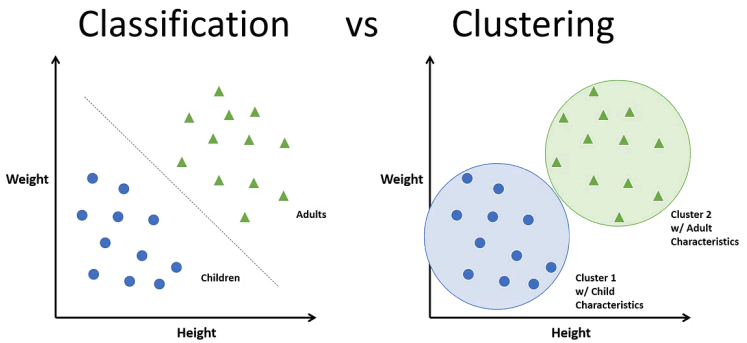
Classification

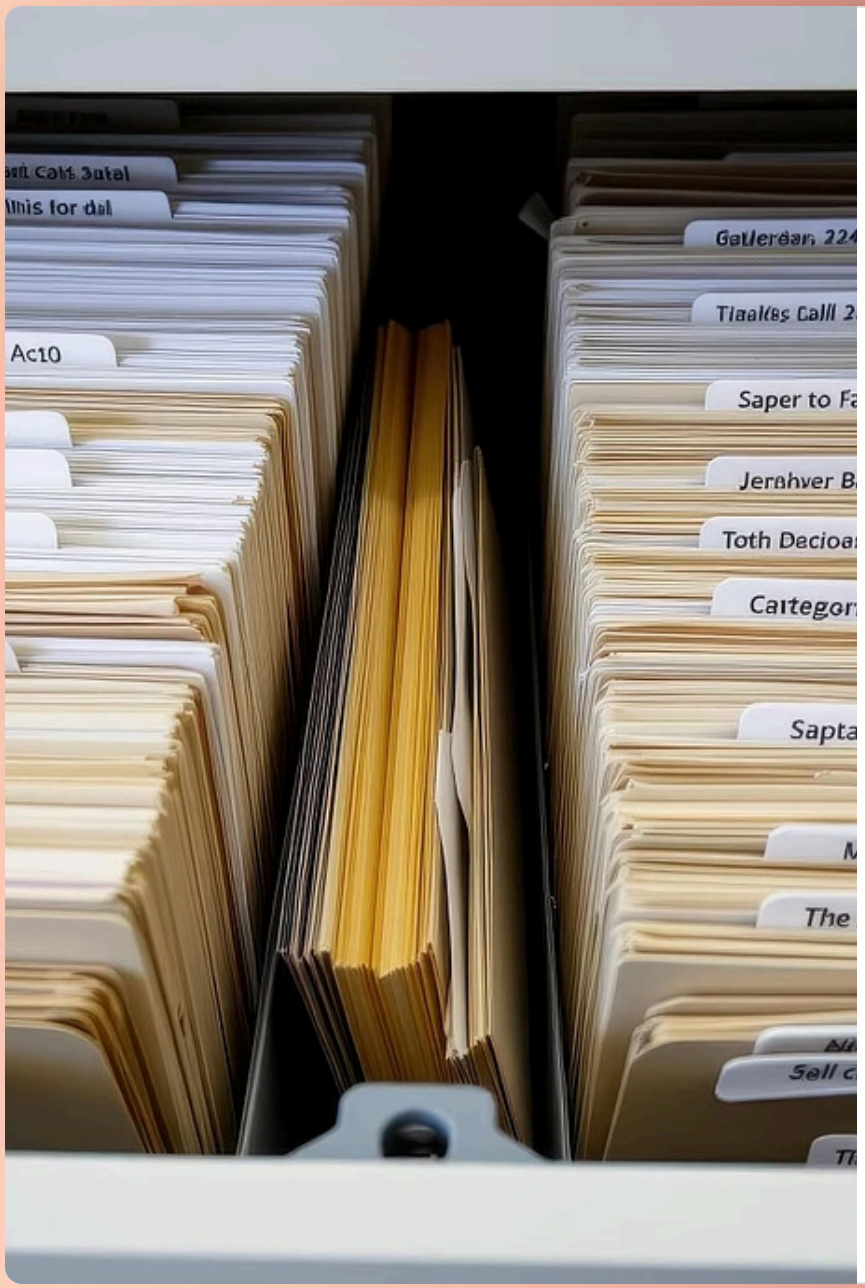
PROF. D. HERREMANS

INGAPORE UNIVERSITY O
ECHNOLOGY AND DESIG
stablished in collaboration with M



Data Science Tasks

Task	Supervised Methods	Unsupervised Methods
*Classification	✓	
*Regression	✓	
Causal Modeling	✓	
Similarity Matching	✓	✓
Link Prediction	✓	✓
Data Reduction	✓	✓
*Clustering		✓
Co-occurrence Grouping		✓
Profiling		✓

