Machine Learning 101

Rajdeep Chatterjee, Ph.D.
Amygdala AI, Bhubaneswar, India *

January 2025

Decision Tree Algorithm

1 Introduction to Decision Trees

A decision tree is a model that partitions the data space into regions based on certain decision rules. It is widely used in decision analysis, machine learning, and statistics. Decision trees are interpretable and easy to understand compared to other models.

2 Decision Trees

2.1 What is a Decision Tree?

Answer: A decision tree is a supervised machine learning model used for classification and regression tasks. It uses a tree-like structure where internal nodes represent features, branches represent decision rules, and leaf nodes represent outcomes. Decision trees can be expressed mathematically as:

$$\text{Output} = f(\text{Feature Set}) \tag{1}$$

where f is a series of decision rules applied hierarchically.

2.2 What are the types of Decision Trees?

Answer: Decision trees are broadly classified into:

1. Classification Trees
2. Regression Trees

Each type is explained in detail below.

3 What is a Decision Tree?

A decision tree is a hierarchical structure consisting of:

*Amygdala AI, is an international volunteer-run research group that advocates for *AI for a better tomorrow* `http://amygdalaai.org/`.

- **Root Node:** Represents the entire dataset and splits it into sub-nodes based on certain conditions.
- **Internal Nodes:** Represent tests on features that divide the data further.
- **Leaf Nodes:** Represent the final decision or outcome.

The decision-making process follows the structure of the tree by applying conditions recursively.

4 Brief History of Decision Trees

The concept of decision trees can be traced back to the 1960s when they were first used in statistics and decision analysis. Notable milestones include:

- 1963: The invention of the ID3 algorithm by J. Ross Quinlan.
- 1986: Development of CART (Classification and Regression Trees) by Breiman et al.
- Present: Integration of decision trees into ensemble methods like Random Forests and Gradient Boosting.

5 Types of Decision Trees

5.1 Classification Trees

Classification trees are used to predict categorical outcomes. The decision rules split the data into classes based on feature values. The Gini Index or Entropy is often used to measure the quality of splits:

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} p_i^2 \tag{2}$$

where p_i is the proportion of instances belonging to class i.

Example: Predicting whether a customer will buy a product based on age and income.

5.2 Regression Trees

Regression trees are used to predict continuous outcomes. Splits are determined by minimizing the Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y})^2 \tag{3}$$

where y_i is the actual value and $\hat{y}$ is the predicted value.

Example: Predicting the price of a house based on size and location.

6 Information Gain

6.1 Definition

Information Gain (IG) is a metric used to select the feature that best splits the data in a Decision Tree. It is based on the concept of entropy from information theory. The Information Gain for a feature is calculated as the difference between the entropy of the entire dataset and the weighted sum of the entropy of the subsets created by splitting the data using that feature.

6.2 Merits

- **Simple and Easy to Understand:** Decision Trees are easy to understand and interpret, even for non-experts.
- **Non-Linear Relationships:** They can model non-linear relationships between the input features and the target variable.
- **No Feature Scaling Required:** Decision Trees do not require normalization or scaling of the features.
- **Handles Both Numerical and Categorical Data:** Decision Trees can handle both types of data, making them versatile.
- **Works Well with Missing Data:** Decision Trees can handle missing data by splitting based on available information.

6.3 Demerits

- **Overfitting:** Decision Trees are prone to overfitting, especially with complex trees that model noise in the data.
- **Instability:** Small changes in the data can lead to large changes in the structure of the tree.
- **Biased Towards Features with More Levels:** Decision Trees may prefer features with more levels, which can lead to biased splits.
- **Greedy Nature:** The algorithm follows a greedy approach and does not always result in the globally optimal tree.

