# Fundamentals of Classification and Decision Tree Methods

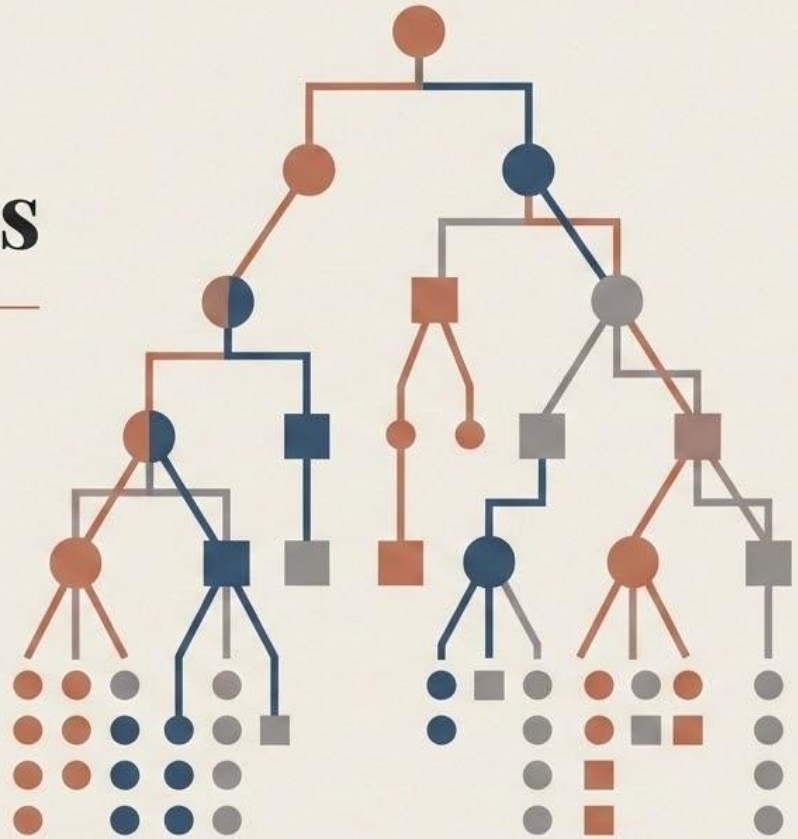
## LECTURE OVERVIEW

**Duration:** 1 Hour

**Target Audience:** Data Science/Computer Science students

**Prerequisites:**

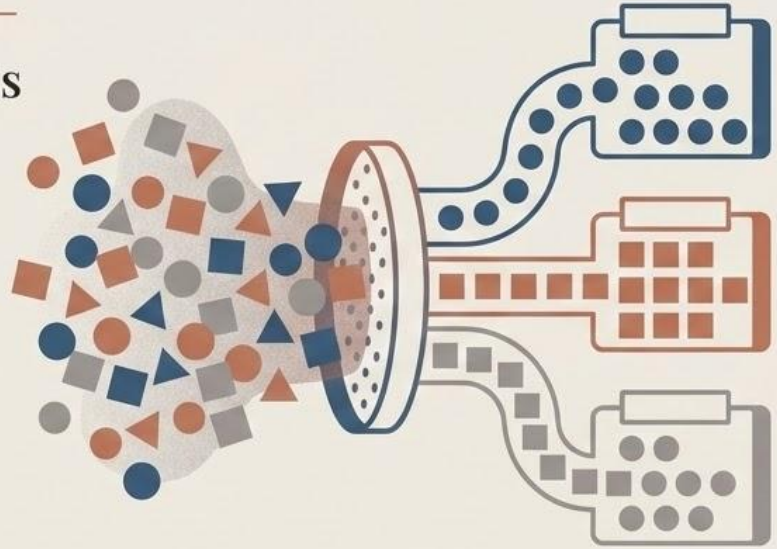
- Basic probability
- Entropy concepts
- Understanding of data mining objectives



# Separating the Signal from the Noise

## Fundamental Techniques for Predictive Modeling

PART 1: INTRODUCTION & LEARNING OBJECTIVES



# The Opening Hook

## Real-World Decisions at Scale

### The Banking Scenario

A bank receives thousands of loan applications daily. How can they automatically and accurately determine which applicants are "safe" and which are "risky"?


### The Healthcare Scenario

A hospital needs to diagnose whether a patient's tumor is benign or malignant based on complex test results.


### The Common Thread

These are classic classification problems.

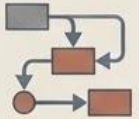
Today, we will learn the fundamental machine learning techniques used to solve them and automate these critical decisions.

# Learning Objectives

## What We Will Cover Today

By the end of this lecture, you will be able to:

**Understand Classification**: Define the core concept and identify high-impact applications of classification models.



**Build the Pipeline**: Explain the general approach and step-by-step process for building classification systems.

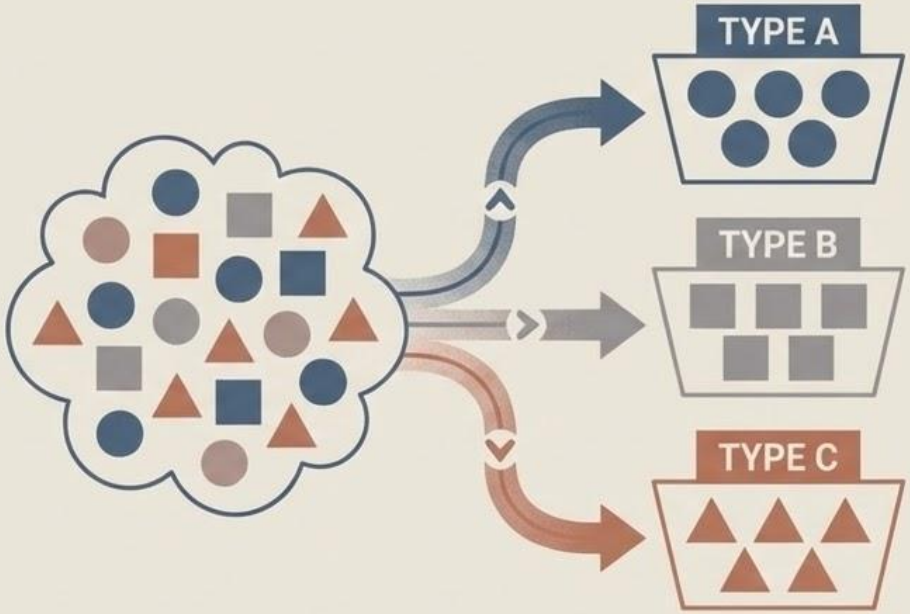
**Master Decision Trees**: Understand decision tree induction and calculate attribute selection measures (like Information Gain) to split data effectively.

**Prevent Overfitting**: Understand tree pruning techniques to ensure your models generalize well to new, unseen data rather than just memorizing the training set.

# Part 2: Basic Concepts of Classification

## Section 6.1.1 - Defining the Problem Space

- **Focus:** What classification is, its variations, and real-world applications.
- **Context:** Part 2



# What is Classification?

## Assigning Data to Categories

**The Definition:**

A data mining technique that assigns items in a dataset to target categories (or classes) to accurately predict outcomes.