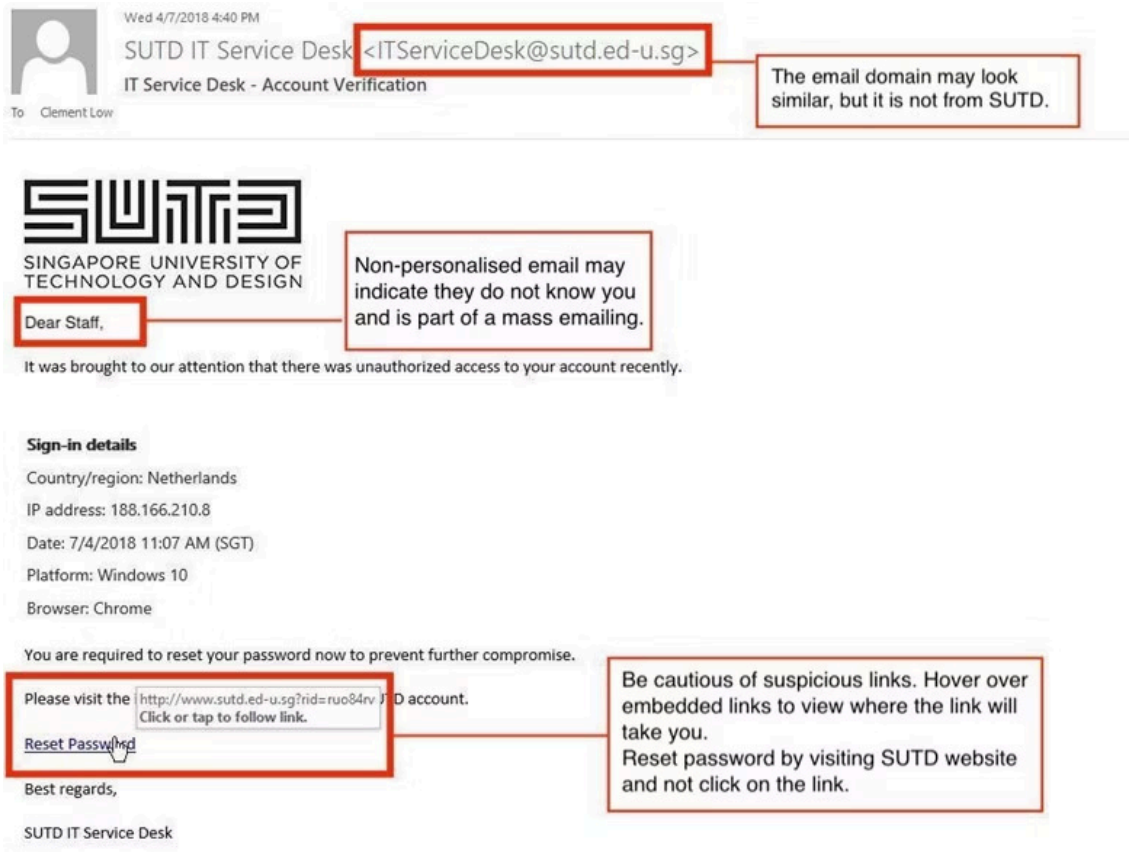


What is Classification?

> **"The act or process of dividing things into groups according to their type"**
>
> Cambridge Dictionary

Classification assigns labels to data based on learned patterns from training examples

Examples of Classification

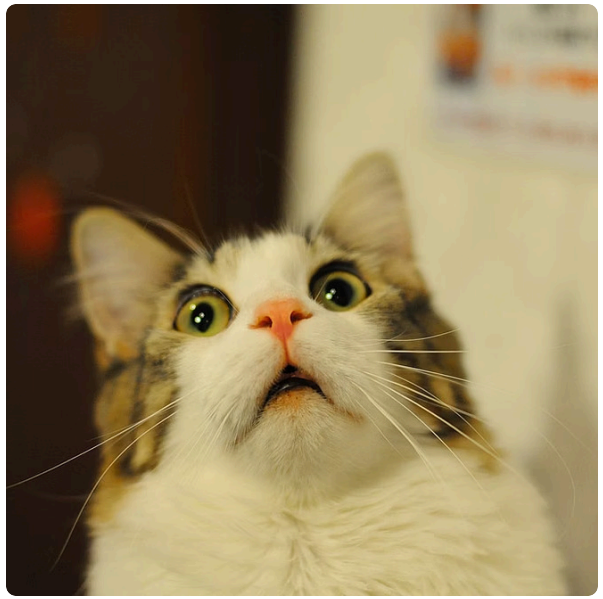


Email Filtering

Spam or legitimate?

