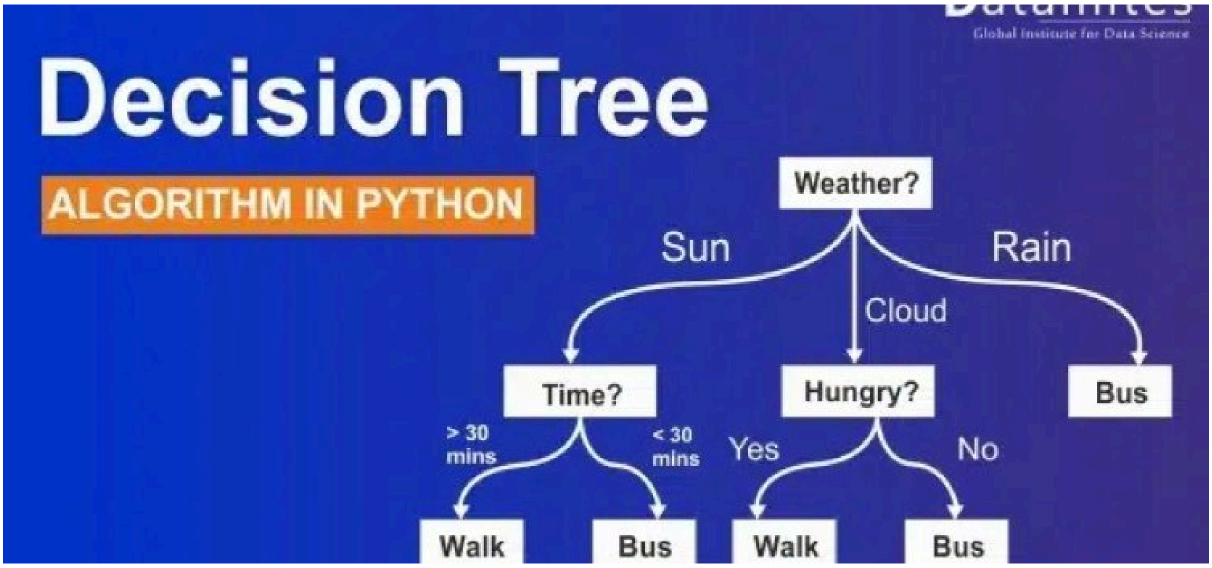


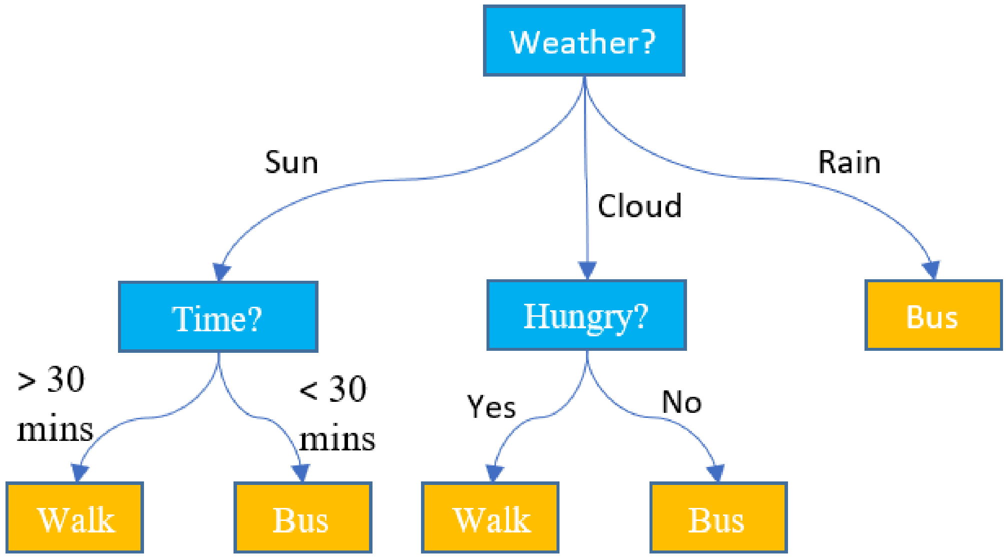
Modules 4: Decision Tree (Cây quyết định)

> Decision Trees are machine learning models with a tree structure, used for classification or regression. Each internal node tests a feature, each branch represents a split condition, and each leaf stores a prediction.

⭐ Highlights: Key formulas, split criterion, and stopping rules are marked below without altering content.

---

1. Overview



What is a Decision Tree?