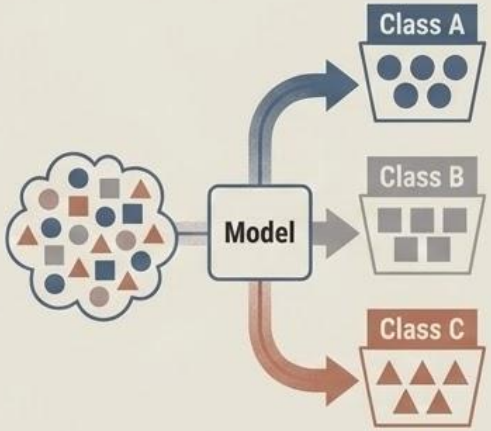
**The Ultimate Goal:**

To accurately predict the target class for new, unseen cases in the real world.

**Key Characteristics:**


**Supervised Learning:** The model learns from labeled training data (where the answers are already known).

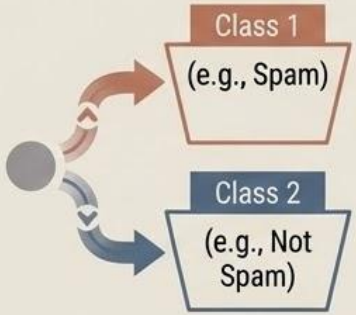
**Discrete Output:** It predicts categorical class labels, not continuous numeric values.



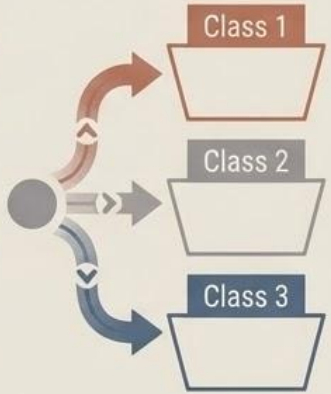
# Types of Classification Problems

## Binary vs. Multiclass

- **Binary Classification:** The target has only two possible values.
  - A tumor is either “cancerous” or “not cancerous.”
  - A loan applicant is “risky” or “safe.”
  - An email is “spam” or “not spam.”




- **Multiclass Classification:** The target has more than two possible values.
  - A tumor is classified as type 1, type 2, or type 3.
  - News stories are categorized as weather, finance, entertainment, or sports.
  - Customer sentiment is tagged as happy, sad, angry, or neutral.

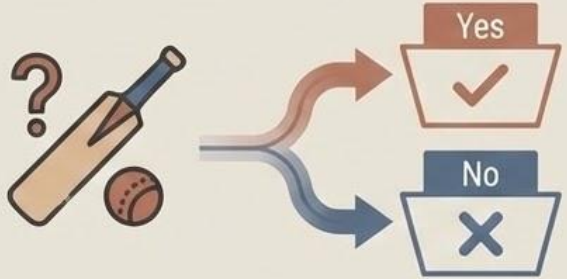


# Classification vs. Regression

## Understanding the Output

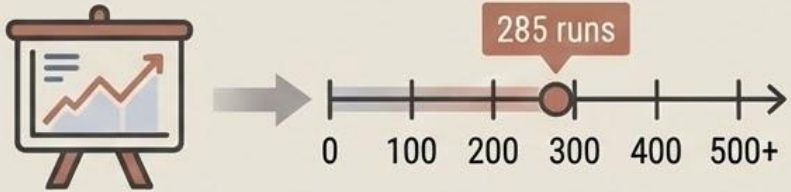
While both are supervised learning techniques, they answer fundamentally different types of questions:

**Classification (Discrete)**
Predicts specific categories or classes.




**Example:** “Will India win the cricket match?” (Yes / No)

**Regression (Continuous)**
Predicts continuous, numeric values.




**Example:** “How many runs will India score?” (285 runs)

# Real-World Applications

## Where Classification is Used Today

Classification models are the engines behind countless modern business and scientific decisions:

**Banking:** Analyzing credit history to identify loan risk.

**Retail:** Predicting whether specific customers will buy certain products.

**Marketing:**
Customer segmentation and targeted ad campaigns.

**Healthcare:** Biomedical analysis, tumor detection, and drug response modeling.

**Business:** Broad credit analysis and automated risk assessment.

**Part 2:** The pipeline for training and deploying machine learning models.

# From Historical Data to Future Predictions

## The 4-Step Classification Lifecycle

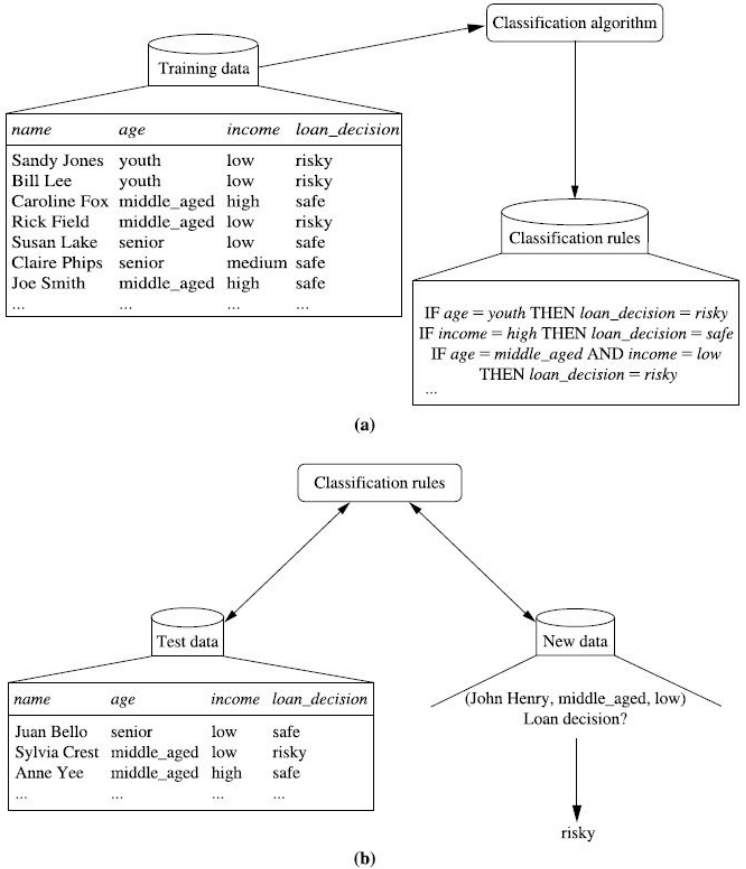



**FIGURE 6.1**

The data classification process: (a) *Learning*: Training data are analyzed by a classification algorithm. Here, the class label attribute is *loan_decision*, and the learned model or classifier is represented in the form of classification rules. (b) *Classification*: Test data are used to estimate the accuracy of the classification rules. If the accuracy is acceptable, the rules can be applied to the classification of new data tuples.

# The Classification Process

## The High-Level Pipeline


## The Strategy:

- Take historical data with known labels.
- Learn the underlying patterns.
- Evaluate the accuracy.
- Finally predict outcomes for new, unknown data.

# Step 1 & Step 2 (Prep & Train)

## Building the Foundation

### Step 1: Data Preparation

Start with a dataset where the class assignments are already known (historical cases).