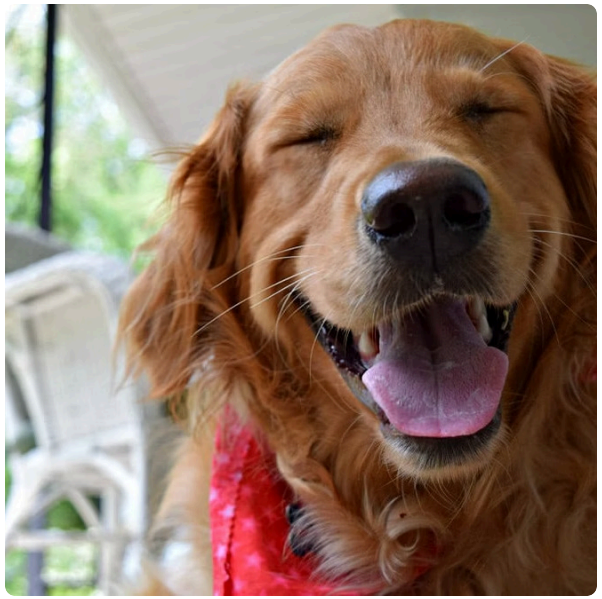
- Analyzes sender patterns
- Scans content for keywords
- Checks link authenticity
- Protects inbox integrity

Examples of Classification



Cat

Feline features: pointed ears, whiskers, retractable claws



Dog

Canine characteristics: floppy ears, snout shape, tail position

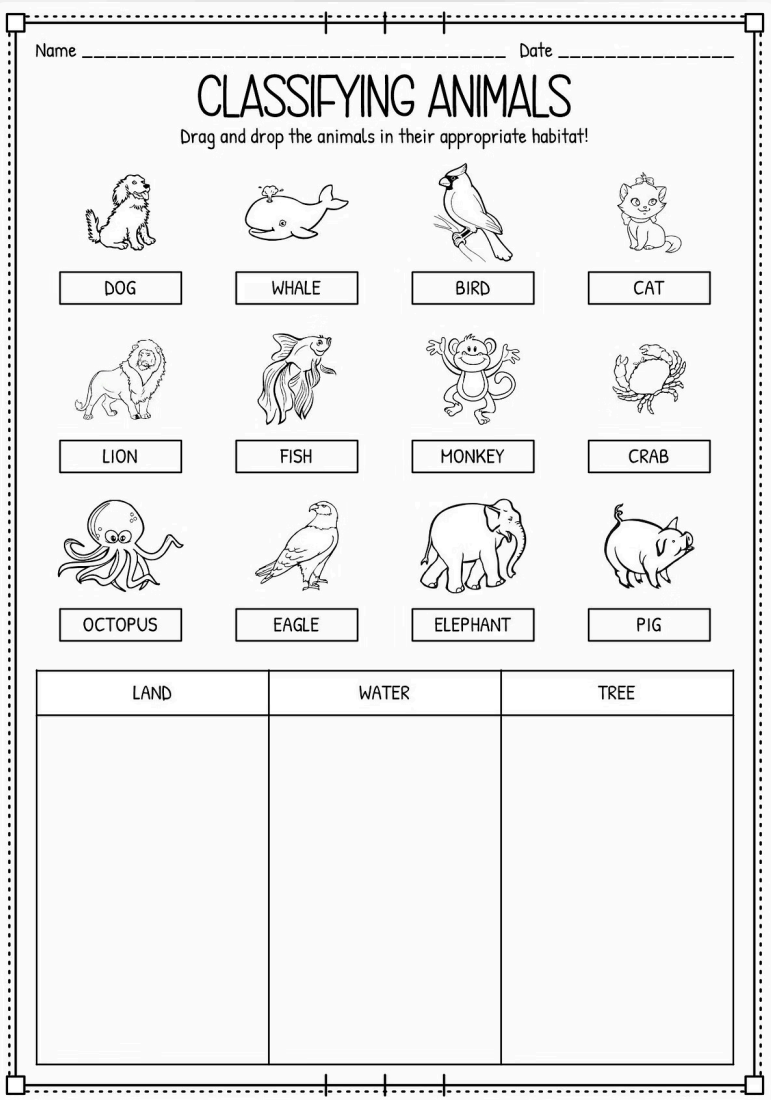
Image sources | Flickr

Examples of Classification

Traffic Analysis

Congestion detection systems

- Real-time vehicle counting
- Speed pattern recognition
- Route optimization alerts
- Predictive traffic modeling



More Classification Examples

Sentiment Analysis

Movie reviews: positive or negative?

