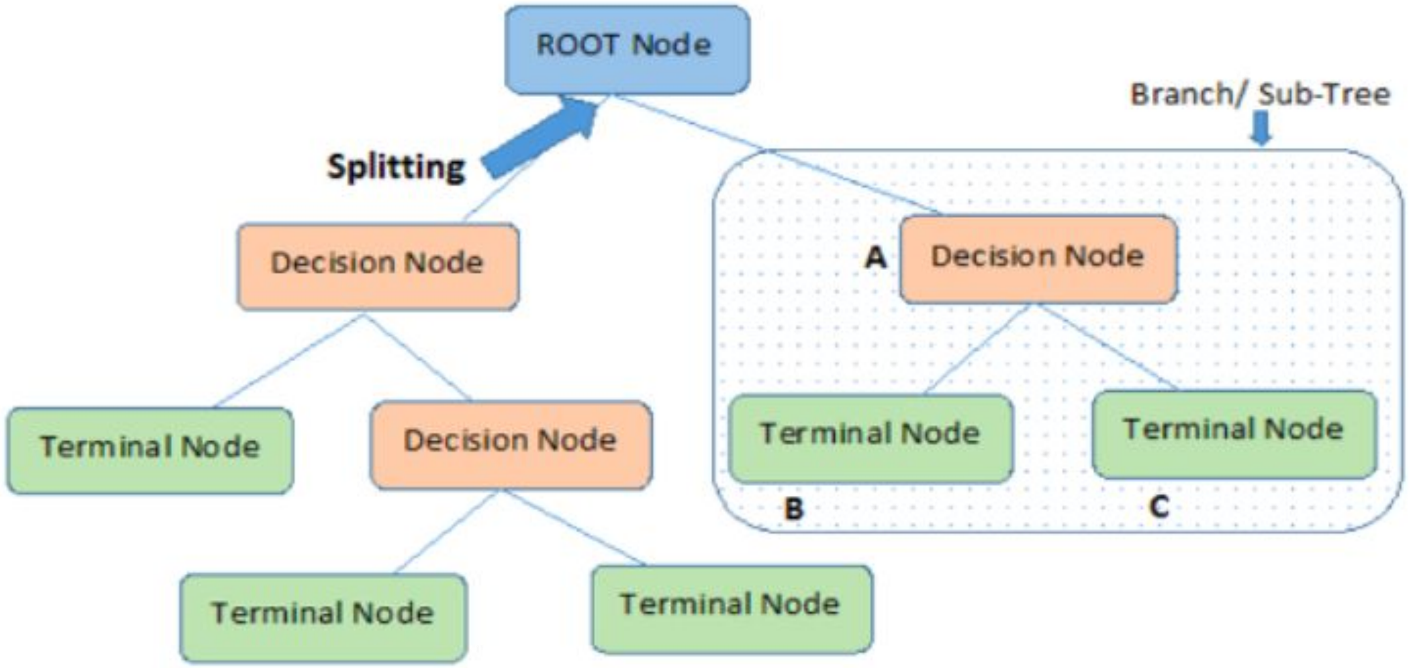
Decision Trees and CART

By Sushmithaa P
ML and SP student coordinator, Tech Club ECE

What is a Decision Tree?

- A decision tree is a popular machine learning algorithm that can be used for both regression and classification tasks
- It is a non-parametric supervised learning algorithm
- It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes
- It starts with a root node and ends with a decision made by leaves.