A Decision Tree is a tree-structured machine learning model used for:

- Classification
- Regression

Structure:

- Node: corresponds to a feature
- Branch: represents a split rule
- Leaf: stores the predicted class or value

Given a dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, \dots, K\}$. For a node t with sample set $S_t \subseteq D$:

- Class c proportion: $p_c(t) = \frac{n_c(t)}{|S_t|}$, where $n_c(t)$ is the number of samples belonging to class c in S_t.
- Convention: $0 \log 0 \equiv 0$.

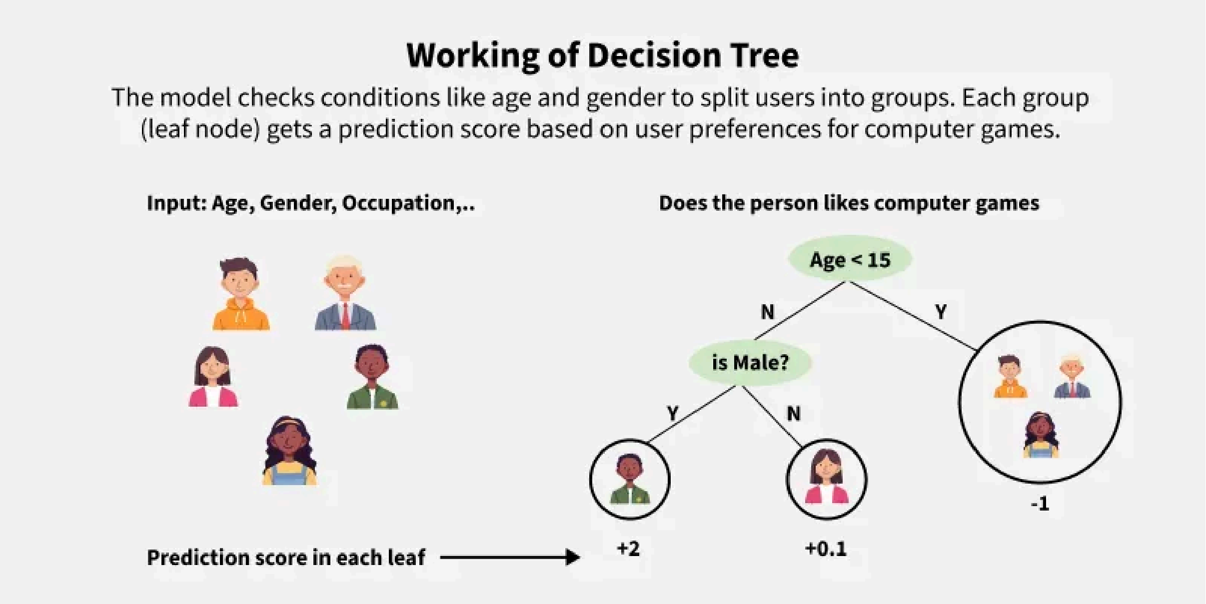
Illustrative example

Predict whether a customer will play tennis based on:

- Outlook
- Temperature
- Humidity
- Windy

```
Outlook?
├── Sunny → Humidity?
│   ├── High → No
│   └── Normal → Yes
├── Overcast → Yes
└── Rain → Windy?
    ├── True → No
    └── False → Yes
```