Species Identification

Categorizing animals, plants, organisms

Geolocation Prediction

Origin detection from messages/posts

Music Genre Classification

Identifying song categories automatically

Types of Data Sources

Text

Documents, emails, social media content

Audio

Speech, music, environmental sounds

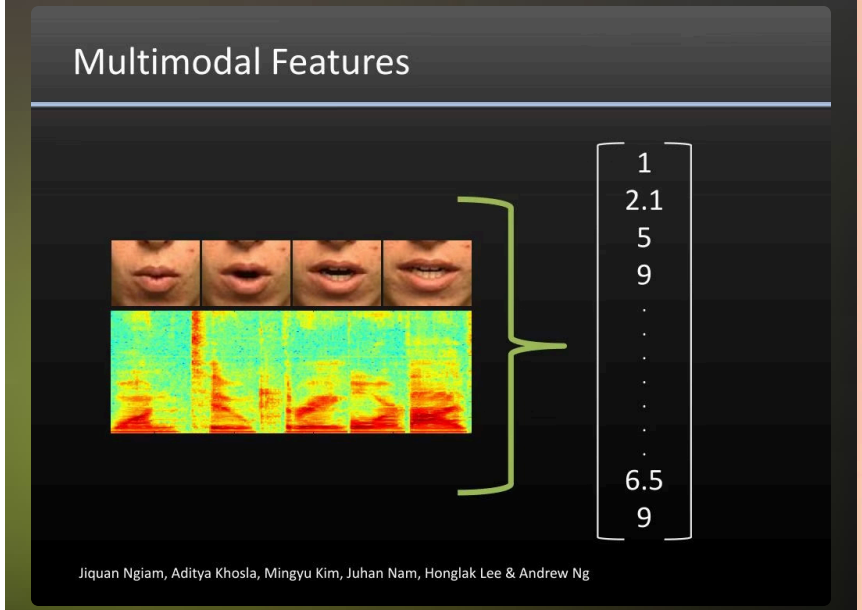
Images

Photos, diagrams, medical scans

Videos

Motion sequences, surveillance footage

Multimodal approaches combine multiple data types for enhanced accuracy



General Approach to Classification

A systematic workflow from data to predictions

1. Gather labeled training examples
2. Learn patterns and relationships (induction)
3. Predict labels for unseen data (deduction)

dreamstime.com
ID 300458776 © Topvectorr

Classification: Formal Definition

01

Training Set

Collection of records characterized by (x, y) where **x** is the attribute set and **y** is the class label

02

Variables

x: attributes, predictors, independent variables, inputs **y**: class, response, dependent variable, output

03

Learning Task

Build a model that maps each attribute set **x** to a predefined class label **y**

Classification Techniques

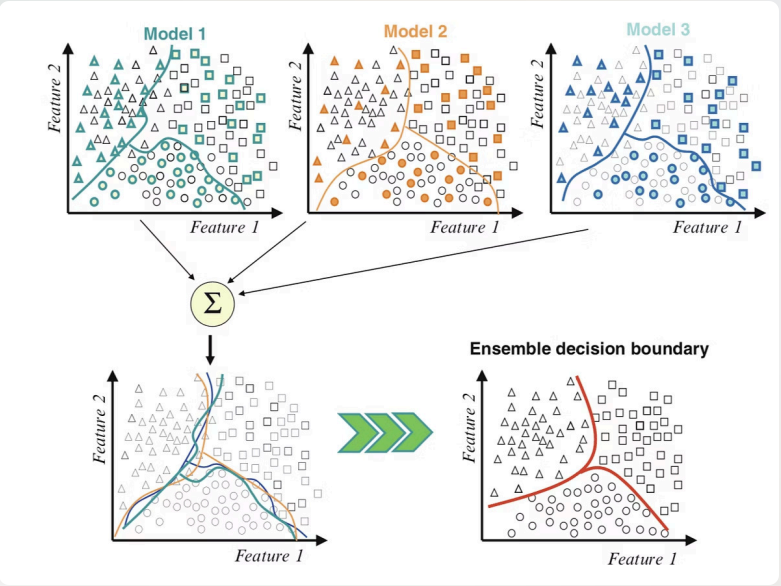
Base Classifiers

- Decision Trees
- Rule-based Methods
- Nearest-neighbor
- Logistic Regression
- Naïve Bayes
- Support Vector Machines
- Neural Networks
- Deep Learning

Ensemble Classifiers

Combining multiple models for improved performance

- Boosting
- Bagging
- Random Forests



Which classification method should we use?