Target Attribute: The specific class you want to predict (e.g., Credit Rating).

Predictor Attributes: The features used to make the prediction (e.g., Employment History, Home Ownership).

The classification algorithm analyzes the training data to find mathematical relationships between the predictors and predictors and the target.

These relationships are compiled and saved as the predictive 'Model.'

# Step 3 & Step 4 (Test & Deploy)

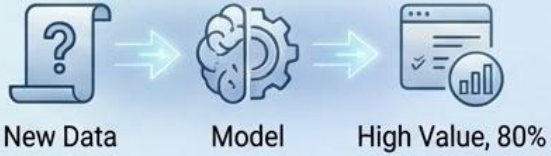
## Validating and Executing

### Step 3: Model Testing


Apply the newly built model to a separate "Test" dataset where the target values are known by us, but hidden from the model.

Compare the model's predicted values against the actual values to evaluate its true accuracy.

### Step 4: Deployment



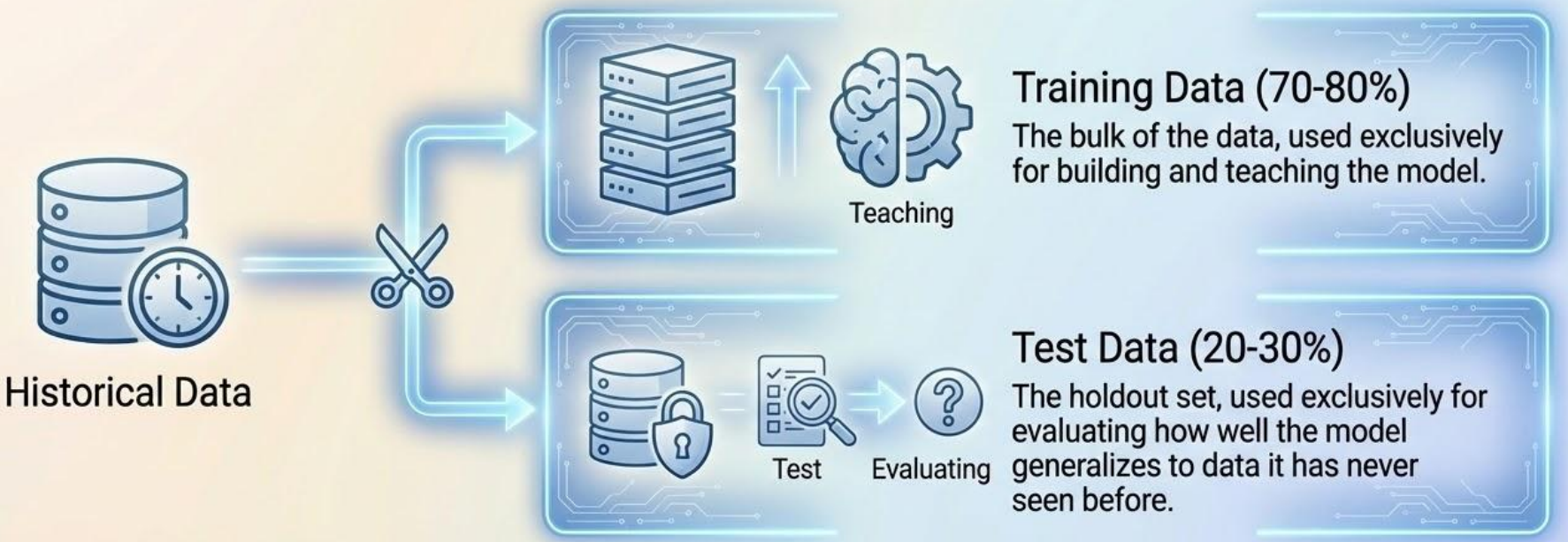

Apply the validated model to brand new data where the outcomes are truly unknown.

The Output: Returns the predicted class assignment and the mathematical probability of that assignment (e.g., "80% probability this customer is High Value").

# The Data Split Strategy

## Protecting Against Memorization

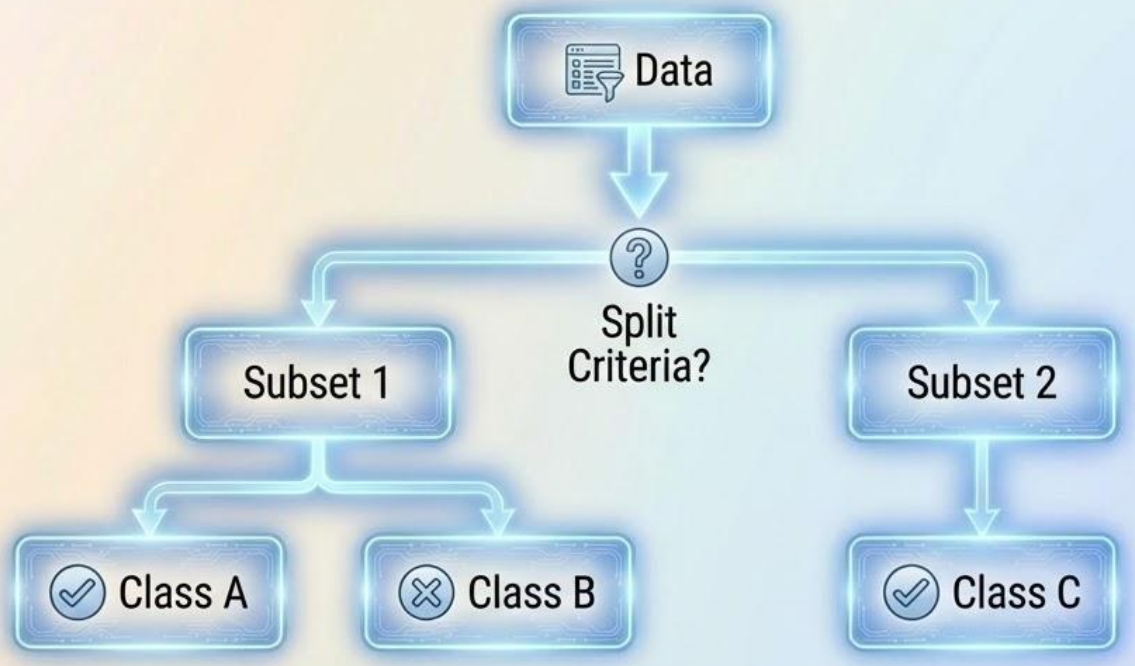
To ensure the model actually learns patterns rather than just memorizing the answers, historical data is strictly divided into two distinct sets before Step 1 begins:



