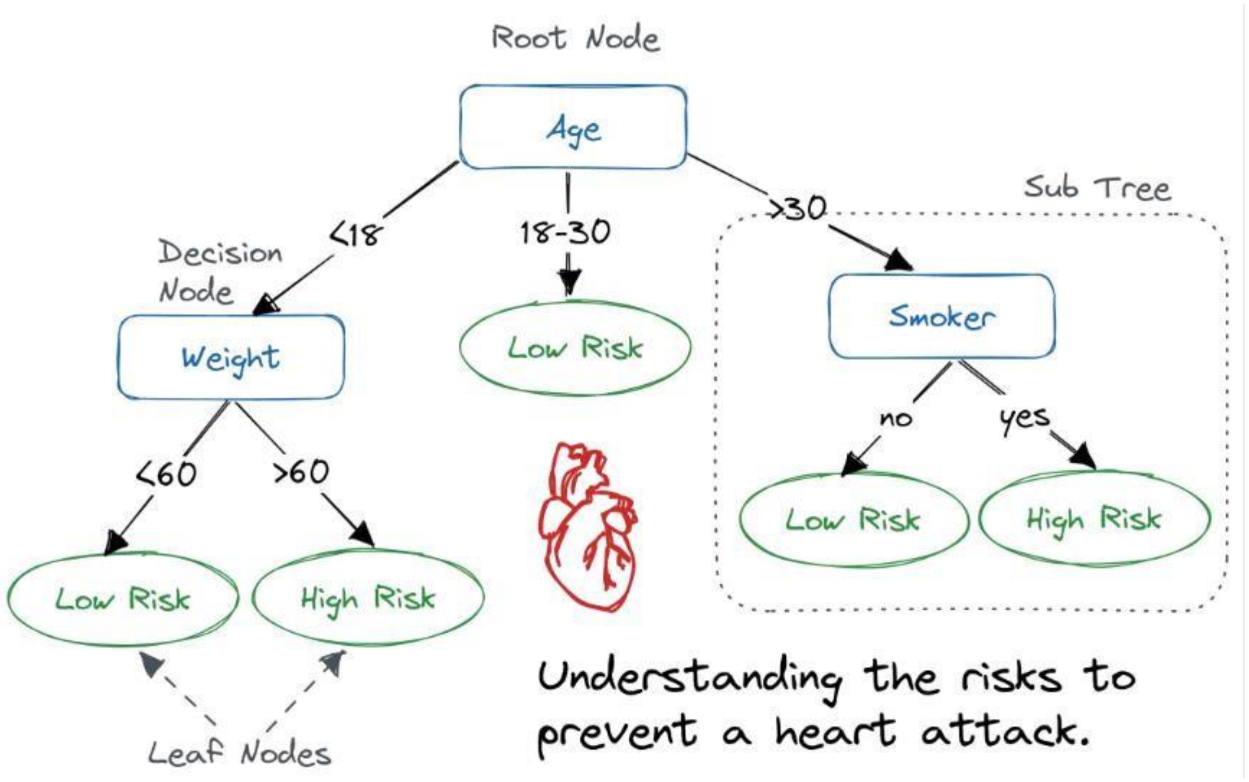
Decision Trees

- It is a non-parametric supervised learning algorithm.
- It has a hierarchical tree structure, including a root node, branches, internal nodes, and leaf nodes.
- It serves as the foundation for classical machine learning algorithms like Random Forests, Bagging, and Boosted Decision Trees.
- The concept involves representing data as a tree, with internal nodes signifying tests on attributes (conditions).
- Each branch represents an outcome of the respective test.
- The leaf nodes, or terminal nodes, hold class labels.
- Decision trees can be used for classification as well as regression problems.
- Decision trees start with a root node and end with a decision made by leaves.
- Decisions are made based on features of the given dataset.
- They are build Using CART Algorithm.
- Decision trees work by asking questions and splitting the tree into subtrees based on the answers.
- Decision trees are graphical representations of possible solutions to a problem based on given conditions.
- A decision tree can contain categorical data (YES/NO) as well as numeric data.