The one that gives the best predictions...

- On the training data
- On the (unseen) test data
- On the (held-out) validation data

The one that is computationally efficient...

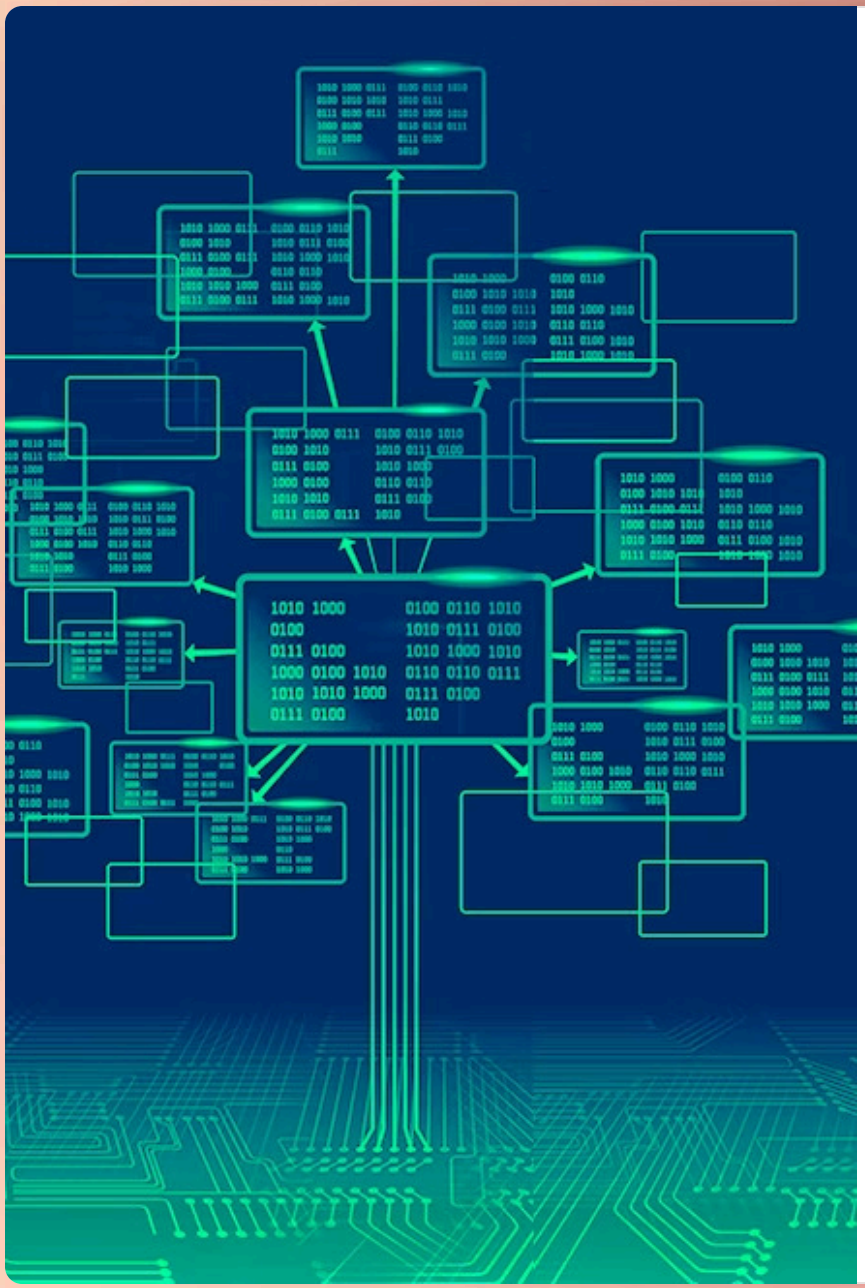
- During training
- During classification (inference)

The most interpretable one...

- In terms of its parameters
- As a model

The one that is easiest to implement...

