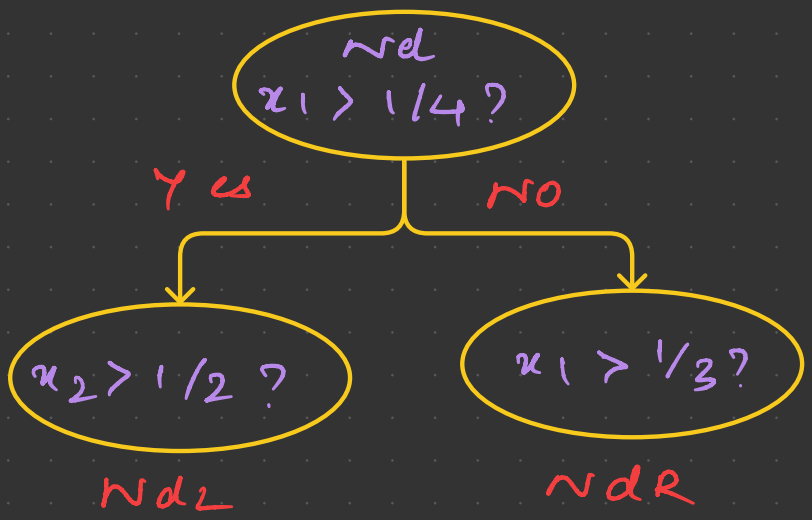
with both :

* The tree structure (Nd → NdL / NdR)
* feature space (x_1, x_2 → Region split)

Feature space view :



Tree view :



Step 1 – Impurity at parent node (Nd)

using Entropy

$$i(Nd) = -\sum P(C_j) \log_2 P(C_j)$$

Probabilities

* $P(C_1) = 4/10$
* $P(C_2) = 4/10$
* $P(C_3) = 2/10$

here,

* number of classes $M = 3$
* we must compute entropy for C_1, C_2, C_3

compute probabilities

$$P(C_1) = 4/10 = 0.4$$

$$P(C_2) = 4/10 = 0.4$$

$$P(C_2) = 2/10 = 0.2$$

substitute in the formula

$$i(Nrd) = -\left[0.4 \log_2(0.4) + 0.4 \log_2(0.4) + 0.2 \log_2(0.2)\right]$$

compute log values

$$\log_2(0.4) \simeq -1.3219$$

$$\log_2(0.2) \simeq -2.3219$$

Multiply Terms

$0.4\,(-1.3219) = -0.5288$

$0.2\,(-2.3219) = -0.4644$

therefore,

$i(Nd) = -0.5288 + (-0.5288) + (-0.4644)$

$i(Nd) = -(-1.522)$

$$i(Nd) \simeq 1.522$$

→ This node is impure because sample belongs to more classes.

Query 1 — First possible split

Left child (NdL)