Part 3: Decision Tree Induction

# The Logic of Splitting

## Building Interpretable Classification Models



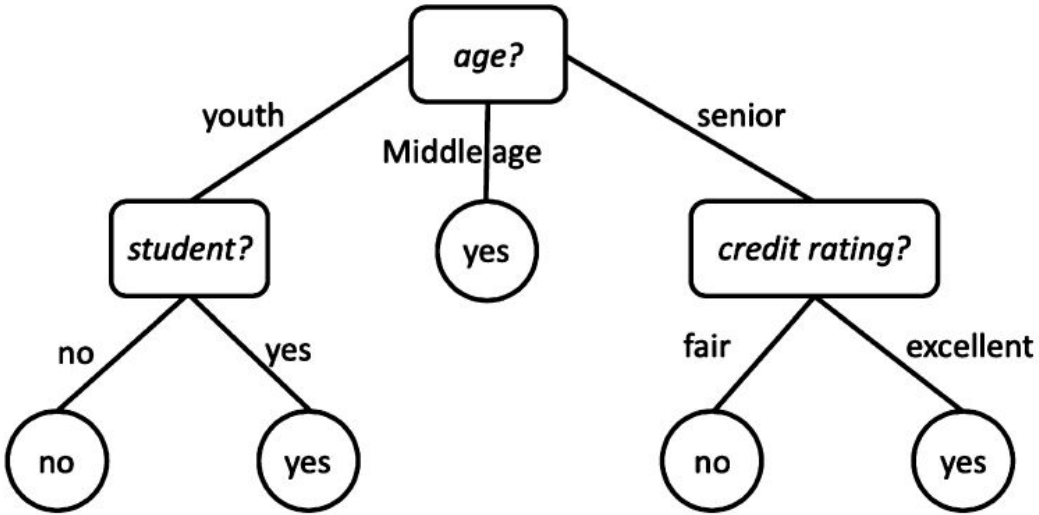


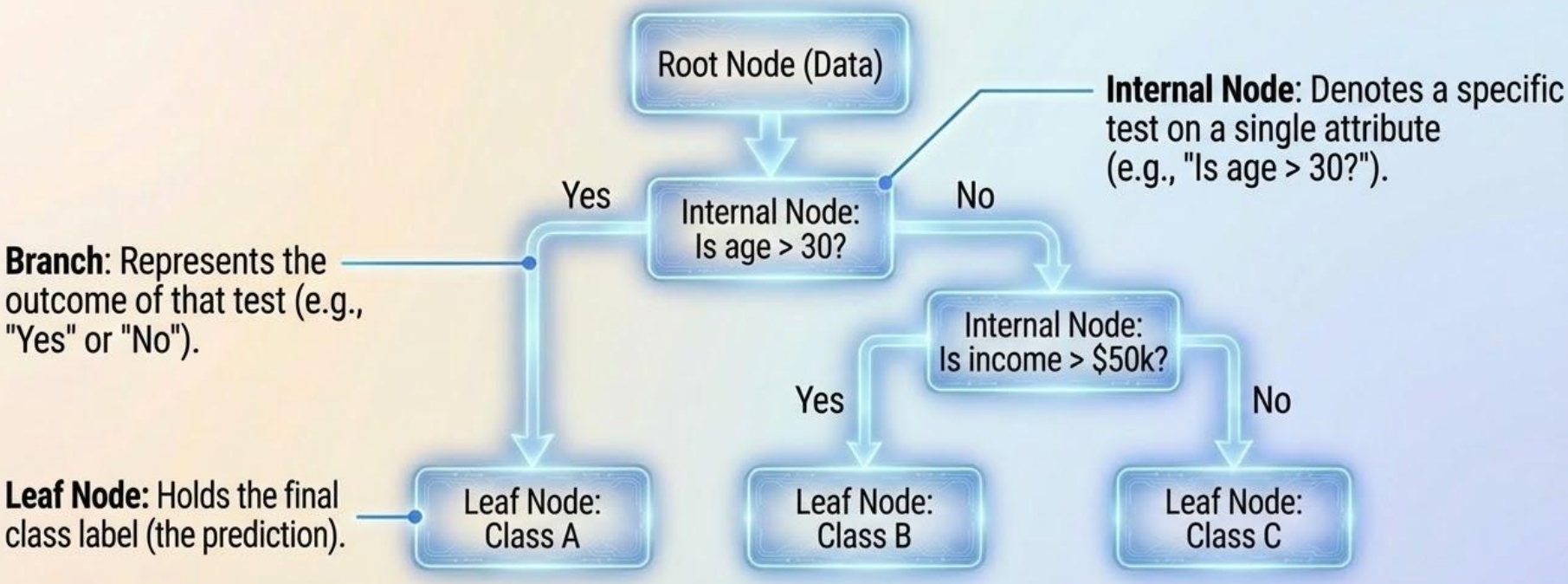

**FIGURE 6.2**

A decision tree for the concept *buys_computer*, indicating whether a customer is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buys_computer* = *no*).

# What is a Decision Tree?

## The Anatomy of the Model

A decision tree is a flowchart-like structure that makes sequential decisions to classify data.



# Example: Loan Approval Decision Tree

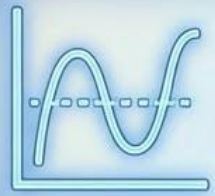
## Following the Logic Path

# Why Use Decision Trees?

## The Advantages of the Method

### Highly Interpretable

They are easy to understand, visualize, and explain to non-technical stakeholders (often reading like a set of human-readable rules).



### Non-Parametric

They make absolutely zero assumptions about the underlying distribution of the data.


### Versatile

They seamlessly handle mixed data types (both numeric and categorical attributes).

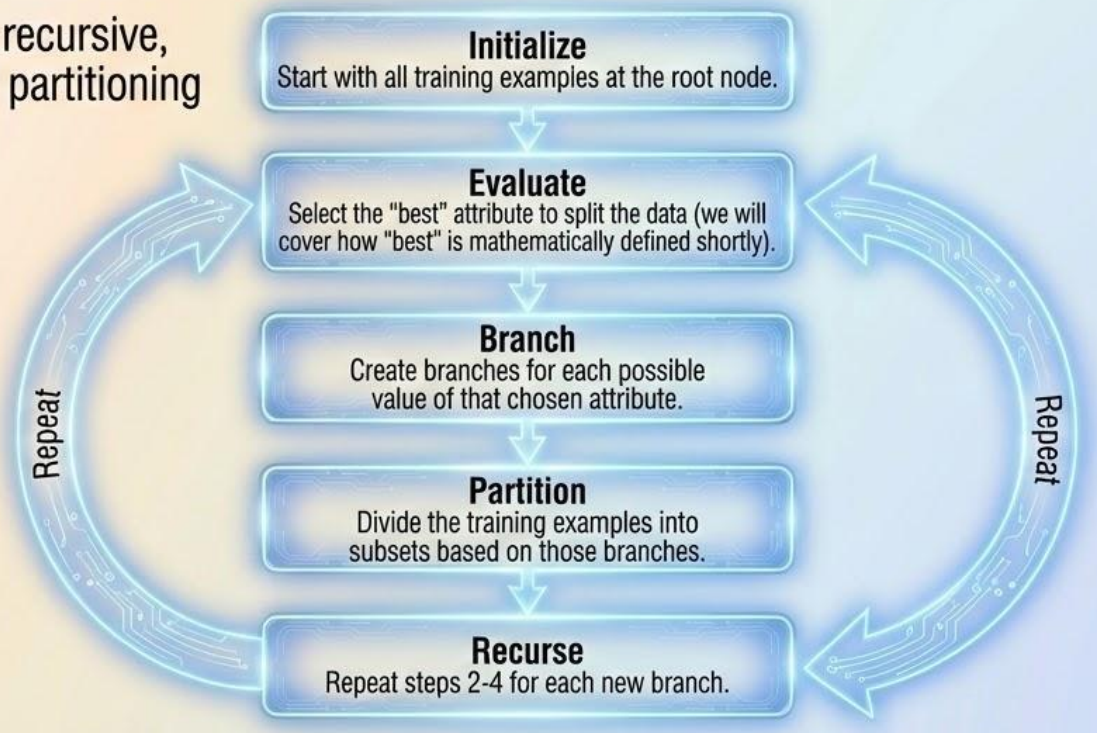

### Inherent Feature Selection

The algorithm naturally ranks data; the most critical attributes for making the prediction automatically appear higher up in the tree.