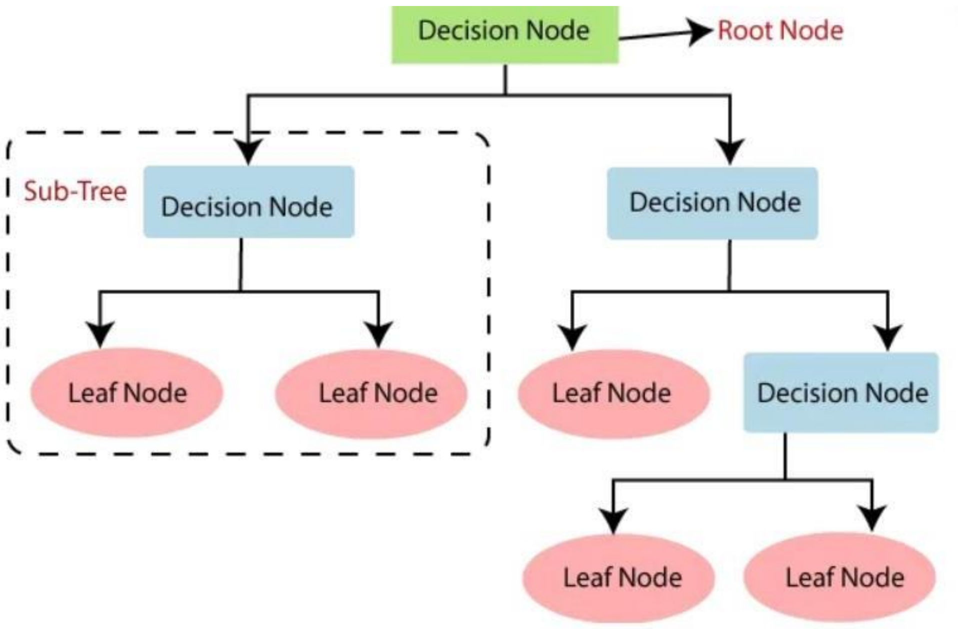
Example:



Terminologies:

- **Root Nodes:** The node at the beginning of a decision tree, where the population initiates division based on various features.
- **Decision Nodes:** Nodes resulting from the splitting of root nodes.
- **Leaf Nodes:** Nodes where further splitting is not possible; also known as leaf nodes or terminal nodes.
- **Branch/Sub-tree:** A sub-section of the decision tree.
- **Pruning**: Involves cutting down nodes to prevent overfitting.
- **Splitting:** A process to divide a node into different subnodes.
- **Parent and child node**: A node divided into sub-nodes. The sub-nodes from the parent node are known as the child node.

Example:



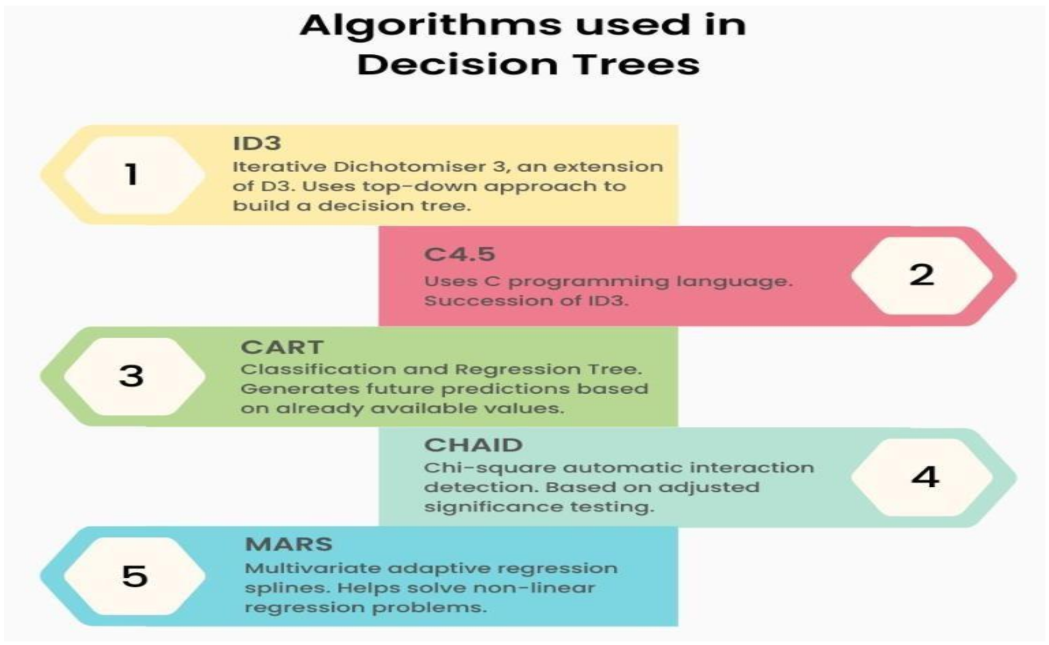
Categories of Decision Trees:

- CART (Classification and Regression Trees):

  Utilizes the Gini Index as a metric for classification.

- ID3 (Iterative Dichotomiser 3):

  Employs the Entropy function and Information gain as metrics.



Algorithms used in Decision Trees

1. **ID3**
   Iterative Dichotomiser 3, an extension of D3. Uses top-down approach to build a decision tree.
2. **C4.5**
   Uses C programming language. Succession of ID3.
3. **CART**
   Classification and Regression Tree. Generates future predictions based on already available values.
4. **CHAID**
   Chi-square automatic interaction detection. Based on adjusted significance testing.
5. **MARS**
   Multivariate adaptive regression splines. Helps solve non-linear regression problems.

ID3 (Iterative Dichotomiser 3)

ID3 (Iterative Dichotomiser 3) is a classic algorithm used for constructing decision trees, primarily for classification tasks. It was developed by Ross Quinlan in the 1980s and is based on the concept of entropy from information theory. ID3 is a precursor to more sophisticated algorithms like C4.5 and CART.

Explanation:

ID3 works by recursively partitioning the dataset into subsets based on the values of different attributes. It selects the attribute that provides the most information gain at each step to split the data. The goal is to create a decision tree that can classify instances accurately by minimizing uncertainty or entropy.

Steps:

1. Entropy Calculation:

- Calculate the entropy of the dataset before any splits. Entropy measures the uncertainty or randomness in the data.

$$Entropy = -\sum_{i=1}^{n} p_i \log_2 p_i$$

2. Attribute Selection:

- For each attribute, calculate the information gain when using that attribute to split the data.
- Information gain is the difference in entropy before and after the

split. It measures how much a particular attribute reduces uncertainty in the data.

- The attribute with the highest information gain is selected as the splitting attribute for the current node of the tree.

3. Splitting the Dataset:

- Partition the dataset into subsets based on the values of the selected attribute.
- Create a child node for each subset and repeat the process recursively for each child node until one of the stopping criteria is met.

4. Stopping Criteria:

- Stop splitting when one of the following conditions is met:
- All instances in a subset belong to the same class.
- No more attributes are left for further splitting.
- A predefined depth limit is reached.
- A minimum number of instances in a node is reached.

5. Tree Construction:

- Construct the decision tree by repeating the attribute selection and splitting steps recursively until the stopping criteria are met.

6. Pruning (Optional):

- After building the tree, pruning techniques may be applied to prevent overfitting by removing branches that do not significantly improve the model's performance on a validation dataset.

7. Prediction:

- To classify a new instance, traverse the decision tree from the root node down to a leaf node, following the appropriate branches based on the attribute values of the instance.

ID3 is simple and efficient, but it has limitations, such as its tendency to create overly complex trees and its inability to handle continuous attributes without discretization. These limitations were addressed in subsequent algorithms like C4.5 and its variants.