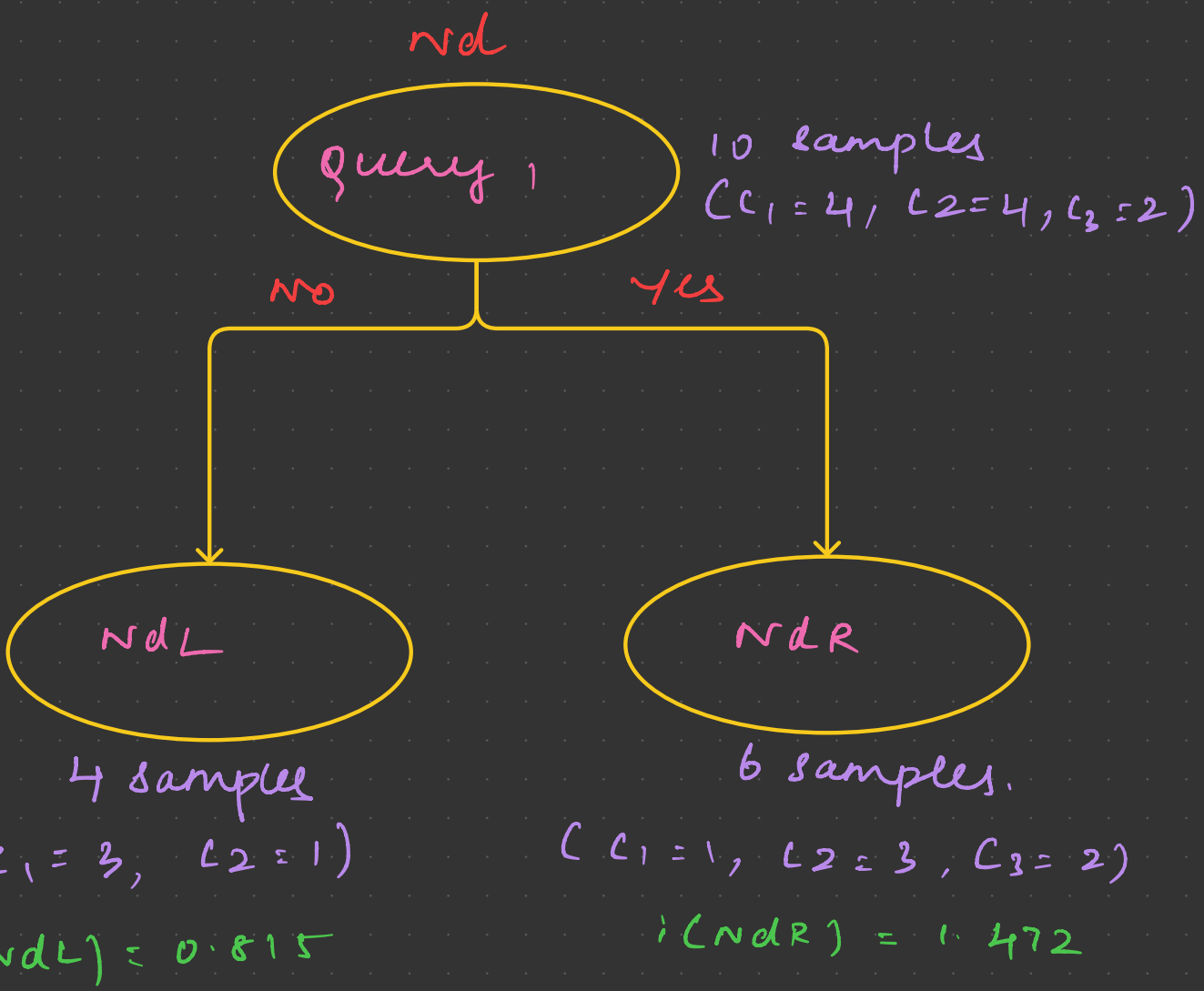
* Total samples = 4
* $C1 = 3$, $C2 = 1$

$$i(NdL) = 0.815$$

Right child (NdR)

* Total samples = 6
* $C_1 = 1$, $C_2 = 3$, $C_3 = 2$

$i(NdR) = 1.472$



Step 2 - Information gain for query 1

Fractions,

* $P_L = 4/10$
* $P_R = 6/10$

Formula

$$\Delta i(Nd) = -i(Nd) - (P_L \cdot i(N_dL) + P_R \cdot i(N_dR))$$

Result

$$\Delta i(Nd) = 0.315$$

Query 2 - second possible split

Left child (N_dL)

* Total samples = 3
* $C_1 = 2$, $C_3 = 1$

$$i(N_dL) = 0.918$$

Right child (N_dR)

* Total samples = 7
* $C_1 = 1$, $C_2 = 4$, $C_3 = 2$

$$i(N_dR) = 1.379$$

Step 3 - Information Gain for Query 2

$$\Delta i(Nd) = 0.280$$

Decision

Compare,

→ Query 1 = 0.135

→ Query 2 = 0.280

from this observation **Query 1** will be choosen because it gives the maximum impurity reduction.

Multiway split

* In multiway split, a node can split into more than two branches.
* Each possible answer to a query creates one child node.
* The number of branches is called as Branching Factor (B)

Branching Factor (B)

