DECISION TREE

A **decision tree** is a **flowchart-like structure** used for classification (or regression).

Each **internal node** → a question on a feature

Each **branch** → outcome of that question

Each **leaf node** → final prediction (class label)

The tree splits the dataset into smaller subsets by asking questions that **reduce uncertainty** (disorder) about the class labels.

Key Components of a Decision Tree:

1. Root Node

- The **top-most node** of the tree.
- Represents the **entire dataset** before any split.
- Splits into branches based on the feature that gives the **best impurity reduction**.

---

2. Internal Node

- A node that represents a **decision point** (test on a feature).
- Example: Weather = Sunny?
- Has at least two child branches.

---

3. Leaf Node (Terminal Node)

- A node that does **not split further**.
- Represents the **final prediction (class label)**.
- Example: "Yes" (Play Tennis) or "No".

---

4. Splitting

- The process of **dividing a node into sub-nodes** based on a feature test.
- Example: Age > 18? splits dataset into Yes and No.

---

5. Branch (Edge)

- The **outcome of a test** at an internal node.

- Connects a parent node to a child node.
- Example: From Weather = Sunny? you have two branches: Yes and No.

---

6. Impurity

- A measure of how **mixed the classes** are in a node.
- Two common metrics:
  - **Entropy**
  - **Gini Index**
- Low impurity = pure node (all samples of same class).

---

7. Information Gain (IG)

- The **reduction in impurity** after a split.
- The split with **highest IG** is chosen.

---

8. Gini Index

- An alternative impurity measure.
- Lower Gini → purer node.

---

9. Depth of Tree

- The **length of the longest path** from root to a leaf.
- Example: Root → Internal Node → Leaf → Depth = 2.
- Too much depth = risk of **overfitting**.

---

10. Pruning

- The process of **removing branches** that do not add much predictive power.
- Helps to **avoid overfitting**.
- Two types:
  - **Pre-pruning**: stop early (using max depth, min samples, etc.)
  - **Post-pruning**: grow full tree then remove unnecessary branches.

---

11. Overfitting

- When the tree becomes too complex (many branches).
- Memorizes training data instead of generalizing.
- Each leaf may contain very few samples → poor performance on test data.

---

12. Underfitting

- When the tree is too shallow.
- Doesn't capture enough patterns.
- Example: Forcing `max_depth = 1` (just one split).

13. Stopping Criteria

Rules to decide **when to stop splitting**:

- Maximum depth reached (`max_depth`)
- Minimum number of samples in a node (`min_samples_split`)
- Node is pure (all samples same class)
- No further IG possible

How Decision Trees Work:

- **Classification**: The tree splits data into classes based on feature values. For example, deciding if an email is spam or not based on features like word frequency.
- **Regression**: The tree predicts a continuous value, like house price, based on feature splits.
- The algorithm recursively splits the input space into regions based on feature conditions and makes a decision based on the majority class or average value in that region.

Example:

Suppose you're deciding whether to play tennis based on weather conditions:

- **Root Node**: Weather condition.
- **Internal Nodes**: Features like "Outlook" (Sunny, Overcast, Rain), "Humidity" (High, Normal), "Wind" (Strong, Weak).
- **Branches**: Conditions like "Outlook = Sunny" or "Humidity = High."
- **Leaf Nodes**: Decisions like "Play" or "Don't Play."

Steps to Build a Decision Tree:

1. **Select the Best Feature**: Use metrics like Gini impurity, entropy (for classification), or variance reduction (for regression) to choose the feature that best splits the data.
2. **Split the Data**: Divide the dataset into subsets based on the selected feature's values.
3. **Repeat**: Recursively apply the process to each subset until a stopping criterion is met (e.g., maximum depth, minimum samples per node, or no further information gain).
4. **Prune (Optional)**: Reduce the tree size to prevent overfitting by removing branches that add little predictive power.

Advantages:

- Easy to understand and interpret.
- Handles both numerical and categorical data.
- Requires minimal data preprocessing (e.g., no need for normalization).
- Can model non-linear relationships.

Disadvantages:

- Prone to overfitting, especially with deep trees.
- Sensitive to small changes in data, leading to different tree structures.
- May struggle with imbalanced datasets.

1. Entropy

- **Definition**: Entropy is a measure of impurity or uncertainty in a dataset. It quantifies the randomness or disorder of a set of class labels, often used in classification tasks.
- **Formula** (for a binary classification problem)

$$\text{Entropy}(S) = -p_1 \log_2(p_1) - p_2 \log_2(p_2)$$

where p_1 and p_2 are the proportions of the two classes in the dataset S. For multi-class problems:

$$\text{Entropy}(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Range: 0 to 1 (for binary classification). Entropy is 0 when all instances belong to one class (pure), and maximum (e.g., 1 for binary) when classes are evenly distributed (maximum uncertainty).

Role in Decision Trees: Entropy is used to measure the impurity of a node. The goal is to reduce entropy after a split, indicating a more organized (purer) subset of data.