2. How a Decision Tree is built

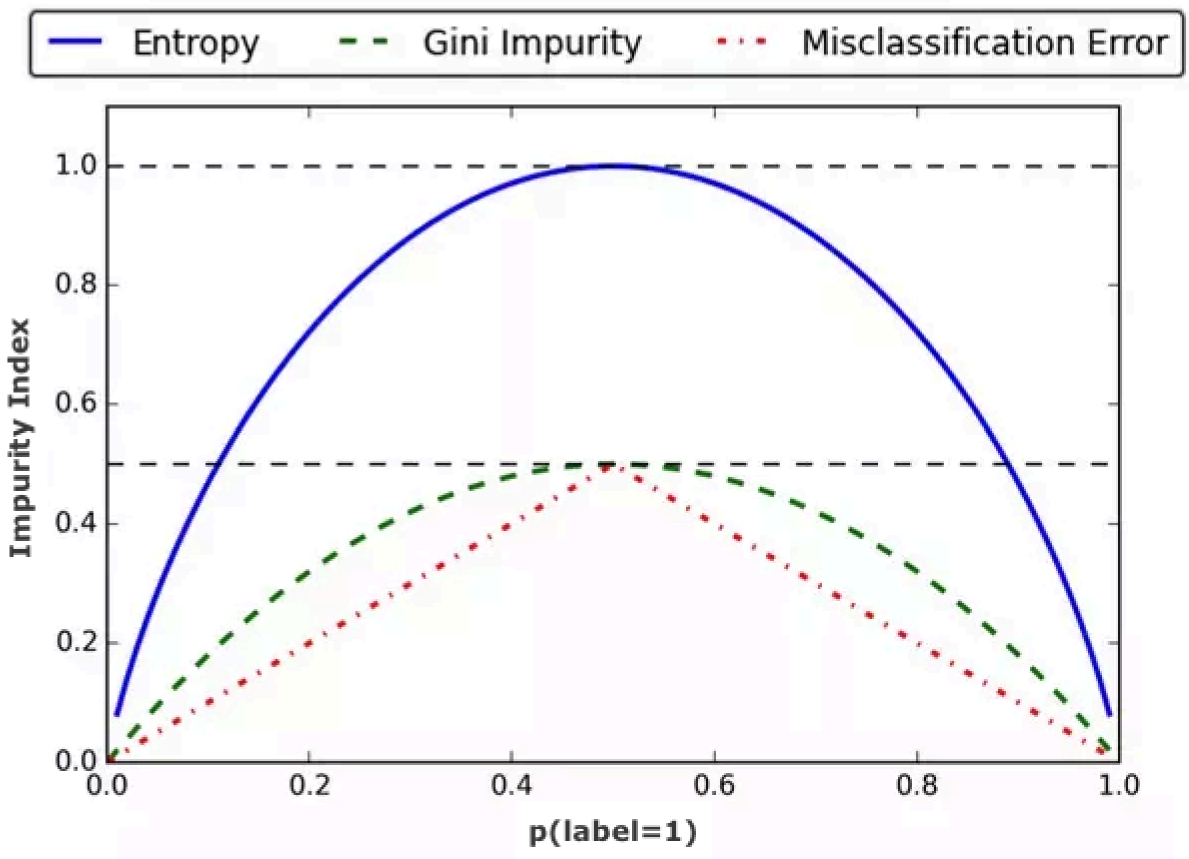


1. Choose the best feature to split the dataset
2. Split the data into child nodes based on that feature
3. Continue splitting until a stopping condition is met:
   - Pure node (only one class)
   - Reaches max_depth

- Too few samples to split (min_samples_split)

---

3. Impurity measures



🧩 3.1. Classification Error

$$E = 1 - \max(p_i)$$

Explanation:

- p_i: probability of class i at the node
- $\max(p_i)$: probability of the majority class

Meaning:

- $E = 0$: pure node
- Higher E → more disorder

Example: Node has 7 "Yes", 3 "No":

- $p_{\text{Yes}} = 0.7,\ p_{\text{No}} = 0.3$
- $E = 1 - 0.7 = 0.3$

🧩 3.2. Entropy

$$H(S) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

Explanation:

- S: dataset at the node
- p_i: proportion of class i
- k: number of classes

Meaning:

- Measures uncertainty at the node
- $H = 0$: pure node
- $H = 1$: most disordered for two balanced classes

Example: Node has 5 "Yes", 5 "No":

- $p_{\text{Yes}} = 0.5,\ p_{\text{No}} = 0.5$
- $H = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$

🧩 3.3. Gini Index

$$G = 1 - \sum_{i=1}^{k} p_i^2$$

Explanation:

- p_i: proportion of class i
- k: number of classes

Meaning:

- Probability of mislabeling if labeled randomly by class distribution
- $G = 0$: nút thuần nhất (pure node)
- Larger G → more disorder

Example: Node has 8 "Yes", 2 "No":

- $G = 1 - (0.8^2 + 0.2^2) = 0.32$

---

4. Choosing a split — Information Gain

$$IG(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

Explanation:

- S: initial sample set
- A: feature used for splitting
- $\text{Values}(A)$: possible values of A
- S_v: subset where $A = v$
- $H(S_v)$: entropy of the subset

Meaning:

- Information Gain is the entropy reduction after the split
- Choose the feature with the largest IG first