CART (Classification and Regression Trees)

CART (Classification and Regression Trees) is a versatile and widely used algorithm in machine learning, capable of both classification and regression tasks. It works by recursively partitioning the input space into regions based on the feature values. Here's an explanation of CART along with its steps:

Explanation:

CART builds a binary tree structure where each non-leaf node represents a decision based on the value of one feature, and each leaf node represents the output (either a class label in classification or a numerical value in regression).

- ***Classification***: In classification tasks, CART assigns a class label to each leaf node. It uses measures like Gini impurity or entropy to determine the best split at each node, aiming to minimize impurity in the resulting partitions.

- ***Regression:*** In regression tasks, CART predicts a numerical value for each leaf node. It uses measures like mean squared error (MSE) to determine the best split at each node, aiming to minimize the variance of the target variable within the resulting partitions.

Steps:

1. Selecting the Feature and Split Point:

- CART examines each feature and evaluates possible split points to determine the one that maximizes purity (for classification) or minimizes variance (for regression) in the resulting subsets.
- It evaluates various splitting criteria (e.g., Gini impurity, entropy, MSE) to find the optimal split.

2. Partitioning the Data:

- Once the best split point is determined, CART divides the dataset into two subsets based on the chosen feature and split point.
- It creates child nodes for each subset and repeats the process recursively for each subset.

3. Stopping Criteria:

- The recursive partitioning process continues until one of the stopping criteria is met, such as reaching a maximum tree depth, minimum number of samples in a node, or no further improvement in purity or variance reduction.

4. Building the Tree:

- CART continues splitting the data until it either reaches the stopping criteria or it's unable to make further splits that improve the impurity or variance.
- The final result is a binary tree where leaf nodes contain the predicted class label (in classification) or numerical value (in regression).

5. Pruning (Optional):

- After building the full tree, pruning techniques may be applied to prevent overfitting by removing nodes that do not significantly improve the model's performance on a validation dataset.

6. Prediction:

- To make predictions for unseen data, CART traverses the tree starting from the root node and follows the appropriate branches based on the feature values until it reaches a leaf node, which provides the predicted output.

CART is popular due to its simplicity, interpretability, and effectiveness in a wide range of applications. It's also a fundamental component of more advanced ensemble methods like Random Forest and Gradient Boosting Machines.

How Does the Decision Tree Algorithm Work?

-

- Attribute Selection:

  Choose the best attribute using Attribute Selection Measures (ASM) to split the records.

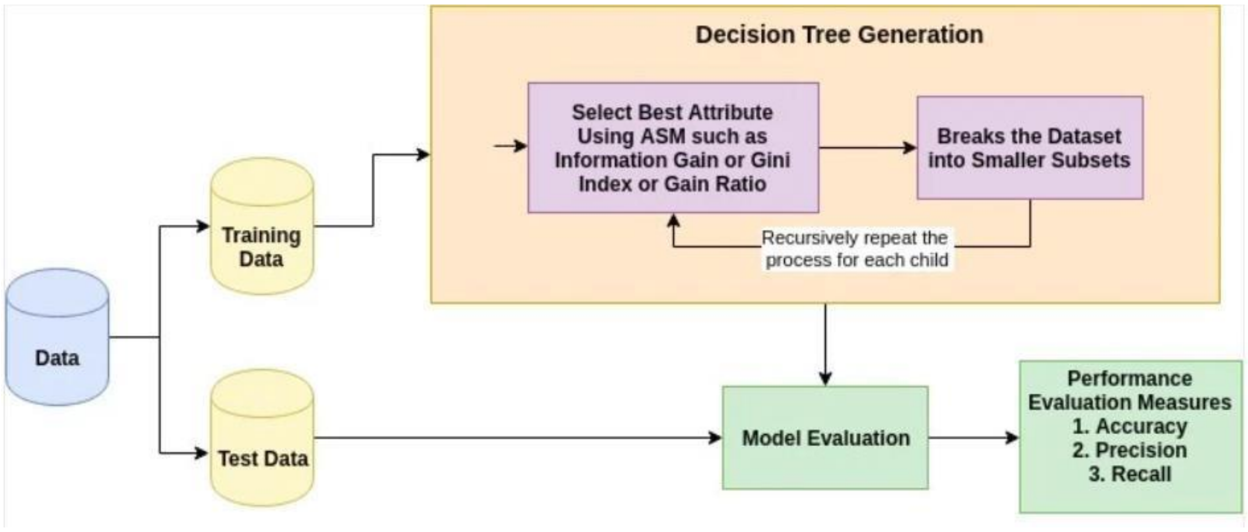
- Decision Node Creation:

  Make the selected attribute a decision node, breaking the dataset into smaller subsets.

- Recursive Tree Building:

  Begin building the tree by repeating the process recursively for each child until one of the conditions is met:

  - All tuples belong to the same attribute value.
  - There are no more remaining attributes.
  - There are no more instances.


Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Attribute Selection Measures:

In the process of implementing a Decision tree, a critical challenge emerges:

Determining the optimal attribute for both the root node and sub-nodes. To address this, an approach known as Attribute Selection Measure (ASM) is employed. This technique enables the selection of the most suitable attribute for the various nodes within the tree.
Two widely used methods for ASM include:

- Information Gain
- Gini Index

Information Gain:

- Information gain quantifies the alteration in entropy following the division of a dataset by a specific attribute.
- In the decision tree construction process, nodes are split based on the value of information gain.
- The objective of a decision tree algorithm is to maximize information gain, prioritizing the split of nodes or attributes with the highest information gain.
- Information gain measures an attribute's effectiveness in dividing training instances based on target types.
- Building a decision tree involves finding attributes with the highest information gain and lowest entropy.
- Information gain represents a reduction in entropy.
- It calculates the difference between entropy before and after splitting the dataset based on specific attribute values.
- A higher information gain indicates a more effective attribute for splitting.

formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

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

The information gain of the split is the difference between the entropy of the original data set and the weighted average of the entropy of the subsets. The weighted average is calculated by multiplying the entropy of each subset by the proportion of data points in that subset. The information gain of the split based on color is:

$$InformationGain = Entropy - \frac{3}{10}Entropy_{yellow} - \frac{7}{10}Entropy_{notyellow}$$
$$InformationGain = 0.881 - \frac{3}{10}\times 0 - \frac{7}{10}\times 0.444$$
$$InformationGain = 0.881 - 0 - 0.311$$
$$InformationGain = 0.570$$

This means that the split based on color reduces the uncertainty in the data by 0.570. This is a high information gain, indicating that the split is good.

Entropy:

- Entropy serves as a metric for gauging impurity within a given attribute, indicating the level of randomness in the data.
- Entropy is a measure of the randomness of information.
- Higher entropy means more randomness and harder to solve.
- For example, flipping a coin has high entropy because the outcome is random.
- In ID3, a branch with zero entropy is a leaf node, meaning there is no more information to split on.
- A branch with entropy more than zero will need splitting because there is still randomness to address.

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Formula:

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

where Pi is the probability of an event i from the S state.

where n is the number of classes, and pi is the probability of a data point belonging to class i. Entropy is always between 0 and 1. Entropy is 0 when all the data points belong to the same class, meaning there is no uncertainty. Entropy is 1 when the data points are evenly distributed among all the classes, meaning there is maximum uncertainty.

For example, suppose we have a data set of 10 fruits, where 4 are apples, 3 are bananas, and 3 are oranges. The entropy of this data set is:

$$Entropy = -\frac{4}{10}\log_2\frac{4}{10} - \frac{3}{10}\log_2\frac{3}{10} - \frac{3}{10}\log_2\frac{3}{10}$$
$$Entropy = 0.156 - 0.158 - 0.158$$
$$Entropy = 0.881$$

This means that the data set is quite uncertain, as the fruits are not clearly separated by their classes.