# Algorithm Overview

## How the Tree is Built (ID3, C4.5, CART)

Building the tree is a recursive, top-down process of partitioning data:



# The Stopping Conditions

## When Does the Algorithm Stop Splitting?

The recursive partitioning halts when one of three conditions is met:


### Pure Node

All examples currently in the node belong to the exact same class (e.g., 100% of the remaining records are “Spam”).


### Attribute Exhaustion

There are no attributes left to split the data any further.


### Data Exhaustion

There are no training examples remaining for that specific branch.

## Part 3: Attribute Selection Measures

# The Math of the Split

### Deciding Which Question to Ask Next

Context: Part 3 (continued)

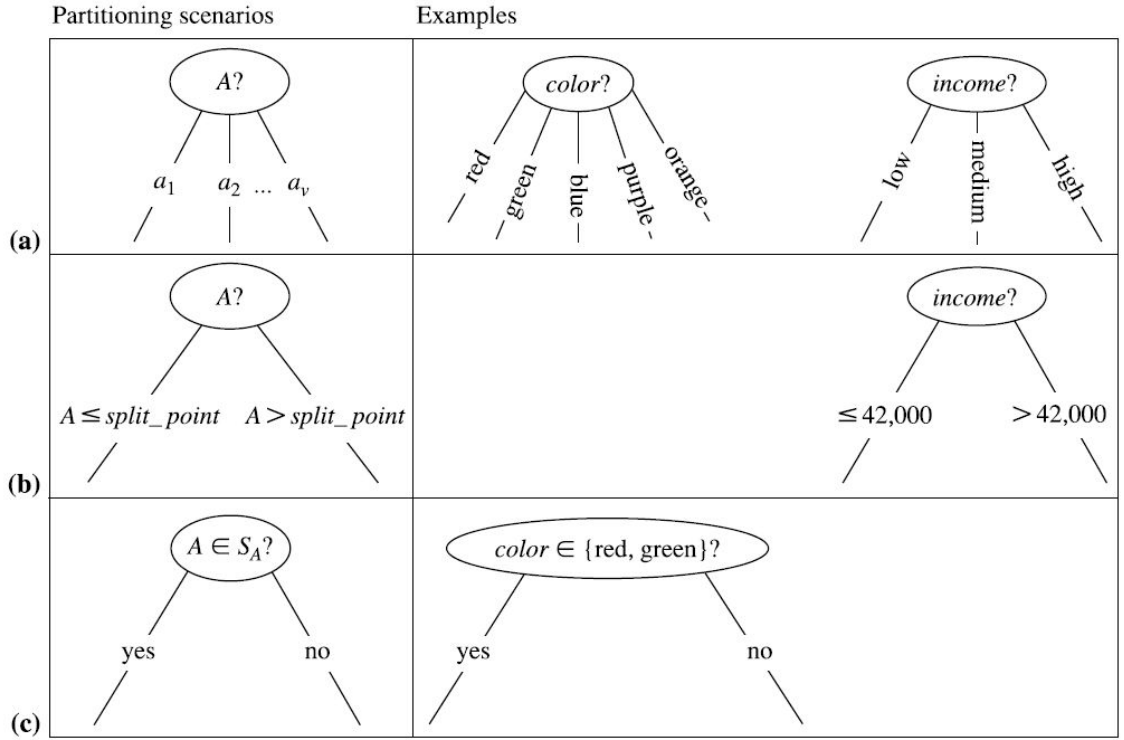



**FIGURE 6.4**

This figure shows three possibilities for partitioning tuples based on the splitting criterion, each with examples. Let $A$ be the splitting attribute. (a) If $A$ is discrete-valued, then one branch is grown for each known value of $A$. (b) If $A$ is continuous-valued, then two branches are grown, corresponding to $A \leq split\_point$ and $A > split\_point$. (c) If $A$ is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A$, where $S_A$ is the splitting subset for $A$.

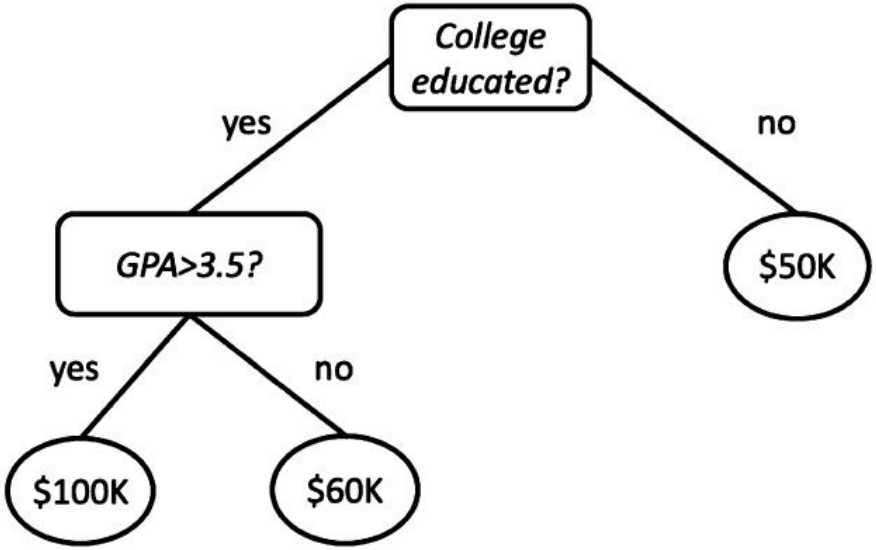



**FIGURE 6.5**

A regression tree for predicting the average yearly income based on an individual's education. The values of the three leaf nodes are calculated as follows. $50K is the average yearly income of all training individuals who do not have a college degree; $60K is the average yearly income of all training individuals who have a college degree with a GPA less than or equal to 3.5; and $100K is the average yearly income of all training individuals who have a college degree with a GPA higher than 3.5. The leaf node values ($50K, $60K, and $100K) are used to predict the yearly income of any test individual who falls into the corresponding leaf nodes.

# The Fundamental Question

## How Do We Choose the “Best” Attribute?

Pure Partitions

### The Goal

At each node, we must choose the attribute that produces the “purest” possible partitions (meaning all examples in a resulting branch belong to the exact same class).


### The Challenge

How do we mathematically measure “purity” or “impurity”?

Attribute Selection Measures

### The Solution

We use Attribute Selection Measures (also known as splitting rules) to rank the attributes and select the one that organizes the data best.