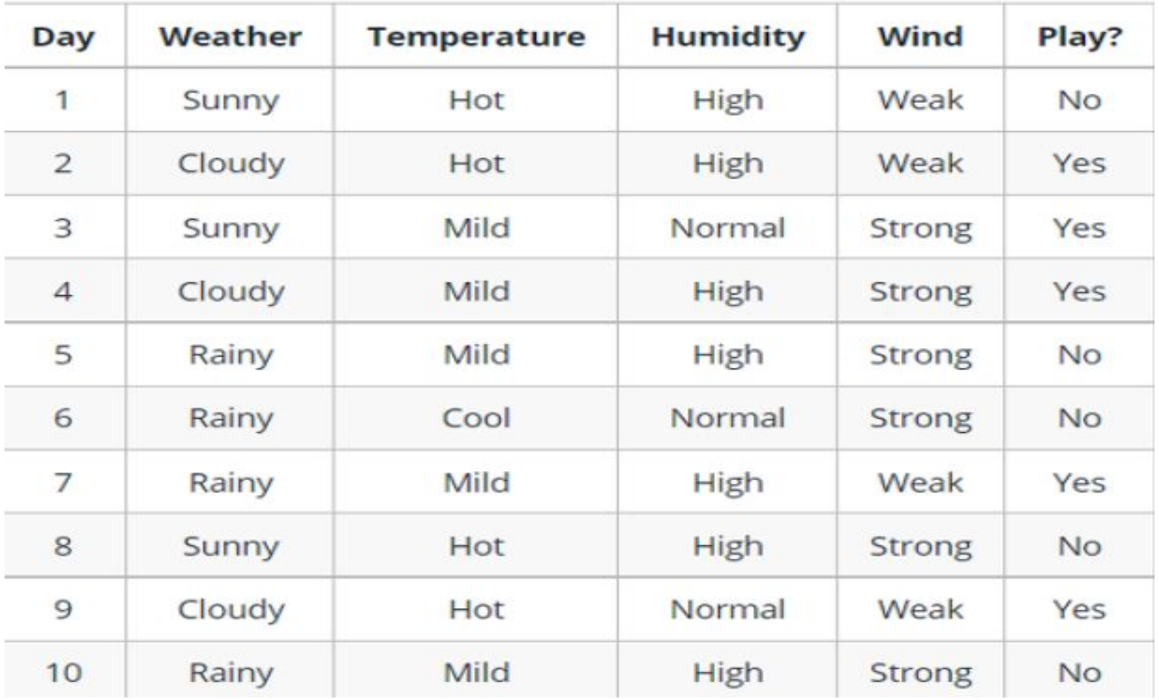
DECISION TREE STRUCTURE



Decision Tree Terminologies

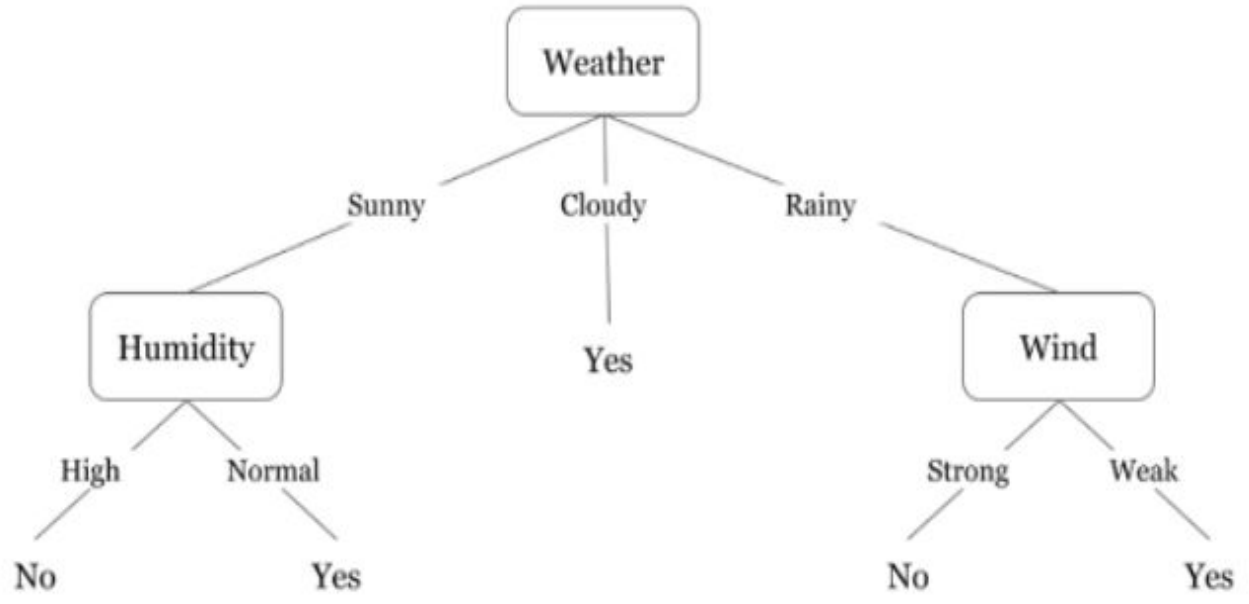
- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

Example



Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

Decision tree diagram



Assumptions while creating Decision Tree

In the beginning, the whole training set is considered as the **root.**

Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.