7 Step-by-Step Explanation Using a Toy Dataset

7.1 Toy Dataset solved with Entropy and Information Gain

Consider the following dataset for classifying whether a person will play tennis based on weather conditions:

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Step 1: Calculate Overall Entropy

The formula for entropy is:

$$E(S) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{4}$$

For **Play Tennis?**, we calculate the probabilities:

$$P(\text{Yes}) = \frac{9}{14},$$
$$P(\text{No}) = \frac{5}{14}$$

Entropy of the dataset:

$$\begin{aligned} E(S) &= -\left(\frac{9}{14}\log_2\frac{9}{14} + \frac{5}{14}\log_2\frac{5}{14}\right) \\ &= -\left(0.643 \times \log_2 0.643 + 0.357 \times \log_2 0.357\right) \\ &= -\left(0.643 \times -0.643 + 0.357 \times -1.485\right) \\ &= 0.940 \end{aligned}$$

Step 2: Entropy for Each Attribute

Outlook:

- **Sunny:** 5 instances (2 Yes, 3 No)

$$\begin{aligned} E(\text{Sunny}) &= -\left(\frac{2}{5}\log_2\frac{2}{5} + \frac{3}{5}\log_2\frac{3}{5}\right) \\ &= 0.971 \end{aligned}$$

- **Overcast:** 4 instances (4 Yes, 0 No)

$$\begin{aligned} E(\text{Overcast}) &= -\left(\frac{4}{4}\log_2\frac{4}{4} + \frac{0}{4}\log_2\frac{0}{4}\right) \\ &= 0 \end{aligned}$$

- **Rain:** 5 instances (3 Yes, 2 No)

$$\begin{aligned} E(\text{Rain}) &= -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right) \\ &= 0.971 \end{aligned}$$

Information Gain for Outlook:

$$\begin{aligned} IG(\text{Outlook}) &= E(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} E(S_i) \\ &= 0.940 - \left(\frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971\right) \\ &= 0.246 \end{aligned}$$

Repeat for Temperature, Humidity, and Wind:

- **Temperature:**
  - **Hot:** 4 instances (2 Yes, 2 No)

$$\begin{aligned} E(\text{Hot}) &= -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) \\ &= 1.000 \end{aligned}$$

- **Mild:** 6 instances (4 Yes, 2 No)

$$E(\text{Mild}) = -\left(\frac{4}{6}\log_2\frac{4}{6} + \frac{2}{6}\log_2\frac{2}{6}\right) = 0.918$$

- **Cool:** 4 instances (3 Yes, 1 No)

$$E(\text{Cool}) = -\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) = 0.811$$

Information Gain for Temperature:

$$IG(\text{Temperature}) = E(S) - \left(\frac{4}{14}\times 1.000 + \frac{6}{14}\times 0.918 + \frac{4}{14}\times 0.811\right) = 0.029$$

- **Humidity:**
  - **High:** 7 instances (3 Yes, 4 No)

$$E(\text{High}) = -\left(\frac{3}{7}\log_2\frac{3}{7} + \frac{4}{7}\log_2\frac{4}{7}\right) = 0.985$$

  - **Normal:** 7 instances (6 Yes, 1 No)

$$E(\text{Normal}) = -\left(\frac{6}{7}\log_2\frac{6}{7} + \frac{1}{7}\log_2\frac{1}{7}\right) = 0.592$$

Information Gain for Humidity:

$$IG(\text{Humidity}) = E(S) - \left(\frac{7}{14}\times 0.985 + \frac{7}{14}\times 0.592\right) = 0.151$$

- **Wind:**
  - **Weak:** 8 instances (6 Yes, 2 No)

$$E(\text{Weak}) = -\left(\frac{6}{8}\log_2\frac{6}{8} + \frac{2}{8}\log_2\frac{2}{8}\right) = 0.811$$

  - **Strong:** 6 instances (3 Yes, 3 No)

$$E(\text{Strong}) = -\left(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right) = 1.000$$

Information Gain for Wind:

$$IG(\text{Wind}) = E(S) - \left(\frac{8}{14}\times 0.811 + \frac{6}{14}\times 1.000\right) = 0.048$$

Step 3: Choose the Attribute with Maximum Gain

Compare the information gains:

$$IG(\text{Outlook}) = 0.246$$
$$IG(\text{Temperature}) = 0.029$$
$$IG(\text{Humidity}) = 0.151$$
$$IG(\text{Wind}) = 0.048$$