# Entropy and Information Gain

## Measuring Impurity (Used in ID3/C4.5)

### Entropy

Measures the impurity or randomness in a collection of examples ($S$).

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

(Where $c$ is the number of classes and $p_i$ is the proportion of examples in class $i$)

**Maximum Impurity (1.0)**

An equal distribution of classes (e.g., 50% Spam, 50% Not Spam).

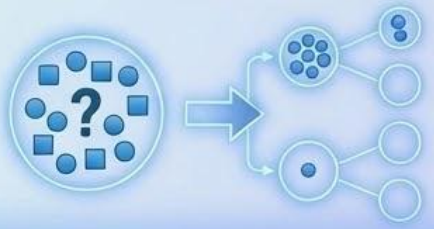
**Minimum Impurity (0.0)**

All examples belong to the exact same class (Pure).


### Information Gain

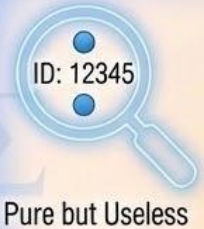
Calculates how much Entropy drops when we split the data using attribute $A$.

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



# Limitation & The Gain Ratio

## Fixing the “Customer ID” Problem




Pure but Useless

### The Limitation of Information Gain

It is heavily biased toward attributes with many distinct values.

Example: “Customer ID” would yield a massive Information Gain because every subset is perfectly pure (1 customer per ID), but it is completely useless for predicting future cases.


### The Solution: Gain Ratio

Penalizes attributes with many values by incorporating “Split Information.”

$$SplitInfo(S,A) = -\sum_{v \in Values(A)} \log_2 \frac{|S_v|}{|S|}$$

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitInfo(S,A)}$$

### Practical Issue

If the denominator becomes zero or very small, the Gain Ratio can become unstable.

# The Gini Index

## The CART Algorithm Approach


### The Concept

Another measure of impurity, favored by the CART (Classification and Regression Trees) algorithm.

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

### Gini Values

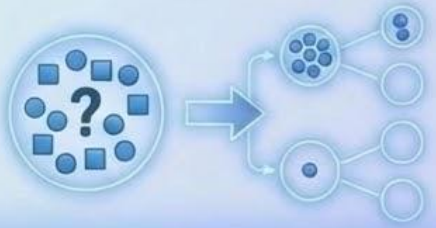

Maximum: Approaches 1 - 1/c (Worst / Most Impure).

Minimum: 0 (Perfectly Pure).


### Gini Gain

Similar to Information Gain, it measures the reduction in impurity after a split.

$$GiniGain(S, A) = Gini(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times Gini(S_v)$$



# Alternative Measures & Experimental Findings

## Other Ways to Split Data

### Alternative Measures

- **R-measure:** Newer measure with promising performance.
- **Relief:** Instance-based feature ranking.
- **J-measure:** Information-theoretic measure.
- **$\chi^2$ measure:** Statistical significance testing.
- **Distance-based measures:** Not biased toward attributes with many values.

### Experimental Findings

**Accuracy**

Predictive accuracy is actually not highly sensitive to the choice of measure. They all perform similarly well.

**Tree Size**

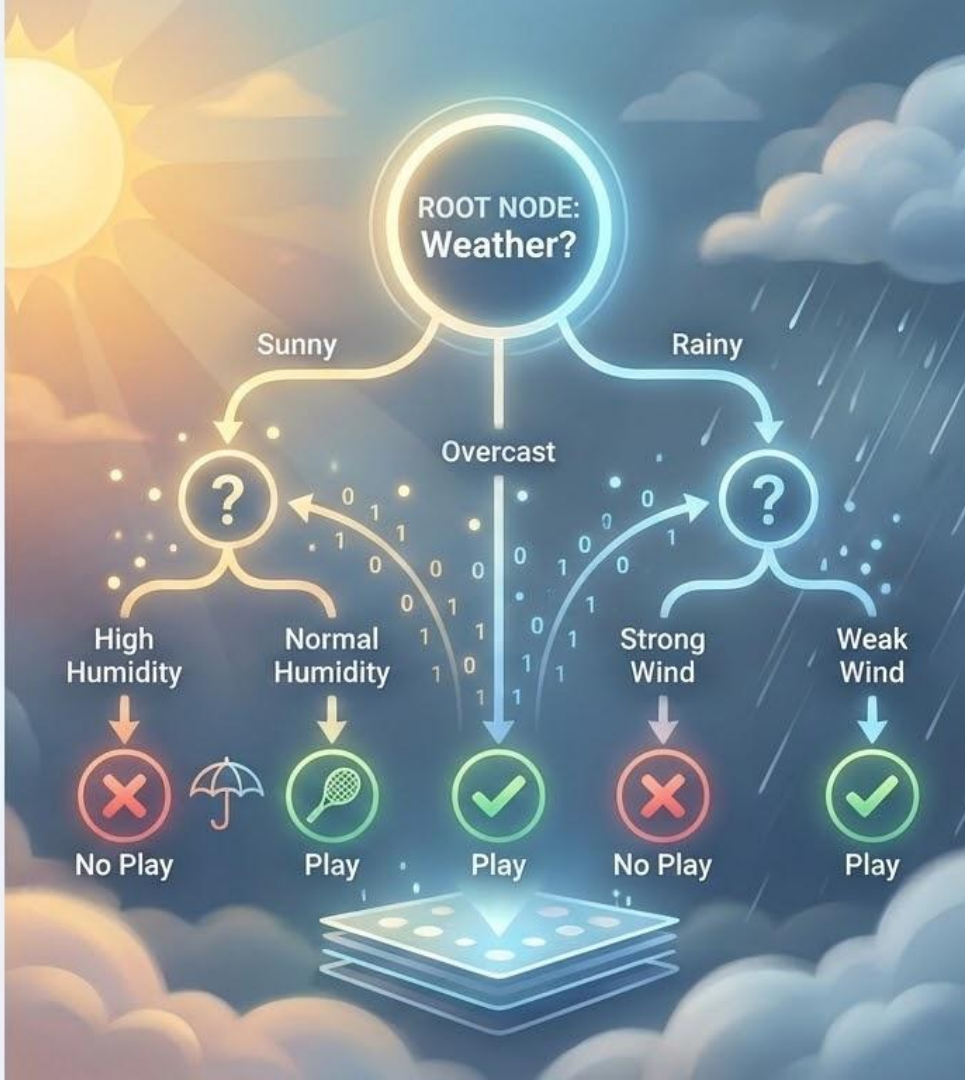
Some measures produce significantly smaller trees (e.g., distance-based measures vs. gain ratio). The Gain Ratio occasionally induces unnecessarily large trees when attributes have varying numbers of values.

WORKED EXAMPLE: ATTRIBUTE SELECTION

# To Play or Not to Play?

## Calculating the Root Node of a Decision Tree

Context: Applying Information Gain to a real dataset.



# The ‘Play Tennis’ Dataset

## The Historical Data

- We have 14 days of historical data.
- Our goal is to predict the Play column (Yes/No) based on four predictor attributes.