Records are **distributed recursively** on the basis of attribute values.

Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

How to choose the best attribute at each node ?

- Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes.
- The creation of sub-nodes increases the homogeneity of resultant sub-nodes.
- In other words, we can say that the purity of the node increases with respect to the target variable.
- The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

Attribute Selection Measures

- If the dataset consists of **N** attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step.
- By just randomly selecting any node to be the root can't solve the issue.
- If we follow a random approach, it may give us bad results with low accuracy
- For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some *criteria* like :

  Entropy,

  Information gain,

  Giny impurity

Entropy

Entropy:

Entropy is the measure of the degree of randomness or uncertainty in the dataset. Suppose you have a group of friends who decides which movie they can watch together on Sunday. There are 2 choices for movies, one is *"Lucy"* and the second is *"Titanic"* and now everyone has to tell their choice. After everyone gives their answer we see that *"Lucy" gets 4 votes* and *"Titanic" gets 5 votes*. Which movie do we watch now? Isn't it hard to choose 1 movie now because the votes for both the movies are somewhat equal.

This is exactly what we call disorderness, there is an equal number of votes for both the movies, and we can't really decide which movie we should watch. It would have been much easier if the votes for "Lucy" were 8 and for "Titanic" it was 2. Here we could easily say that the majority of votes are for "Lucy" hence everyone will be watching this movie.

In a decision tree, the output is mostly "yes" or "no"

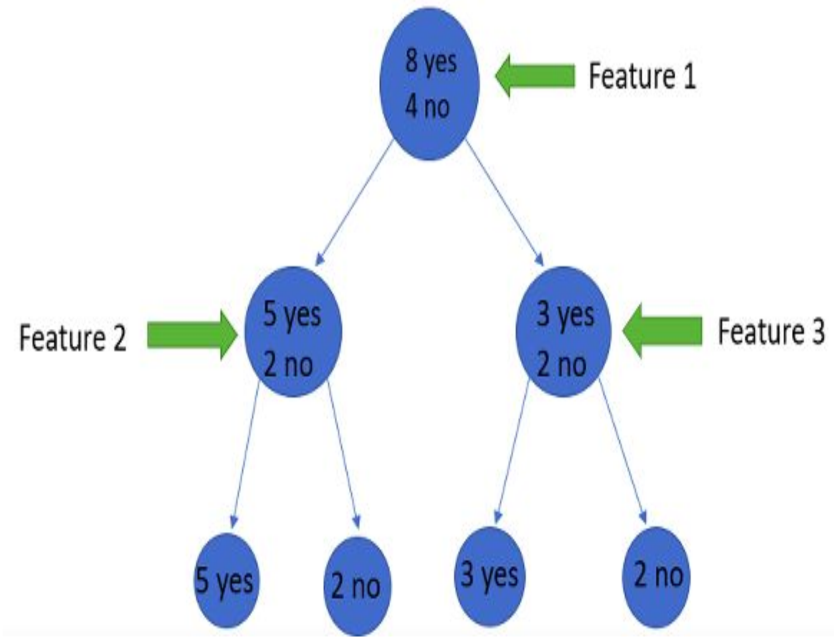
$$E(S) = -p_{(+)}\log p_{(+)} - p_{(-)}\log p_{(-)}$$

Entropy basically measures the impurity of a node. Impurity is the degree of randomness; it tells how random our data is. A pure sub-split means that either you should be getting “yes”, or you should be getting “no”.

Suppose *feature 1* has **8 “yes” and 4 “no”** initially, after the first split the left node *gets* ***5 ‘yes’ and 2 ‘no’*** whereas right node *gets* ***3 ‘yes’ and 2 ‘no’.***

We see here the split is not pure, why? Because we can still see some negative classes in both the nodes.

In order to make a decision tree, we need to calculate the impurity of each split, and when the purity is 100%, we make it as a leaf node.