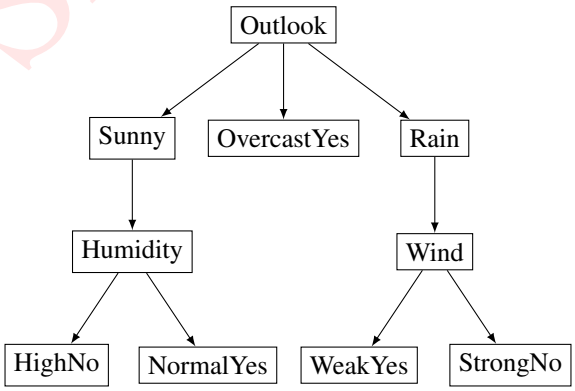
The attribute with the highest information gain is **Outlook**, which becomes the root node of the decision tree.

Step 4: Recursively Split the Dataset

After choosing **Outlook** as the root node, divide the dataset based on its values (Sunny, Overcast, Rain). For each subset:

- **Sunny:**
  - Split on **Humidity**:
    - **High:** All instances are **No**.
    - **Normal:** All instances are **Yes**.
- **Overcast:**
  - All instances are **Yes**.
- **Rain:**
  - Split on **Wind**:
    - **Weak:** All instances are **Yes**.
    - **Strong:** All instances are **No**.

Now, draw the Decision Tree diagram



8 How Gini Index Helps in Splitting Better

The Gini Index is a metric used to evaluate the "impurity" of a dataset. It is commonly used in Decision Trees (like CART) to make splits. The Gini Index for a set of items is calculated as:

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

where p_i is the probability of class i in the dataset.

8.1 How it Helps in Splitting

- **Lower Gini Index:** A split that results in a lower Gini Index is preferred, as it means the resulting subsets are more homogeneous (i.e., contain mostly one class).
- **Efficient for Binary Classification:** Gini Index is often preferred for binary classification because it is computationally simpler than Information Gain.
- **Better for Handling Impurities:** The Gini Index is less sensitive to the number of categories than Information Gain, making it better suited for problems with skewed or unbalanced class distributions.

Toy Dataset

Consider a toy dataset with two features: `Color` and `Class`. We want to calculate the Information Gain and Gini Index for the feature `Color`.

Dataset:

Color	Class
Red	Yes
Blue	No
Red	Yes
Blue	Yes
Red	No

The total number of examples is 5, with 3 instances of `Red` and 2 instances of `Blue`.

Step 1: Calculate the Entropy of the Entire Dataset

The entropy of a dataset is calculated as:

$$H(D) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

where p_i is the proportion of each class in the dataset.

In this case, the dataset has 3 instances of `Yes` and 2 instances of `No`, so the probabilities are:

$$p(\text{Yes}) = \frac{3}{5}, \quad p(\text{No}) = \frac{2}{5}$$

Now, we compute the entropy:

$$H(D) = -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right)$$

$$H(D) \approx -\left(0.6 \times (-0.736) + 0.4 \times (-1.322)\right)$$

$$H(D) \approx 0.971$$

Step 2: Calculate the Entropy for Each Split by `Color`

We now calculate the entropy of the subsets created by splitting on the feature `Color`.

Subset for `Red`

For `Red`, there are 3 instances: 2 `Yes` and 1 `No`. The probabilities are:

$$p(\text{Yes}) = \frac{2}{3}, \quad p(\text{No}) = \frac{1}{3}$$

The entropy for the `Red` subset is:

$$H(\text{Red}) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right)$$

$$H(\text{Red}) \approx 0.918$$

Subset for `Blue`

For `Blue`, there are 2 instances: 1 `Yes` and 1 `No`. The probabilities are:

$$p(\text{Yes}) = \frac{1}{2}, \quad p(\text{No}) = \frac{1}{2}$$

The entropy for the `Blue` subset is:

$$H(\text{Blue}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right)$$

$$H(\text{Blue}) = 1$$

Step 3: Calculate the Information Gain

The Information Gain is calculated as:

$$\text{IG}(D, \text{Color}) = H(D) - \left(\frac{|D_{\text{Red}}|}{|D|}H(\text{Red}) + \frac{|D_{\text{Blue}}|}{|D|}H(\text{Blue})\right)$$