- For learning
- For deployment



DECISION TREES

Decision Trees

Hierarchical models that partition data through sequential decisions

Decision Tree Structure

Components

Internal Node

Test of a specific attribute value

Branch

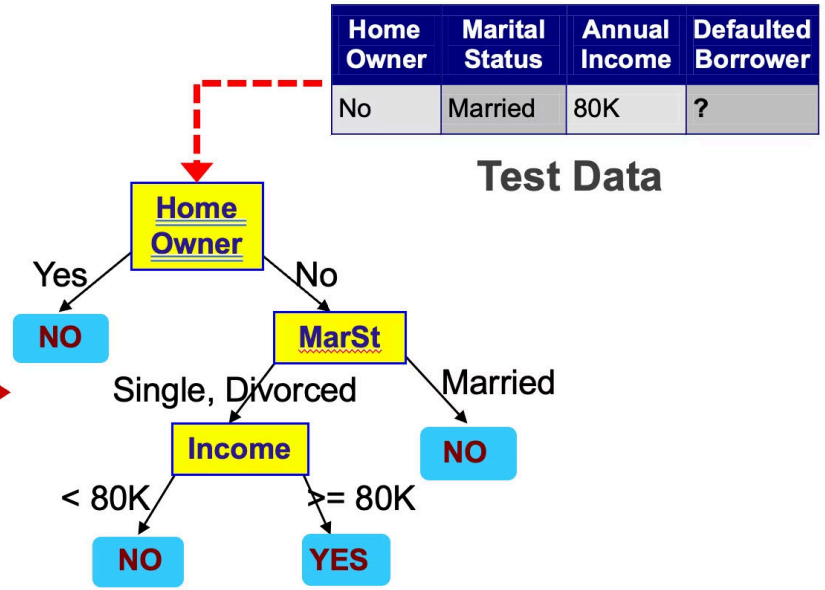
Outcome representing attribute value

Leaf Node

Final predicted class category

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

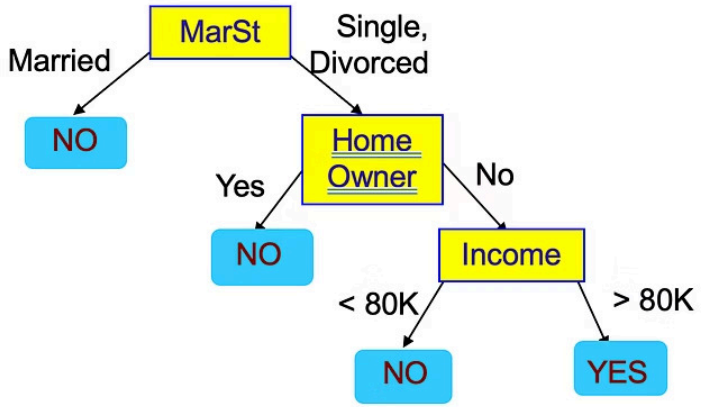
Training Data



Model: Decision Tree

Decision points guide data through the tree until reaching a classification

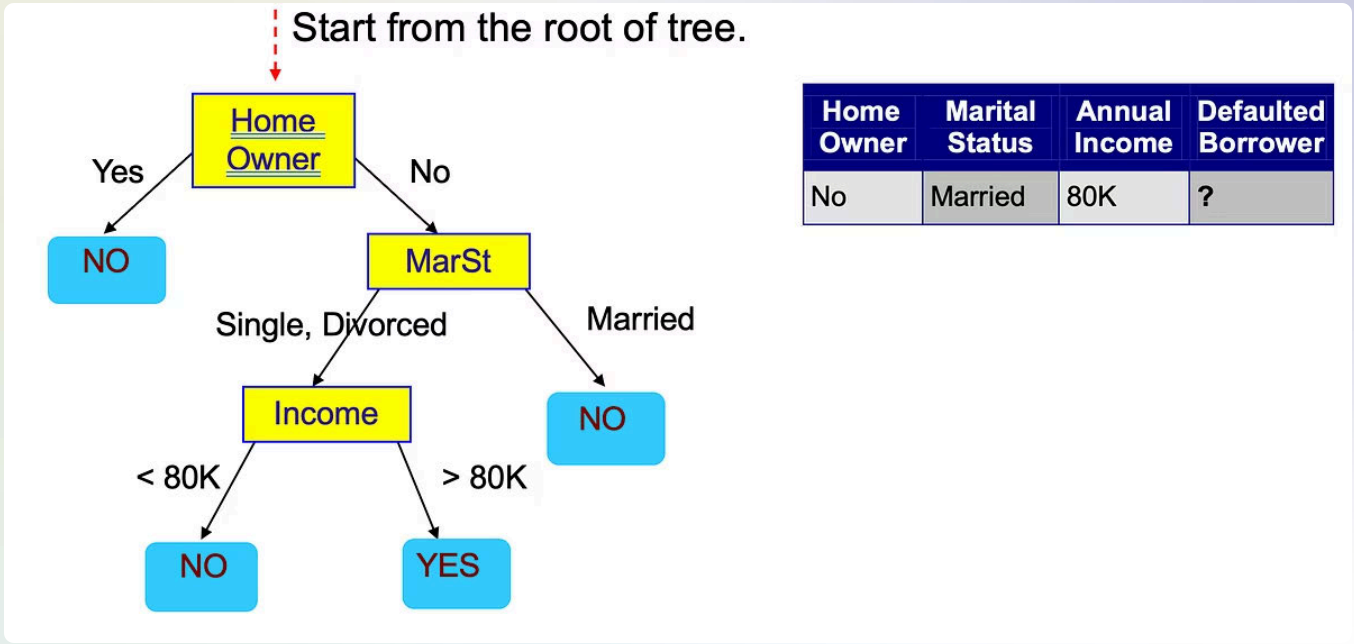
	categorical	categorical	continuous	class
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Multiple Trees for Same Data