Higher the entropy, the lower will be the purity and the higher will be the impurity.

The goal of machine learning is to decrease the uncertainty or impurity in the dataset, here by using the entropy we are getting the impurity of a particular node,

For feature 2

$$\Rightarrow -\left(\frac{5}{7}\right)log_2\left(\frac{5}{7}\right) - \left(\frac{2}{7}\right)log_2\left(\frac{2}{7}\right)$$

$$\Rightarrow -(0.71 * -0.49) - (0.28 * -1.83)$$

$$\Rightarrow -(-0.34) - (-0.51)$$

$$\Rightarrow 0.34 + 0.51$$

$$\Rightarrow 0.85$$

Information Gain

Information gain measures the reduction of uncertainty given some feature and it is also a deciding factor for which attribute should be selected as a decision node or root node.

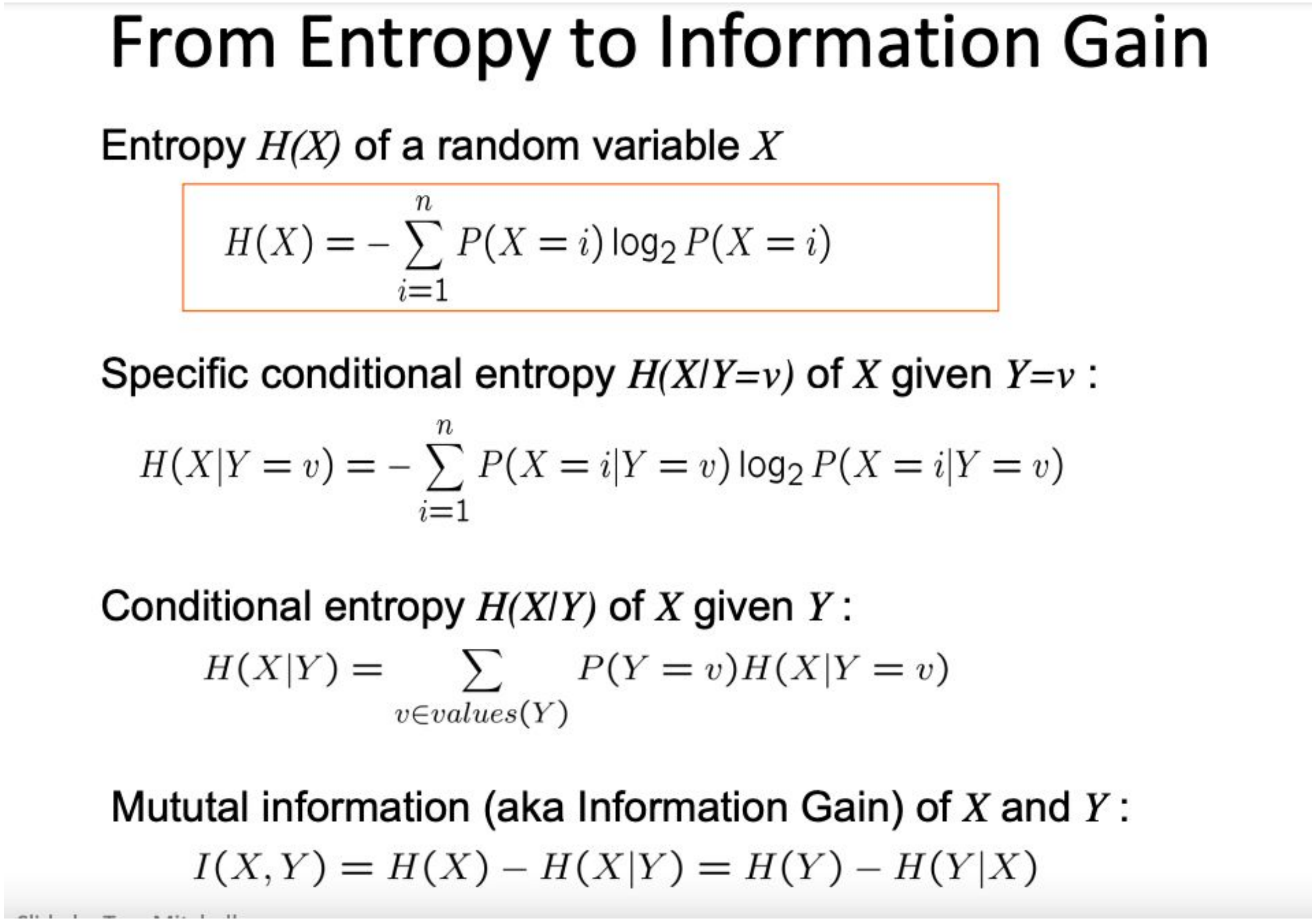
$$Information\ Gain\ =\ E(Y)\ -\ E(Y|X)$$