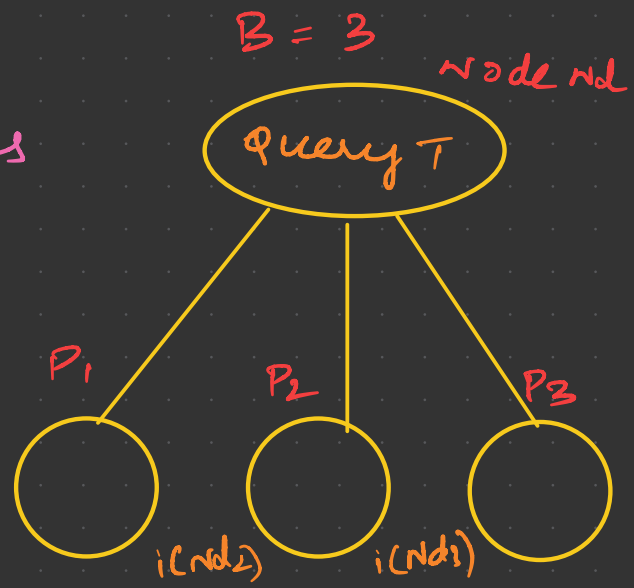
* B = number of child nodes

* Examples

  → Binary split (B = 2)

  → Ternary split (B = 3)

  → Multiway split (B > 2)



How the split works

Parent Node

* Let the parent node Nd

* It contains a subset of training samples.

After Query T

* The nodes splits into

  → $Nd_1, Nd_2, Nd_3 \ldots, Nd_n$

* Each child gets a fraction of samples.

Fraction of Samples

P_K → fraction of samples that can go to the K th child node

Sum of all fractions

$$P_1 + P_2 + \cdots + P_n = 1$$

Impurity Reduction

Impurity terms

$i(Nd)$ → Impurity of parent node

$i(Ndk)$ → Impurity of k-th child node

Formula

$$\Delta i(Nd) = i(Nd) - \sum_{k=1}^{B} P_k \cdot i(Ndk)$$

Decision Rule

* Try all possible queries
* For each query
  → compute $\Delta i(Nd)$
* Select the query with the maximum $\Delta i(Nd)$

Splitting Process

* Tree construction starts from the root node
* At each node, CART:
  - → Test all possible queries
  - → Measures impurity reduction
  - → chooses the best query.

Step by Step process

Step 1 — Start at Root Node

* All training samples are placed at the root node (Nd)
* This node usually impure.

Step 2 - Select Best Query

* Try many queries (questions)
* For each query
  - → Split the data
  - → compute $\Delta i(Nd)$ (impurity reduction)
* Choose the query with maximum $\Delta i(Nd)$

Step 3 - Create child Nodes

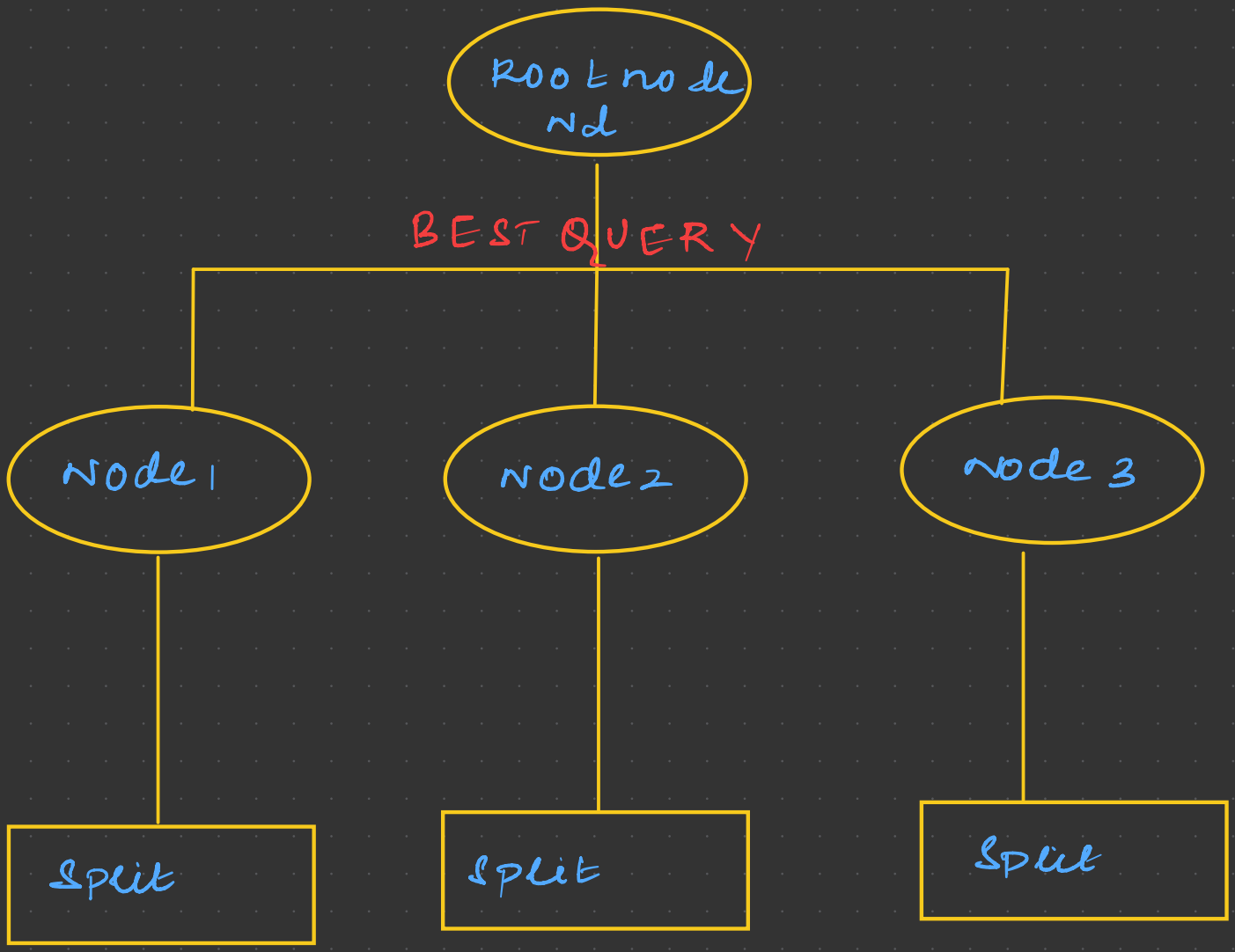
* The selected query splits into
  → Nd_1, Nd_2, Nd_3 ..... Nd_n (child nodes)
* Each child gets subset of data

