Decision Tree

Hao Le Vu

IUH

2025

Decision Tree Classification: Concept

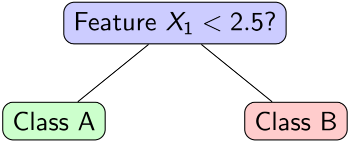
Definition

- A **supervised learning algorithm** used for classification tasks.
- Splits the dataset into branches based on feature values.
- Final decisions are made at **leaf nodes**, each representing a class.

Key Idea

- Partition the feature space recursively to maximize class purity.
- Each split aims to create groups that are as homogeneous as possible.

Decision Tree Visualization



Each internal node represents a decision rule, each leaf represents a predicted class.

Splitting Criterion: Gini Index

Formula

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

Where:

- C = number of classes
- p_i = proportion of class i samples in the node

Intuition

- Measures **impurity** of a node.
- Lower Gini $\rightarrow$ higher purity (samples mostly from one class).
- Splits are chosen to minimize the weighted average Gini of child nodes.
- For binary problems with class proportions p and $1 - p$:

$$\text{Gini} = 2p(1 - p)$$

Dataset: single feature (sorted by feature)

#	Feature X	Class
1	1	A
2	2	A
3	3	B
4	4	A
5	5	B
6	6	B
7	7	B
8	8	A
9	9	B

Totals: $N = 9$, $\#A = 4$, $\#B = 5$.

Step 1 — Root Node: compute Gini

$$p_A = \frac{4}{9}, \qquad p_B = \frac{5}{9}$$

$$\text{Gini}_{root} = 1 - \left(\left(\frac{4}{9}\right)^2 + \left(\frac{5}{9}\right)^2 \right) = 1 - \left(\frac{16}{81} + \frac{25}{81} \right) = 1 - \frac{41}{81} = \frac{40}{81} \approx 0.4938$$

This is the impurity we want to reduce by splitting.

Step 2 — Candidate split: $X < 3.5$

Partition:

- Left (X $\leq$ 3): samples $\{1,2,3\}$ with classes A,A,B $\Rightarrow N_L = 3$.
- Right (X $>$ 3): samples $\{4,5,6,7,8,9\}$ with classes A,B,B,B,A,B $\Rightarrow N_R = 6$.

Left node:

$$p_{A|L} = \frac{2}{3},\; p_{B|L} = \frac{1}{3} \quad \Rightarrow \quad \text{Gini}_L = 1 - \left(\left(\tfrac{2}{3}\right)^2 + \left(\tfrac{1}{3}\right)^2\right) = 1 - \left(\tfrac{4}{9} + \tfrac{1}{9}\right) = \tfrac{4}{9} \approx 0.4444$$

Right node:

$$p_{A|R} = \frac{2}{6} = \tfrac{1}{3},\; p_{B|R} = \tfrac{4}{6} = \tfrac{2}{3} \quad \Rightarrow \quad \text{Gini}_R = 1 - \left(\left(\tfrac{1}{3}\right)^2 + \left(\tfrac{2}{3}\right)^2\right) = \tfrac{4}{9} \approx 0.4444$$

Weighted Gini after split:

$$\text{Gini}_{split(3.5)} = \frac{3}{9} \cdot 0.4444 + \frac{6}{9} \cdot 0.4444 = 0.4444$$

Improvement (Gini decrease):

$$\Delta = \text{Gini}_{root} - \text{Gini}_{split(3.5)} \approx 0.4938 - 0.4444 = 0.0494$$

Step 3 — Candidate split: $X < 6.5$

Partition:

- Left (X $\leq$ 6): $\{1,2,3,4,5,6\}$ classes A,A,B,A,B,B $\Rightarrow N_L = 6$.
- Right (X $>$ 6): $\{7,8,9\}$ classes B,A,B $\Rightarrow N_R = 3$.

Left node:

$$p_{A|L} = \frac{3}{6} = \tfrac{1}{2},\ p_{B|L} = \tfrac{1}{2} \quad \Rightarrow \quad \text{Gini}_L = 1 - \left(\tfrac{1}{4} + \tfrac{1}{4}\right) = 0.5$$

Right node:

$$p_{A|R} = \frac{1}{3},\ p_{B|R} = \frac{2}{3} \quad \Rightarrow \quad \text{Gini}_R = \tfrac{4}{9} \approx 0.4444$$

Weighted Gini:

$$\text{Gini}_{split(6.5)} = \frac{6}{9} \cdot 0.5 + \frac{3}{9} \cdot 0.4444 = 0.3333 + 0.1481 = 0.4815$$

Improvement:

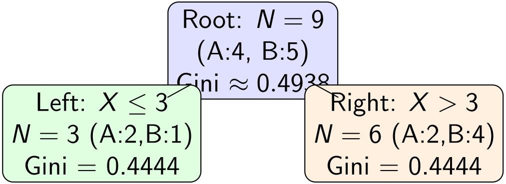
$$\Delta = 0.4938 - 0.4815 = 0.0123$$

(smaller improvement than the split at 3.5)

Step 4 — Compare candidate splits

- $\text{Gini}_{root} \approx 0.4938$
- $\text{Gini}_{split(3.5)} \approx 0.4444 \quad (\Delta \approx 0.0494)$
- $\text{Gini}_{split(6.5)} \approx 0.4815 \quad (\Delta \approx 0.0123)$

Decision: Choose the split at $\mathbf{X} < \mathbf{3.5}$ since it gives the **largest Gini decrease**.



Step 5 — Split the left child at $X < 2.5$

Left child (before): samples $\{1(A), 2(A), 3(B)\}$.
Candidate split at $X < 2.5$:

- Left-left: $\{1,2\}$ classes A,A $\Rightarrow N = 2$, pure, Gini $= 0$.
- Left-right: $\{3\}$ class B $\Rightarrow N = 1$, pure, Gini $= 0$.

Weighted Gini for this local split:

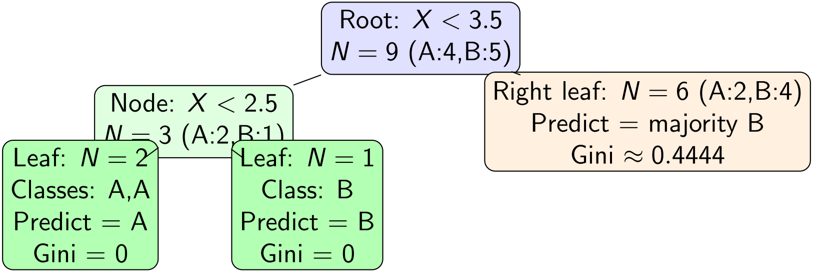
$$\text{Gini}_{left\_split} = \frac{2}{3} \cdot 0 + \frac{1}{3} \cdot 0 = 0$$

Local improvement for that node:

$$\Delta_L = \text{Gini}_L - 0 = 0.4444$$

(a large reduction — we make both child leaves pure)

Final Tree (after chosen splits)



Final predictions: left-most leaves pure; right node still mixed (could be considered for further splitting).

Splitting Criterion: Entropy and Information Gain

Entropy

$$Entropy = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

Where:

- C = number of classes
- p_i = proportion of class i samples in the node

Information Gain

$$IG = Entropy_{parent} - \sum_{children} \frac{N_{child}}{N_{parent}} \times Entropy_{child}$$

Measures how much uncertainty is reduced after a split.

Step 1: Root Node Entropy

For the root node:

$$p_A = \frac{4}{9} = 0.444, \quad p_B = \frac{5}{9} = 0.556$$

$$Entropy_{root} = -\big(0.444 \log_2(0.444) + 0.556 \log_2(0.556)\big)$$

$$Entropy_{root} = -(-0.519 + -0.471) = 0.990$$

Interpretation: Entropy close to 1 $\rightarrow$ high impurity (mixed classes).

Step 2: Candidate Split at $X < 3.5$

Left branch (Samples {1,2,3}): - Class A = 2, Class B = 1

$$p_A = \frac{2}{3} = 0.667, \quad p_B = \frac{1}{3} = 0.333$$

$$Entropy_{left} = -(0.667 \log_2 0.667 + 0.333 \log_2 0.333) = 0.918$$

Right branch (Samples {4,5,6,7,8,9}): - Class A = 2, Class B = 4

$$p_A = \frac{2}{6} = 0.333, \quad p_B = \frac{4}{6} = 0.667$$

$$Entropy_{right} = -(0.333 \log_2 0.333 + 0.667 \log_2 0.667) = 0.918$$

Step 3: Weighted Entropy After Split at 3.5

Weighted average entropy:

$$Entropy_{split(3.5)} = \frac{3}{9} \cdot 0.918 + \frac{6}{9} \cdot 0.918$$

$$Entropy_{split(3.5)} = 0.918$$

Information Gain:

$$IG = Entropy_{root} - Entropy_{split(3.5)} = 0.990 - 0.918 = 0.072$$

Interpretation: Low IG $\rightarrow$ split is not very helpful.