It is just entropy of the full dataset – entropy of the dataset given some feature.

• We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.

• Information gain tells us how important a given attribute of the feature vectors is.

• We will use it to decide the ordering of attributes in the nodes of a decision tree.



From Entropy to Information Gain

Entropy *H(X)* of a random variable *X*

$$\boxed{H(X) = -\sum_{i=1}^{n} P(X=i)\log_2 P(X=i)}$$

Specific conditional entropy *H(X|Y=v)* of *X* given *Y=v* :

$$H(X|Y=v) = -\sum_{i=1}^{n} P(X=i|Y=v)\log_2 P(X=i|Y=v)$$

Conditional entropy *H(X|Y)* of *X* given *Y* :

$$H(X|Y) = \sum_{v \in values(Y)} P(Y=v)H(X|Y=v)$$

Mututal information (aka Information Gain) of *X* and *Y* :

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Example :

Suppose our entire population has a total of 30 instances. The dataset is to predict whether the person will go to the gym or not. Let’s say 16 people go to the gym and 14 people don’t

Now we have two features to predict whether he/she will go to the gym or not.

- Feature 1 is “Energy” which takes two values *“high” and “low”*
- Feature 2 is “Motivation” which takes 3 values *“No motivation”, “Neutral” and “Highly motivated”.*

Let’s see how our decision tree will be made using these 2 features. We’ll use information gain to decide which feature should be the root node and which feature should be placed after the split.

Let's calculate the entropy

$$E(Parent) = -\left(\frac{16}{30}\right)\log_2\left(\frac{16}{30}\right) - \left(\frac{14}{30}\right)\log_2\left(\frac{14}{30}\right) \approx 0.99$$

$$E(Parent|Energy = \text{"high"}) = -\left(\frac{12}{13}\right)\log_2\left(\frac{12}{13}\right) - \left(\frac{1}{13}\right)\log_2\left(\frac{1}{13}\right) \approx 0.39$$

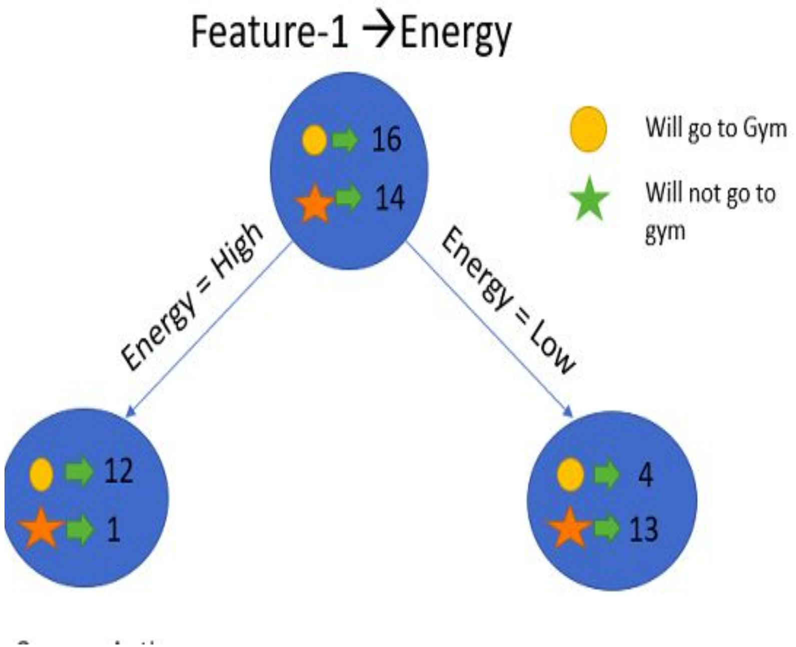
$$E(Parent|Energy = \text{"low"}) = -\left(\frac{4}{17}\right)\log_2\left(\frac{4}{17}\right) - \left(\frac{13}{17}\right)\log_2\left(\frac{13}{17}\right) \approx 0.79$$