**Dataset: "Should we play tennis?"**

| Day | Outlook | Temperature | Humidity | Wind | Play |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

# Step 1 - Calculate Total Entropy

## Measuring the Impurity of the Entire Dataset ($S$)

- Total Examples: 14 days
- Class Distribution: 9 ‘Yes’, 5 ‘No’

To find the baseline impurity, we calculate the Entropy of the entire dataset ($S$):


$$Entropy(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$$

**Entropy(S) = 0.940**

# Step 2 - Evaluate the ‘Outlook’ Attribute

## Splitting by Sunny, Overcast, and Rain

If we split the data using ‘Outlook’, we get three subsets. We must calculate the Entropy for each subset:

☀️ Sunny Subset (5 examples):
2 Yes, 3 No

$$Entropy(S_{Sunny}) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$$

☁️ Overcast Subset (4 examples):
4 Yes, 0 No (Perfectly Pure!)

$$Entropy(S_{Overcast}) = -\frac{4}{4}\log_2\frac{4}{4} - 0 = 0$$

🌧️ Rain Subset (5 examples):
3 Yes, 2 No

$$Entropy(S_{Rain}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$$

# Step 3 - Calculate Information Gain for ‘Outlook’

## How much did Entropy drop?

Now, we calculate the Information Gain by subtracting the weighted average of these subset entropies from our original total entropy (0.940).

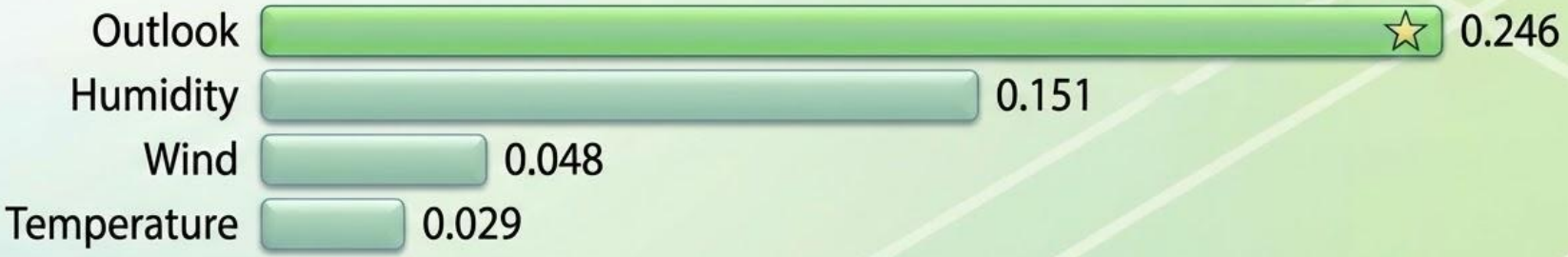
$$Gain(S, Outlook) = 0.940 - \left(\frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971\right)$$

→ **Gain(S, Outlook) = 0.246**

# Step 4 - Select the Winning Attribute

## Comparing the Gains

We repeat this exact same mathematical process for the other three attributes (Humidity, Wind, and Temperature) to see which provides the highest Information Gain.




**The Conclusion:** Because Outlook has the highest Information Gain (0.246), it provides the 'purest' initial split. Therefore, the algorithm selects **Outlook** as the absolute **Root Node** of our Decision Tree.

# Part 3: Tree Pruning

# Cutting the Noise to See the Forest

## Preventing Overfitting in Decision Trees

**Context:** Ensuring our models generalize to the real world.

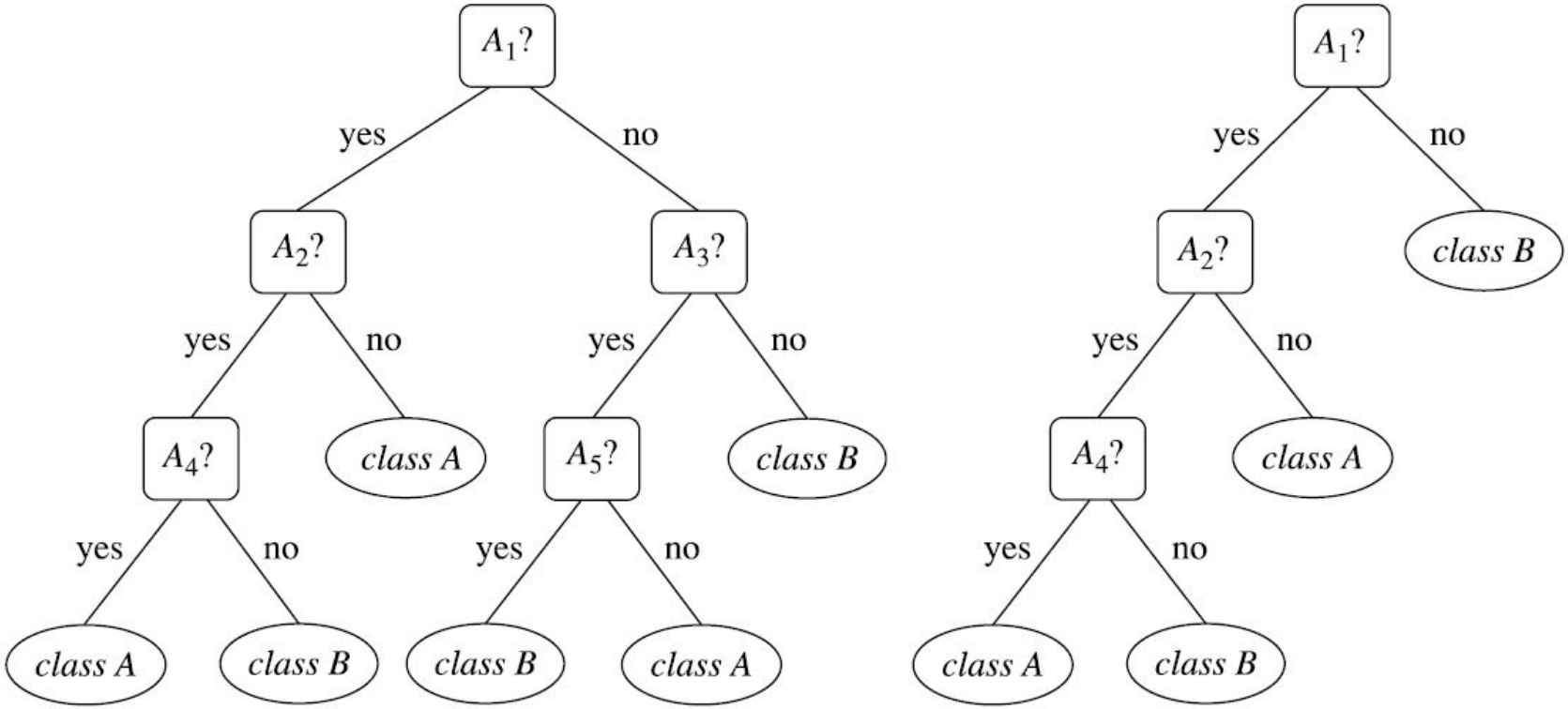



**FIGURE 6.7**

An unpruned decision tree (left) and a pruned version of it (right).