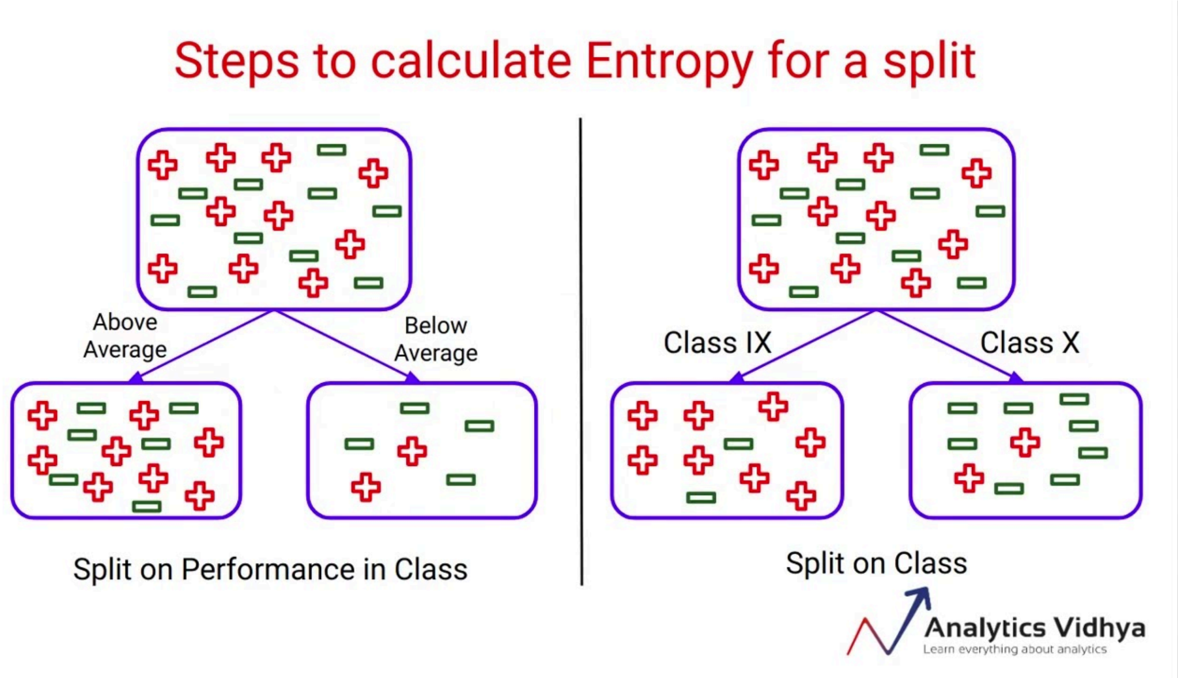
Example calculation

Assume $H(S) = 0.97$. After splitting by "Weather":

Value	Count	Entropy
Sunny	5	0.8
Rain	4	0.0
Overcast	5	0.0

$$IG = 0.97 - \left(\frac{5}{14} \times 0.8 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0 \right) = 0.684$$

→ "Weather" is the best feature!



---

5. Stopping criteria and Pruning

🔶 Stopping criteria

- Pure node (single class)
- Reached `max_depth`
- No remaining features to split

🔶 Pruning

- Pre-pruning: stop early by limiting depth, minimum samples, etc.
- Post-pruning: build the full tree, then remove weak branches based on validation error

🎯 Goal: Reduce overfitting and improve generalization

---

6. Pros and cons of Decision Trees

Pros	Cons
Easy to interpret ("if...then...else")	Prone to overfitting
Handles numeric and categorical data	Sensitive to data changes
No need for feature scaling	Can become deep and complex
Fast inference	Unstable with small perturbations

7. Real-world applications

- Customer churn prediction
- Credit risk classification
- Medical diagnosis classification
- Forecasting sales or house prices (regression trees)

8. Python code example

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

# 1️⃣ Load sample data
data = load_iris()
X, y = data.data, data.target
feature_names = data.feature_names
class_names = data.target_names

# 2️⃣ Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# 3️⃣ Train model
clf = DecisionTreeClassifier(criterion="gini", max_depth=3, random_state=42)
```

```
clf.fit(X_train, y_train)

# 4️⃣ Quick evaluation
acc = clf.score(X_test, y_test)
print(f"Accuracy: {acc:.3f}")

# 5️⃣ Plot the tree
plt.figure(figsize=(12, 8))
plot_tree(
    clf,
    feature_names=feature_names,
    class_names=class_names,
    filled=True,
    rounded=True
)
plt.tight_layout()
plt.show()
```

---

9. Extended example – Predicting student "Pass/Fail"

Sample data

GPA	Study hours/day	Outcome
8.5	4	Pass
6.0	2	Fail
7.5	3	Pass
5.5	1	Fail

Decision tree

```
GPA > 7 ?
├── Yes → Pass
└── No → Study hours > 2.5 ?
    ├── Yes → Pass
    └── No → Fail
```

💡 Helps educators see which factors influence student outcomes.

---

10. Summary

Topic	Key idea
Goal	Split data to make the most accurate predictions
Metrics	Entropy, Gini, Classification Error
Split criterion	Maximize Information Gain
Main issue	Overfitting → pruning helps
Pros	Interpretability, no scaling needed, good for classification
Cons	Overfitting, sensitive to noise
Common improvements	Random Forest, Gradient Boosted Trees