To see the weighted average of entropy of each node we will do as follows:

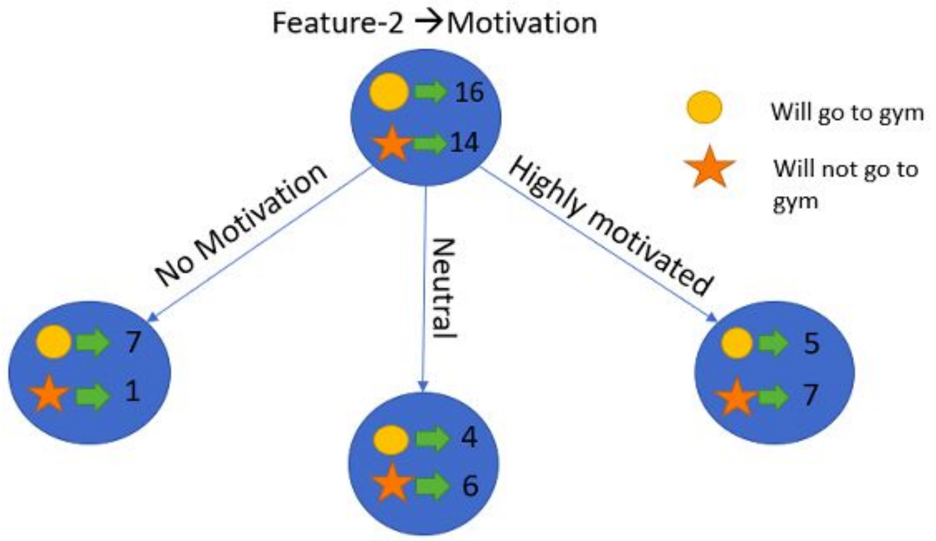
$$E(Parent|Energy) = \frac{13}{30} * 0.39 + \frac{17}{30} * 0.79 = 0.62$$

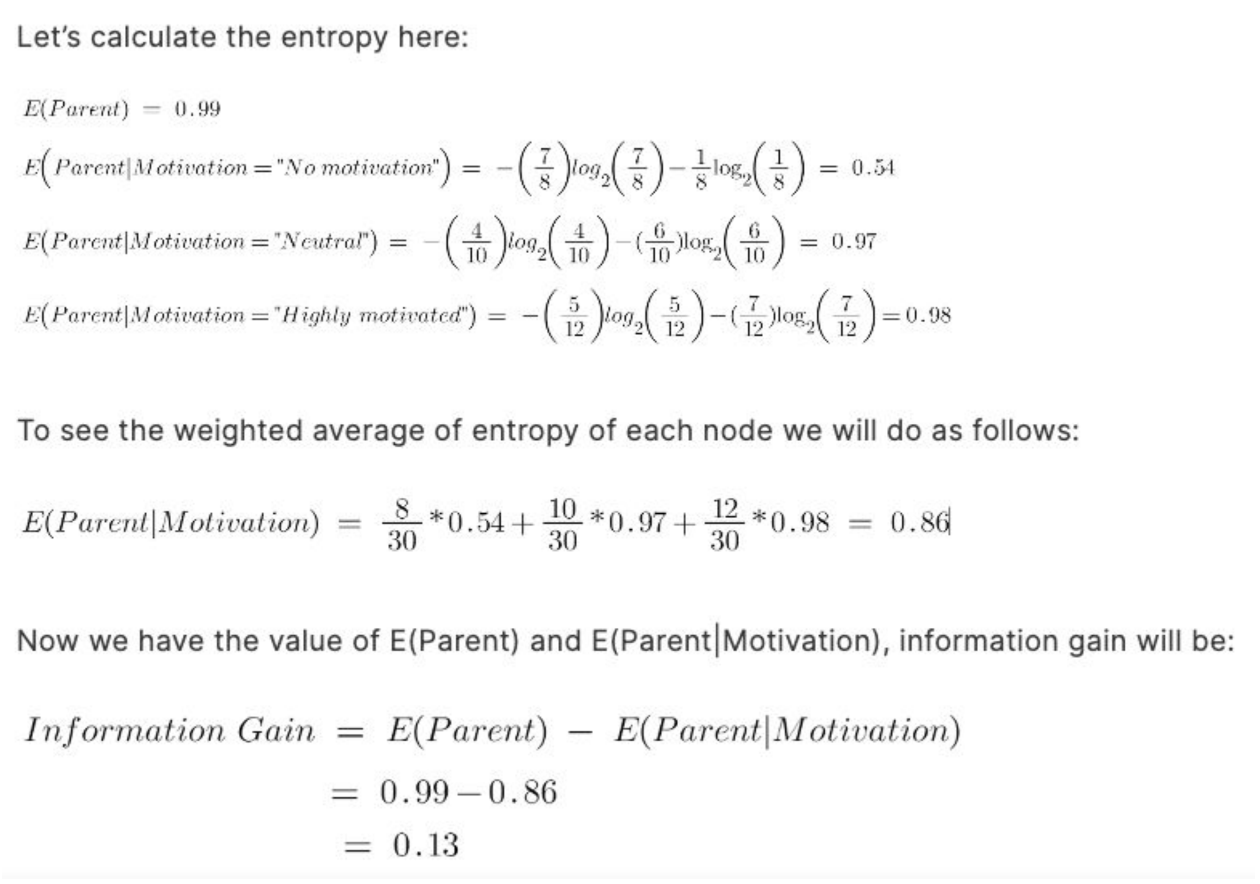
Now we have the value of E(Parent) and E(Parent|Energy), information gain will be:

$$\begin{aligned} Information\ Gain &= E(parent) - E(parent|energy) \\ &= 0.99 - 0.62 \\ &= 0.37 \end{aligned}$$



Similarly, we will do this with the other feature “Motivation” and calculate its information gain.





Let's calculate the entropy here:

$$E(Parent) = 0.99$$

$$E\left(Parent|Motivation = \text{"No motivation"}\right) = -\left(\frac{7}{8}\right)log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) = 0.54$$

$$E(Parent|Motivation = \text{"Neutral"}) = -\left(\frac{4}{10}\right)log_2\left(\frac{4}{10}\right) - (\frac{6}{10})\log_2\left(\frac{6}{10}\right) = 0.97$$

$$E(Parent|Motivation = \text{"Highly motivated"}) = -\left(\frac{5}{12}\right)log_2\left(\frac{5}{12}\right) - (\frac{7}{12})\log_2\left(\frac{7}{12}\right) = 0.98$$

To see the weighted average of entropy of each node we will do as follows:

$$E(Parent|Motivation) = \frac{8}{30} * 0.54 + \frac{10}{30} * 0.97 + \frac{12}{30} * 0.98 = 0.86$$

Now we have the value of E(Parent) and E(Parent|Motivation), information gain will be:

$$\begin{aligned} Information\ Gain &= E(Parent) - E(Parent|Motivation) \\ &= 0.99 - 0.86 \\ &= 0.13 \end{aligned}$$

- We now see that the “Energy” feature gives more reduction which is 0.37 than the “Motivation” feature. Hence we will select the feature which has the highest information gain and then split the node based on that feature.
- Our parent entropy was near 0.99 and after looking at this value of information gain, we can say that the entropy of the dataset will decrease by 0.37 if we make “Energy” as our root node.
- we will select the feature which has the highest information gain and then split the node based on that feature.
- In this example “Energy” will be our root node and we’ll do the same for sub-nodes

Classification And Regression Trees

CART(Classification And Regression Trees) is a variation of the decision tree algorithm. It can handle both classification and regression tasks.

CART(Classification And Regression Tree) for Decision Tree
CART is a predictive algorithm used in Machine learning and it explains how the target variable's values can be predicted based on other matters. It is a decision tree where each fork is split into a predictor variable and each node has a prediction for the target variable at the end.