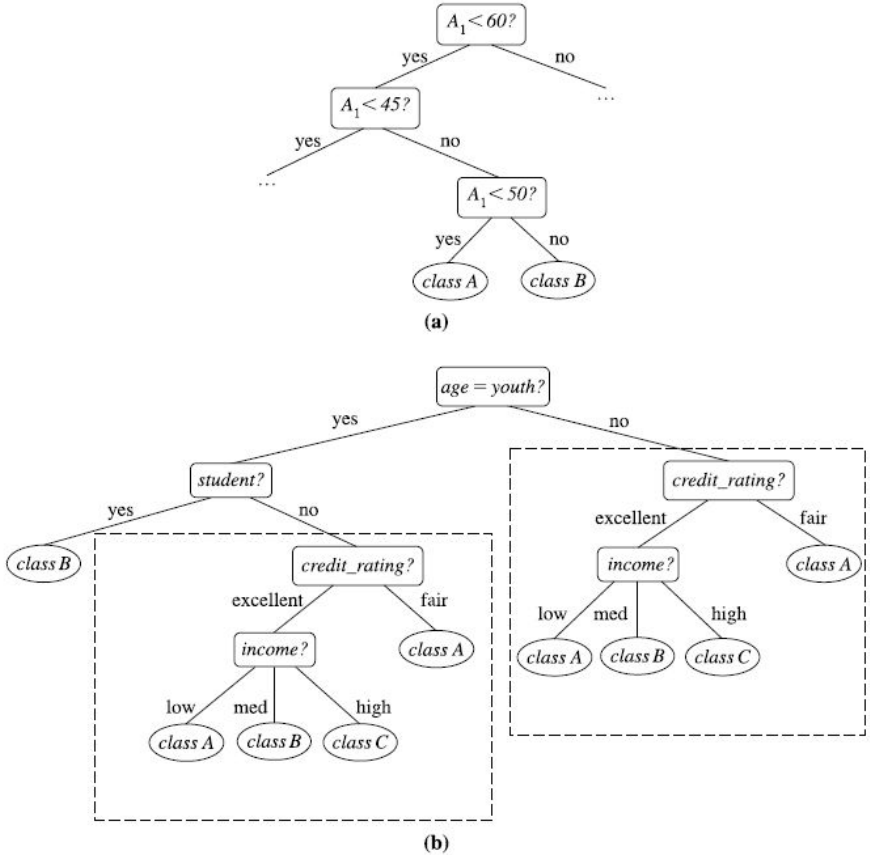



**FIGURE 6.8**

An example of (a) subtree **repetition**, where an attribute is repeatedly tested along a given branch of the tree (e.g., *age*), and (b) subtree **replication**, where duplicate subtrees exist within a tree (e.g., the subtree headed by the node "*credit_rating?*").

# The Overfitting Problem

## The Danger of Perfect Memorization

**The Trap:** Decision trees naturally want to grow until every single leaf node contains examples of only one class (perfect purity).

**The Result:** While this perfectly classifies the training data, the tree becomes too specialized and captures random noise rather than actual patterns.

**The Consequence:** The model will perform very poorly when introduced to new, unseen data in the real world. This phenomenon is called **overfitting.**

# What is Pruning?

## Trimming for Generalization

**The Definition:** In data mining, pruning is the process of selectively removing branches from a decision tree to improve its ability to generalize to new data.

**The Gardener Analogy:**

- Just as gardeners prune physical trees to remove dead branches, improve structure, and direct healthy new growth...

Data scientists prune decision trees to remove branches that capture useless noise, simplify the model's structure, and direct the learning toward actual, generalizable patterns.

# Pre-Pruning (Early Stopping)

## Halt Before It Gets Complicated

**The Concept:** Stop the tree's growth before it becomes fully grown.

**Common Stopping Criteria:**

- The node is entirely pure (only one class remains).
- There are no remaining attributes to split on.
- The number of instances in a node falls below a set threshold (e.g., < 5 examples).
- The Information Gain of a split falls below a minimum threshold (e.g., < 0.01).
- The tree reaches a maximum allowed depth limit.

**The Trade-off:**

⊕ **Advantage:** Much faster to build; avoids creating unnecessary branches.

⊖ **Disadvantage:** Risks stopping too early and missing complex patterns (underfitting).

# Post-Pruning

## Grow First, Trim Later

**The Concept:** Allow the tree to grow completely, and then evaluate and remove branches that do not improve accuracy.

**The Process:** The algorithm evaluates each internal node to see what happens if it replaces the entire subtree below it with a single leaf node. If validation accuracy improves (or stays the same), the branch is permanently cut.

**Common Methods:**

- Reduced Error Pruning: Uses a separate validation dataset to test the cuts.
- Cost-Complexity Pruning: Mathematically balances the tree's size against its training error.

**The Trade-off:**

⊕ **Advantage:** Highly reliable; excellent at finding the optimal tree complexity.

⊖ **Disadvantage:** Computationally expensive and requires holding back validation data.

# The Benefits of Pruning

## Setting Up for Long-Term Health

- **Removes the Dead Wood:** Eliminates branches that only capture random noise and anomalies in the training data.
- **Saves from Damage:** Prevents the catastrophic failure of overfitting when the model faces new data.
- **Green Light for Growth:** Simplifies the model, Simplifies the model, allowing it to apply broad, healthy, generalizable rules.
- **Strong Foundation:** Sets the decision tree up with a robust structure for long-term, highly accurate predictive performance in production.

Healthy, Pruned Tree

# Bringing It All Together

## Core Concepts of Classification and Decision Trees

**Classification Fundamentals:** Understanding the goal of categorizing data into predefined classes.

**Decision Tree Structure:** Hierarchical models with root, internal, and leaf nodes for decision-making.

**Tree Induction & Splitting:** Using metrics like Gini Impurity or Information Gain to find the best splits.

**Addressing Overfitting:** The critical role of pruning (pre- and post-) to ensure model generalization.

**Model Evaluation:** Assessing performance using metrics like accuracy, precision, and recall.

# Classification Fundamentals

## The Core Concepts

**Definition:** A supervised learning technique that assigns items to target categories based on patterns learned from historical data.


**Supervised Learning:** Strictly requires labeled training data to build the model.

**Binary vs. Multiclass:** Solves problems with two discrete outcomes (e.g., Spam / Not Spam) or multiple discrete outcomes (e.g., Weather / Finance / Sports).

# Decision Tree Advantages

## Why Use This Algorithm?

**Highly Interpretable:** The logic is transparent, making it easy to understand and explain to non-technical stakeholders.


**Handles Mixed Data:** Seamlessly processes both numeric and categorical attributes without requiring complex transformations.

**Built-In Feature Selection:** The algorithm naturally surfaces the most important attributes at the top of the tree, automatically filtering out irrelevant data.

# Key Insights for Practitioners

## Final Best Practices

**Tailor the Measure:** Choose the appropriate attribute selection measure based on your specific data characteristics.

**Always Prune:** Unpruned trees will memorize noise. Always prune to prevent overfitting and ensure real-world reliability.

**Validate Strictly:** Test your model thoroughly on a dedicated holdout set before deploying it into production.


**Find the Balance:** Optimize for the sweet spot between model complexity and predictive accuracy to achieve the best performance.