Step 4 - Repeat Recursively

* For each child node:
  → Treat it as a new parent
  → Again select the best query
  → Split it further.

Step 5 - Stop at Leaf Nodes

* The process continues until:
  → Node becomes pure
  or
  → A stopping condition is met



Summary

→ Start at the top
→ Ask the best question
→ Split the data
→ Repeat for every branch
→ Stop when no split is possible or stopping condition is met.

Criterion to Stop Splitting

* CART must decide when to stop growing the decision tree so that:

→ The tree does not become too large

→ The model does not overfit

→ The tree remains simple and compact

Stopping Criterion 1 — Impurity reduction Threshold

Rule

* At a node nd, compute the maximum impurity reduction

$$\Delta i(nd)$$

* If this value is below a small threshold, then stop splitting

Typical threshold values

→ 10^{-2} or 10^{-3}

Meaning,

If no query can reduce impurity significantly — Further splitting is not useful

Stopping Criteria - 2 — Tree Size control (Regularization)

Rule

Stop splitting if

$$\sum_{nd \in SL} i(nd) + \alpha \times (\text{Number of nodes in tree})$$

$SL \rightarrow$ Set of all Leaf Nodes

$i(nd) \rightarrow$ Impurity of each leaf node

$\alpha \rightarrow$ Hyperparameter that controls Penalty for tree nodes

Meaning

* First term → Measures total impurity
* Second term → Penalizes large trees
* Together → Prefer small, low impurity trees

Why this is Important

→ Prevents the tree from growing too deep

→ Reduce Overfitting

→ Improves generalization on new data.

Assignment of class label to leaf Node

After the decision tree is constructed CART must decide what class label to assign to each leaf node

case 1 – Pure leaf Node

A leaf node is pure if

→ All Training samples that reach this node belongs to one class only

Rule

Assign the leaf node the same class label

for example,

> If leaf node :
>
> $C_1 = 5$, $C_2 = 0$, $C_3 = 0$
>
> then
>
> Label = C_1

Case 2 - Impure Leaf Node

A leaf node is impure if :

→ samples belongs to more than one classes

Rule

* Find the class with the largest number of samples
* Assign that class as the leaf node label

for example,

If at a leaf node

$C_1 = 2, \quad C_2 = 5, \quad C_3 = 3$

Then

Label = C_2 (majority class)

Why this needed ?

* sometimes CART stops splitting early
* so leaf nodes may still contain mixed classes
* we still need a final decision rule.