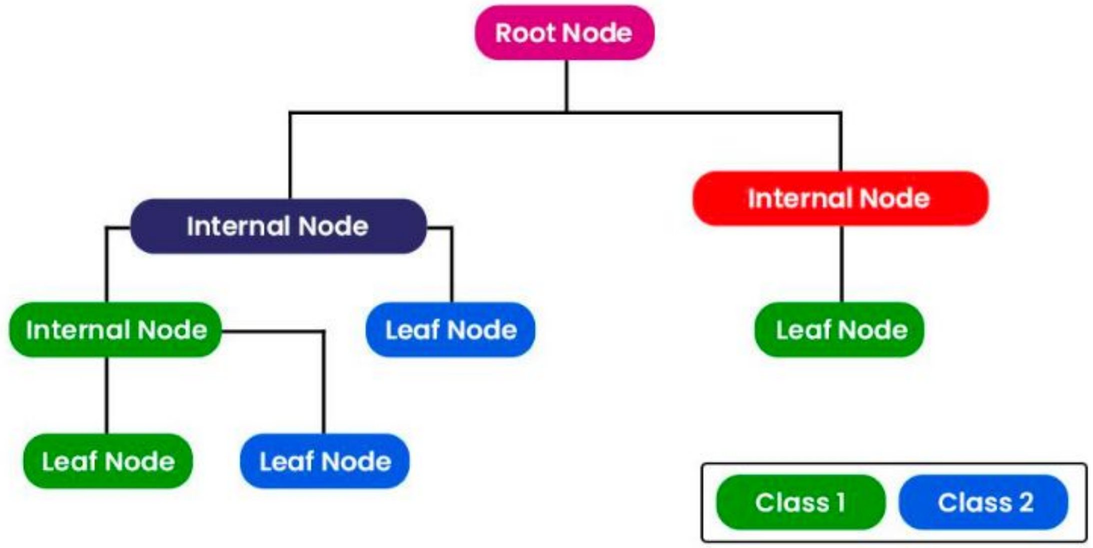
The term CART serves as a generic term for the following categories of decision trees:

- **Classification Trees:** The tree is used to determine which "class" the target variable is most likely to fall into when it is continuous.
- **Regression trees:** These are used to predict a continuous variable's value.

- **Tree structure:** CART builds a tree-like structure consisting of nodes and branches. The nodes represent different decision points, and the branches represent the possible outcomes of those decisions. The leaf nodes in the tree contain a predicted class label or value for the target variable.
- **Splitting criteria:** CART uses a greedy approach to split the data at each node. It evaluates all possible splits and selects the one that best reduces the impurity of the resulting subsets. For classification tasks, CART uses Gini impurity as the splitting criterion. The lower the Gini impurity, the more pure the subset is. For regression tasks, CART uses residual reduction as the splitting criterion. The lower the residual reduction, the better the fit of the model to the data.
- **Pruning:** To prevent overfitting of the data, pruning is a technique used to remove the nodes that contribute little to the model accuracy. Cost complexity pruning and information gain pruning are two popular pruning techniques. Cost complexity pruning involves calculating the cost of each node and removing nodes that have a negative cost. Information gain pruning involves calculating the information gain of each node and removing nodes that have a low information gain.

The CART algorithm works via the following process:

- The best-split point of each input is obtained.
- Based on the best-split points of each input in Step 1, the new "best" split point is identified.
- Split the chosen input according to the "best" split point.
- Continue splitting until a stopping rule is satisfied or no further desirable splitting is available.



CART ALGORITHM

- The CART (Classification and Regression Trees) algorithm is a decision tree-based algorithm that can be used for both classification and regression problems in machine learning.
- It works by recursively partitioning the training data into smaller subsets using binary splits and creates a binary tree .
- The algorithm works by recursively partitioning the training data into smaller subsets using binary splits.
- The tree starts at the root node, which contains all the training data, and recursively splits the data into smaller subsets until a stopping criterion is met (such as max depth of the tree, etc)

- At each node of the tree, the algorithm selects a feature and a threshold that best separates the training data into two groups, based on the values of that feature. This is done by choosing the feature and threshold that maximizes the information gain or the Gini impurity
- Once the tree is built, it can be used to make predictions by traversing the tree from the root node to a leaf node that corresponds to the input data.
- For regression problems, the prediction is the average of the target values in the leaf node.
- For classification problems, the prediction is the majority class in the leaf node.