Step 4: Candidate Split at $X < 6.5$

Left branch (Samples {1,2,3,4,5,6}): - Class A = 3, Class B = 3

$$p_A = 0.5, \quad p_B = 0.5$$

$$Entropy_{left} = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1.000$$

Right branch (Samples {7,8,9}): - Class A = 1, Class B = 2

$$p_A = \frac{1}{3} = 0.333, \quad p_B = \frac{2}{3} = 0.667$$

$$Entropy_{right} = -(0.333 \log_2 0.333 + 0.667 \log_2 0.667) = 0.918$$

Step 5: Weighted Entropy and IG for 6.5

Weighted average entropy:

$$Entropy_{split(6.5)} = \frac{6}{9} \cdot 1.000 + \frac{3}{9} \cdot 0.918$$

$$Entropy_{split(6.5)} = 0.972$$

Information Gain:

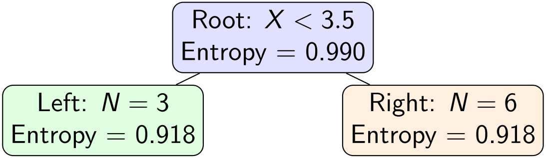
$$IG = Entropy_{root} - Entropy_{split(6.5)} = 0.990 - 0.972 = 0.018$$

Interpretation: Even worse than split at 3.5.

Step 6: Compare Candidate Splits

- Split at $X < 3.5$: $IG = 0.072$
- Split at $X < 6.5$: $IG = 0.018$

Decision: Choose the split at $X < 3.5$ because it has the **highest Information Gain**.

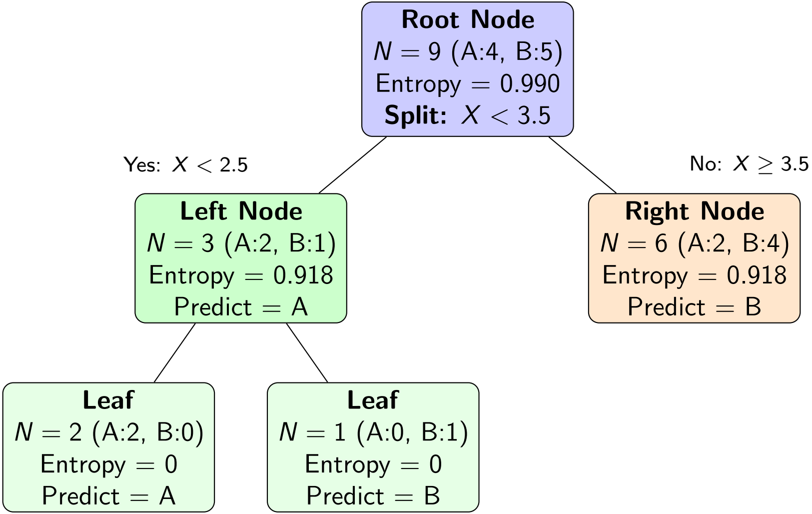


Summary of Information Gain Process

1. Compute entropy for the root node.
2. For each possible threshold:
   - Split dataset into left/right subsets.
   - Compute entropy of each subset.
   - Compute weighted average entropy of the split.
   - Compute Information Gain = root entropy - weighted entropy.
3. Choose the split with the **highest Information Gain**.

Result: Splitting at $X < 3.5$ is the best first step for building the tree.

Final Decision Tree Using Information Gain



Interpretation: The root split at $X < 3.5$ was chosen because it had the **highest Information Gain (0.072)**

Overfitting and Regularization

Overfitting Problem

- Deep trees capture noise $\rightarrow$ poor generalization.
- Perfectly fitting training data is not always desirable.

Solution: Pruning and Constraints

- **Pre-pruning (early stopping):**
  - **max_depth**: Maximum depth of the tree.
  - **min_samples_split**: Minimum samples required to split a node.
  - **min_samples_leaf**: Minimum samples in a leaf node.
- **Post-pruning:** Build full tree, then remove weak branches.

Decision Tree: Advantages and Limitations

Advantages

- Easy to interpret and visualize.
- Handles both numerical and categorical data.
- Requires little data preprocessing.
- Works well with non-linear relationships.

Limitations

- Prone to overfitting if not regularized.
- Unstable: small data changes can cause large structure changes.
- Greedy splitting might miss the globally optimal tree.

Python Example: Decision Tree Classification

```python
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Load dataset
iris = load_iris()
X, y = iris.data, iris.target

# Train decision tree classifier
model = DecisionTreeClassifier(max_depth=3, random_state=42)
model.fit(X, y)

# Visualize the tree
plt.figure(figsize=(12,6))
plot_tree(model, feature_names=iris.feature_names,
          class_names=iris.target_names, filled=True)
plt.show()
```