## **Decision Tree Algorithm**

A decision tree in machine learning is a versatile, interpretable algorithm used for predictive modelling. It structures decisions based on input data, making it suitable for both classification and regression tasks. A [decision tree](https://www.geeksforgeeks.org/decision-tree/) is a type of supervised learning algorithm that is commonly used in machine learning to model and predict outcomes based on input data. It is a tree-like structure where each internal node tests on attribute, each branch corresponds to attribute value and each leaf node represents the final decision or prediction.  They can be used to solve both regression and classification problems.

There are specialized terms associated with decision trees that denote various components and facets of the tree structure and decision-making procedure:

* **Root Node:** A decision tree’s root node, which represents the original choice or feature from which the tree branches, is the highest node.
* **Internal Nodes (Decision Nodes)**: Nodes in the tree whose choices are determined by the values of particular attributes. There are branches on these nodes that go to other nodes.
* **Leaf Nodes (Terminal Nodes)**: The branches’ termini, when choices or forecasts are decided upon. There are no more branches on leaf nodes.
* **Branches (Edges)**: Links between nodes that show how decisions are made in response to particular circumstances.
* **Splitting**: The process of dividing a node into two or more sub-nodes based on a decision criterion. It involves selecting a feature and a threshold to create subsets of data.
* **Parent Node**: A node that is split into child nodes. The original node from which a split originates.
* **Child Node**: Nodes created as a result of a split from a parent node.
* **Decision Criterion**: The rule or condition used to determine how the data should be split at a decision node. It involves comparing feature values against a threshold.
* **Pruning**: The process of removing branches or nodes from a decision tree to improve its generalisation and prevent overfitting.

The process of forming a decision tree involves recursively partitioning the data based on the values of different attributes. The algorithm selects the best attribute to split the data at each internal node, based on certain criteria such as **Information gain** or **Gini impurity.** This splitting process continues until a stopping criterion is met, such as reaching a maximum depth or having a minimum number of instances in a leaf node. Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.

**Entropy**

Entropy measures the impurity or uncertainty in a dataset. A dataset is pure if it contains only one class, meaning entropy is 0. If the dataset is evenly split between classes, entropy is at its maximum (1 for binary classification). For a dataset with ccc classes, entropy is:

A black and white math equation

Description automatically generated

where:

* p\_i is the proportion of instances in class i.
* The sum runs over all classes.

**Information Gain**

Information Gain measures the **reduction in entropy** when splitting a dataset based on an attribute. The higher the Information Gain, the better the attribute is for splitting.

A black and white math equation

Description automatically generated

A white background with black text

Description automatically generated

**Example**: Consider a dataset with **10 instances**, **3 features** (Weather, Temperature, Windy), and **1 target feature** (Play):

A screenshot of a computer

Description automatically generated

A math equations on a white background

Description automatically generated

A math equation with numbers and symbols

Description automatically generated

A screenshot of a math test

Description automatically generated

A screenshot of a calculator

Description automatically generated

A screenshot of a math table

Description automatically generated

A close-up of a white background

Description automatically generated

A white background with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

Conclusion:

* Entropy helps measure uncertainty.
* Information Gain helps decide the best split.
* The final decision tree is built step-by-step by choosing the attribute with the highest Information Gain at each level.

**Gini Index in Decision Trees:**

The **Gini Index (Gini Impurity)** is another metric used in **decision trees** to determine the best feature for splitting the data. It measures the **impurity** of a dataset, similar to entropy, but is computationally simpler. The Gini Index quantifies how often a randomly chosen instance from the dataset would be incorrectly classified **if randomly labeled** according to the class distribution. For a dataset with ccc classes, the Gini Index is:

A number of numbers in a math equation

Description automatically generated with medium confidence

where:

* p\_i​ is the probability of class iii in the dataset.
* c is the total number of classes.

Lower Gini Index → Higher purity  
Higher Gini Index → Higher impurity

Gini Index = 0 → Dataset is pure (all instances belong to one class).

Gini Index = 0.5 → Dataset is equally distributed among classes (most impure in binary classification).

A screenshot of a math problem

Description automatically generated

A screenshot of a graph

Description automatically generated

A math equations on a white background

Description automatically generated

A screenshot of a graph

Description automatically generated

Key Differences

* Entropy uses logarithms (slower computation), while Gini Index uses simple squares.
* Gini Index favours larger class separations over entropy.
* Entropy vs. Gini Index: Entropy is more computationally expensive, but Gini is faster and often leads to similar results.

**Pruning** is a technique used in decision trees to reduce overfitting by removing unnecessary branches. It simplifies the tree, making it more generalizable to unseen data. There are two main types of pruning:

**Pre-Pruning (Early Stopping):** Stops the tree from growing before it becomes too complex.

* Limit the tree depth (max\_depth)
* Set a minimum number of samples per split (min\_samples\_split)
* Set a minimum number of samples per leaf (min\_samples\_leaf)
* Limit the number of splits based on impurity decrease (min\_impurity\_decrease)

A black text with black text

Description automatically generated with medium confidence

* **Pros:** Faster, prevents overfitting early.
* **Cons:** Might stop too soon and miss important patterns.

**Post-Pruning (Cost Complexity Pruning):** Grows the full tree first, then removes weak branches.

* Uses cost complexity pruning (ccp\_alpha in Scikit-Learn).
* A higher ccp\_alpha value means more pruning (simpler tree).

A close-up of a computer code

Description automatically generated

* **Pros:** Finds the optimal tree size.
* **Cons:** Slower since the full tree is built first.

## **Why is Pruning Important?**

* **Prevents overfitting** (reduces variance)
* **Improves generalization** on new data
* **Reduces complexity** (faster and easier to interpret)

## **Random Forest Algorithm**

Random Forest algorithm is a powerful tree learning technique in Machine Learning to make predictions and **then we do voting of all the tress to make prediction**. They are widely used for classification and regression task.

* It is a type of classifier that uses many decision trees to make predictions.
* It takes different random parts of the dataset to train each tree and then it combines the results by averaging them. This approach helps improve the accuracy of predictions. Random Forest is based on [ensemble learning](https://www.geeksforgeeks.org/a-comprehensive-guide-to-ensemble-learning/).

Process starts with a dataset with rows and their corresponding class labels (columns).

* Then - Multiple Decision Trees are created from the training data. Each tree is trained on a random subset of the data (with replacement) and a random subset of features. This process is known as **bagging** or **bootstrap aggregating**.
* Each Decision Tree in the ensemble learns to make predictions independently.
* When presented with a new, unseen instance, each Decision Tree in the ensemble makes a prediction.
* The randomness in data samples and feature selection helps to prevent the model from overfitting making the predictions more accurate and reliable.

The final prediction is made by combining the predictions of all the Decision Trees. This is typically done through a majority vote (for classification) or averaging (for regression).

**Key Features of Random Forest**

* **Handles Missing Data**: Automatically handles missing values during training, eliminating the need for manual imputation.
* Algorithm ranks **features based on their importance in making predictions** offering valuable insights for feature selection and interpretability.
* **Scales Well with Large and Complex Data** without significant performance degradation.
* Algorithm is versatile and can be applied to both classification tasks (e.g., predicting categories) and regression tasks (e.g., predicting continuous values).