Substitute the values:

$$\text{IG}(D,\text{Color}) = 0.971 - \left(\frac{3}{5} \times 0.918 + \frac{2}{5} \times 1\right)$$

$$\text{IG}(D,\text{Color}) = 0.971 - (0.5508 + 0.4)$$

$$\text{IG}(D,\text{Color}) = 0.971 - 0.9508$$

$$\text{IG}(D,\text{Color}) = 0.0202$$

Step 4: Calculate the Gini Index

The Gini Index for the entire dataset is calculated as:

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

For the entire dataset:

$$Gini(D) = 1 - \left(\left(\frac{3}{5}\right)^2 + \left(\frac{2}{5}\right)^2\right)$$

$$Gini(D) = 1 - (0.36 + 0.16)$$

$$Gini(D) = 1 - 0.52 = 0.48$$

For the `Red` subset:

$$Gini(\text{Red}) = 1 - \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)$$

$$Gini(\text{Red}) = 1 - (0.4444 + 0.1111)$$

$$Gini(\text{Red}) = 1 - 0.5555 = 0.4444$$

For the `Blue` subset:

$$Gini(\text{Blue}) = 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right)$$

$$Gini(\text{Blue}) = 1 - (0.25 + 0.25)$$

$$Gini(\text{Blue}) = 1 - 0.5 = 0.5$$

Step 5: Calculate the Weighted Gini Index for the Split

The weighted Gini Index for the split by `Color` is:

$$Gini(\text{Split}) = \frac{|D_{\text{Red}}|}{|D|} Gini(\text{Red}) + \frac{|D_{\text{Blue}}|}{|D|} Gini(\text{Blue})$$

$$Gini(\text{Split}) = \frac{3}{5} \times 0.4444 + \frac{2}{5} \times 0.5$$

$$Gini(\text{Split}) = 0.2666 + 0.2 = 0.4666$$

9 Purpose of the Weighted Gini Index for Split

The Gini Index is a metric used to measure the impurity or purity of a dataset. When building a Decision Tree, the goal is to split the data into subsets that are as pure as possible, i.e., subsets where most of the data points belong to a single class. The Gini Index for a split is calculated as the weighted average of the Gini indices of the subsets created by the split. This helps evaluate how well the feature divides the data.

9.1 Weighted Gini Index Calculation

The formula for the Weighted Gini Index for a split is:

$$Gini(\text{Split}) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} \cdot Gini(D_i)$$

where:

- $|D|$ is the total number of examples in the dataset.
- $|D_i|$ is the number of examples in subset D_i after the split.
- $Gini(D_i)$ is the Gini Index of subset D_i.
- The sum is taken over all subsets D_i created by the split.

9.2 How it Helps in Splitting?

- **Evaluating the Quality of a Split:** The Weighted Gini Index helps assess how well a feature divides the data into subsets. A lower Gini value indicates a better split, as it means the subsets are more homogeneous (i.e., they contain mostly one class).
- **Balance Between Subsets:** The weighted Gini index considers both the size of the subsets and their impurity. A good split not only creates pure subsets but also ensures that the subsets are large enough to be statistically significant.

- **Optimal Split Selection:** By calculating the weighted Gini index for all possible splits, the algorithm can choose the split that minimizes the overall impurity, leading to better classification performance.
- **Handling Imbalanced Classes:** If the dataset has imbalanced classes, the weighted Gini index still provides an effective way to measure how well the split isolates the majority class from the minority class, contributing to better decision tree learning.

10 Conclusion

By following this step-by-step process, decision trees can be constructed to classify or predict outcomes based on given features. In summary, the Weighted Gini Index is a crucial metric that helps the Decision Tree algorithm choose the best feature to split the data at each node. By minimizing the weighted Gini index, the tree is more likely to generate pure and efficient splits, improving the overall performance of the model.