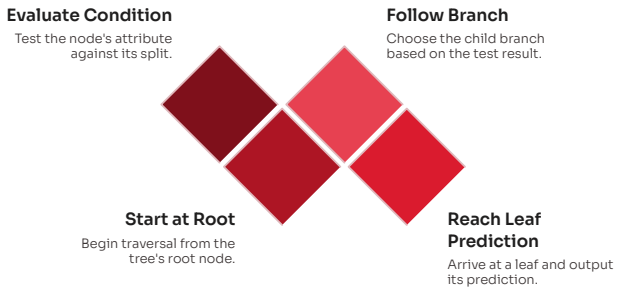
There could be more than one tree that fits the same data!

→ varying computational efficiency!



Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Applying the Model to Test Data



Decision Tree Classification

Trees make predictions by partitioning the feature space into regions

Hierarchical Structure

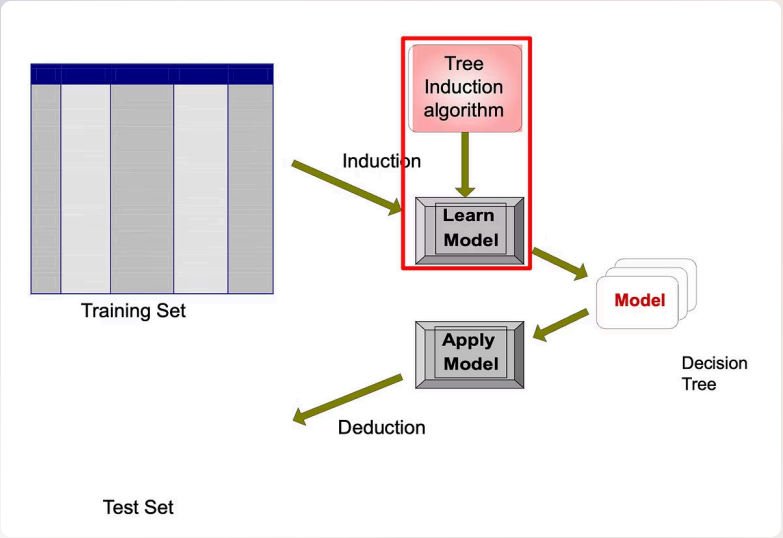
Sequential decisions from root to leaves

Clear Interpretability

Easy to understand and visualize

Handles Mixed Data

Works with categorical and continuous features



Candidate	Degree	Work Experience	Coding Test Score	Hired?
A	Master	High	Pass	Yes
B	Bachelor	Medium	Pass	No
C	PhD	Low	Pass	Yes
D	Master	Medium	Fail	No
E	Bachelor	High	Pass	Yes
F	Bachelor	Low	Fail	No
G	PhD	High	Pass	Yes
H	Master	Low	Pass	?

Building a Decision Tree: Hiring Example

Company evaluates candidates on three features

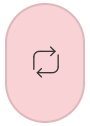
- Experience level
- Education background
- Interview performance

Goal: Classify as Hired (Yes/No)