Now, suppose we split the data set based on the color feature. We would have two subsets: one where color is yellow, and one where color is not yellow. The entropy of each subset is:

$$Entropy_{yellow} = -\frac{3}{3}\log_2\frac{3}{3} - \frac{0}{3}\log_2\frac{0}{3} - \frac{0}{3}\log_2\frac{0}{3}$$
$$Entropy_{yellow} = 0 - 0 - 0$$
$$Entropy_{yellow} = 0$$
$$Entropy_{notyellow} = -\frac{4}{7}\log_2\frac{4}{7} - \frac{0}{7}\log_2\frac{0}{7} - \frac{3}{7}\log_2\frac{3}{7}$$
$$Entropy_{notyellow} = 0.222 - 0 - 0.222$$
$$Entropy_{notyellow} = 0.444$$

The entropy of the yellow subset is 0, meaning that there is no uncertainty in this subset. All the fruits in this subset are bananas. The entropy of the not yellow subset is 0.444, meaning that there is some uncertainty in this subset. The fruits in this subset are either apples or oranges.

Gini Index:

- The Gini index serves as a metric of impurity or purity in the context of creating a decision tree, particularly in the CART (Classification and Regression Tree) algorithm.
- Attributes with lower Gini indices are favored over those with higher Gini indices.
- Comparing attributes with lower Gini indices is only possible against attributes with higher Gini indices.
- The Gini index can only create binary splits, and the CART algorithm utilizes it for the same purpose.
- A cost function based on the Gini index can be used to evaluate splits in the dataset.
- The Gini index is calculated by subtracting the sum of squared probabilities for each class from one.
- It favours large partitions and is simple to implement, but information gain may gain fewer partitions with unique values.

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

Gain ratio

- Information gain tends to select attributes with higher values as root nodes, favoringattributes with higher and unique values.
- C4.5, an advancement of ID3, introduces the gain ratio, a modification of information gainthat reduces this bias, making it a more balanced choice.
- The gain ratio addresses the issue of information gain by considering the branch count thatwould result before the split.
- It refines information gain by incorporating the intrinsic information of a split.

$$Gain\ Ratio = \frac{Information\ Gain}{SplitInfo} = \frac{Entropy\ (before) - \sum_{j=1}^{K} Entropy(j, after)}{\sum_{j=1}^{K} w_j \log_2 w_j}$$

Reduction in Variance

- Reduction in variance is an algorithm for regression problems that utilizes the standarddeviation formula to select the optimal split.
- The split with the lowest variance is chosen as the criterion for dividing the population.

Steps to calculate Variance:

- Calculate variance for each node.
- Calculate variance for each split as the weighted average of each node variance.

$$\text{Variance} = \frac{\Sigma(X - \overline{X})^2}{n}$$

Chi-Square

- CHAID stands for Chi-squared Automatic Interaction Detector, a classification method that identifies statistically significant differences between sub-nodes and parent nodes.
- It measures this significance using the sum of squares of standardized differences between observed and expected frequencies of the target variable.
- CHAID works with categorical target variables like "Success" or "Failure" and can performmultiple splits.
- A higher Chi-squared value indicates stronger statistical significance of differences betweensub-nodes and parent nodes.
- CHAID generates a tree structure called CHAID (Chi-square Automatic Interaction Detector)to represent these relationships.

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

Where:

χ^2 = Chi Square obtained
$\sum$ = the sum of
O = observed score
E = expected score

Decision Tree Advantages:

- Simple to understand, mirroring human decision-making processes.
- Useful for solving decision-related problems.
- Encourages consideration of all possible outcomes.
- Requires less data cleaning compared to other algorithms.
- Easy to interpret and understand
- Relatively robust to outliers and missing data
- Can handle both categorical and numerical data
- Computationally efficient to train and predict with
- Can generate feature importance scores

Disadvantages:

- Complexity due to numerous layers.
- Potential overfitting issues, addressable with Random Forest.
- Computational complexity may increase with more class labels.
- Can be overfitted
- Sensitive to the order in which the features are split
- Difficult to interpret when they are very deep or complex.