2. Gini Index

- **Definition**: The Gini index (or Gini impurity) measures the probability of incorrectly classifying a randomly chosen element in the dataset if it were labeled according to the class distribution in the subset.
- **Formula** (for a dataset S)

$$\text{Gini}(S) = 1 - \sum_{i=1}^{n} p_i^2$$

where p_i is the proportion of class i, and n is the number of classes.

Range: 0 to 0.5 (for binary classification). A Gini index of 0 indicates a pure node (all instances belong to one class), while 0.5 indicates maximum impurity (equal distribution of classes).

Role in Decision Trees: The Gini index is an alternative to entropy for measuring impurity. It's computationally lighter (no logarithms) and often used in algorithms like CART (Classification and Regression Trees) to decide the best split.

3. Information Gain

- **Definition**: Information gain measures the reduction in entropy (or sometimes Gini impurity) after splitting a dataset based on a feature. It quantifies how much information a feature provides about the class.
- **Formula**

$$\text{Information Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

where:

- S is the dataset before the split.
- A is the feature used for splitting.
- S_v is the subset of S for a specific value v of feature A.
- $|S_v|/|S|$ is the proportion of instances in subset S_v.

Role in Decision Trees: Information gain is used to select the feature that results in the largest reduction in entropy (or Gini impurity) after a split. The feature with the highest information gain is chosen for the split, as it best reduces uncertainty.

🎯 Example Dataset

Suppose we're predicting **Play Tennis (Yes/No)** based on **Weather**.

Sample	Weather	PlayTennis
1	Sunny	No
2	Sunny	No
3	Overcast	Yes
4	Rain	Yes
5	Rain	No
6	Rain	Yes

Class distribution (target = PlayTennis):

- **Yes = 3**
- **No = 3**

So dataset is perfectly balanced (50-50).

1. Entropy Before Split

$$Entropy(S) = -\sum p_i \log_2(p_i)$$

Here:

- $p(Yes) = 3/6 = 0.5$
- $p(No) = 3/6 = 0.5$

$$Entropy(S) = -[0.5\log_2(0.5) + 0.5\log_2(0.5)] = -[0.5(-1) + 0.5(-1)] = 1$$

👉 **Entropy before split = 1 (maximum disorder).**

2. Split on Feature: "Weather = Sunny?"

Let's split dataset into two groups:

- **Sunny branch** → Samples {1,2} = {No, No}
- **Not Sunny branch** → Samples {3,4,5,6} = {Yes, Yes, No, Yes}

(a) Sunny Branch

- Labels: {No, No}
- $p(No) = 1, p(Yes) = 0$

$$Entropy(Sunny) = -[1 \cdot \log_2(1) + 0 \cdot \log_2(0)] = 0$$

👉 Pure branch (all "No").

(b) Not Sunny Branch

- Labels: {Yes, Yes, No, Yes}
- $p(Yes) = 3/4 = 0.75, p(No) = 1/4 = 0.25$

$$Entropy(NotSunny) = -(0.75 \log_2 0.75 + 0.25 \log_2 0.25)$$

$$= -(0.75(-0.415) + 0.25(-2)) = 0.811$$

---

3. Weighted Entropy After Split

$$Entropy_{after} = \frac{2}{6} \cdot Entropy(Sunny) + \frac{4}{6} \cdot Entropy(NotSunny)$$

$$= \frac{2}{6} \cdot 0 + \frac{4}{6} \cdot 0.811 = 0.541$$

4. Information Gain

$$IG = Entropy_{before} - Entropy_{after}$$

$$IG = 1 - 0.541 = 0.459$$

👉 **Information Gain for "Sunny?" = 0.459**

---

5. Gini Impurity

Formula:

$$Gini(S) = 1 - \sum(p_i^2)$$

(a) Gini before split

$$Gini(S) = 1 - (0.5^2 + 0.5^2) = 1 - (0.25 + 0.25) = 0.5$$

(b) Sunny branch

$$Gini(Sunny) = 1 - (1^2 + 0^2) = 0$$

(c) Not Sunny branch

$$Gini(NotSunny) = 1 - (0.75^2 + 0.25^2) = 1 - (0.5625 + 0.0625) = 0.375$$

(d) Weighted Gini after split

$$Gini_{after} = \frac{2}{6} \cdot 0 + \frac{4}{6} \cdot 0.375 = 0.25$$

(e) Gini Gain (optional)

$$\Delta Gini = Gini_{before} - Gini_{after} = 0.5 - 0.25 = 0.25$$

🎯 Goal of a Split

When we split a dataset, we want the **children nodes to be purer** (less mixed) than the parent node.

So:

- **Entropy after split** should be **lower** than entropy before split.
- **Gini after split** should be **lower** than gini before split.

Both Entropy/IG and Gini tell us the split is **good** because it reduces impurity.

Underfitting vs Overfitting

Aspect	Underfitting	Overfitting
Definition	The model is **too simple** to capture the underlying patterns in the data.	The model is **too complex** and learns not only the patterns but also the **noise** in the training data.
Model Complexity	Too simple	Too complex
Training Error	High	Low
Test Error	High	High
Generalization	Poor	Poor
Example	Linear model on curved data	Very deep decision tree