Decision Tree Induction Algorithms

Hunt's Algorithm
One of the earliest, foundational approach

CART
Classification And Regression Trees



ID3
Iterative Dichotomiser 3

C4.5
Successor of ID3 with improvements

SLIQ
Supervised Learning in Quest

SPRINT
Scalable PaRallelizable INndution of decision Trees

Hunt's Algorithm Structure

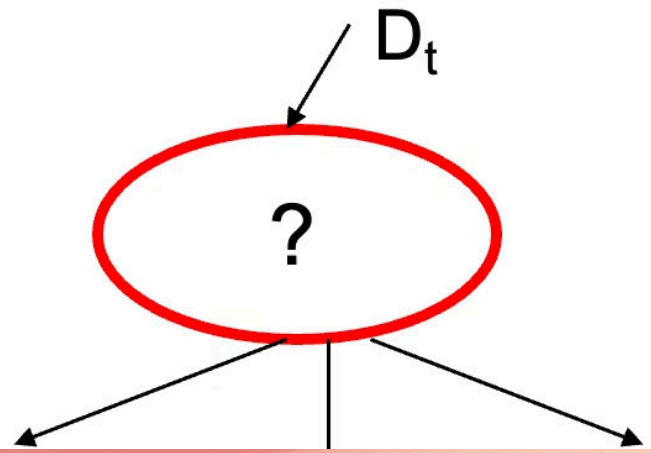
General Procedure

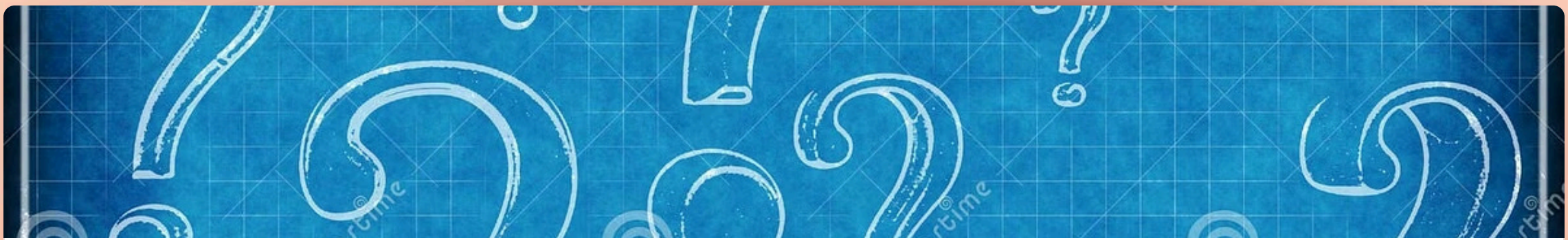
1. If D_t contains records that only belong to the class y_t, then t is a leaf node labeled as y_t
2. If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets.
3. Recursively apply the procedure to each subset.

Key Concepts

D_t: Training records reaching node t

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes





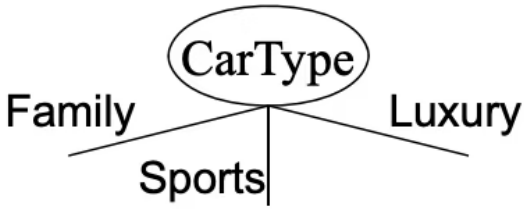
Design Issues in Tree Induction

How should the tree be split?

Attribute Selection	Test Condition	Split Arity	Stopping Criteria
Which feature to split on? How to determine best split?	How to specify the test condition? e.g., x < 1 or x+y < 1	Binary splits vs multi-way splits?	When to stop splitting? Early termination rules? e.g. *Stop splitting if all the records belong to the same class or have identical attribute values*

Splitting Nominal Attributes

Multi-way Split



Use as many partitions as distinct values

We need to find optimal partitioning...

Binary Split

Divide values into two optimal subsets

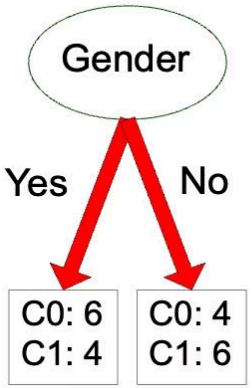
Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

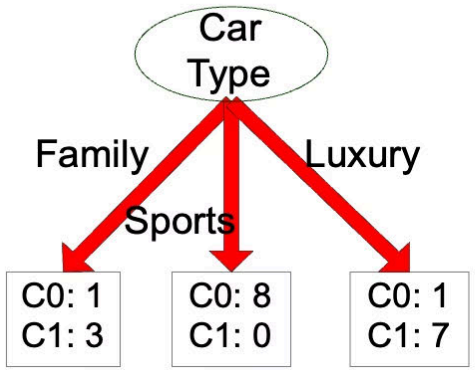
Determining the Best Split

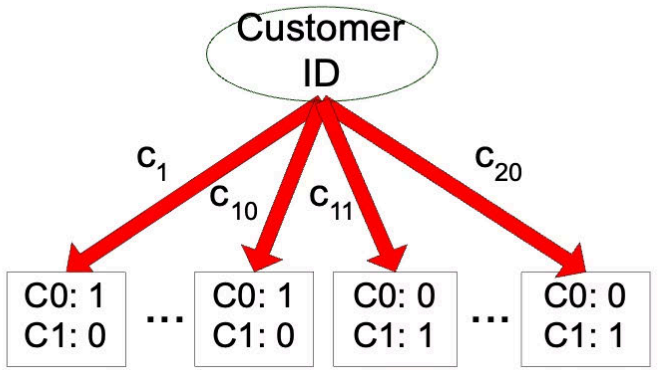
Before Splitting: 10 records of class 0, 10 records of class 1

Which test condition is optimal?

Need a metric to measure split quality







Node Purity Criterion

Greedy Strategy

Prefer nodes with **purer, more homogeneous** class distributions

> We need an impurity measure **M** to quantify this preference

High Impurity

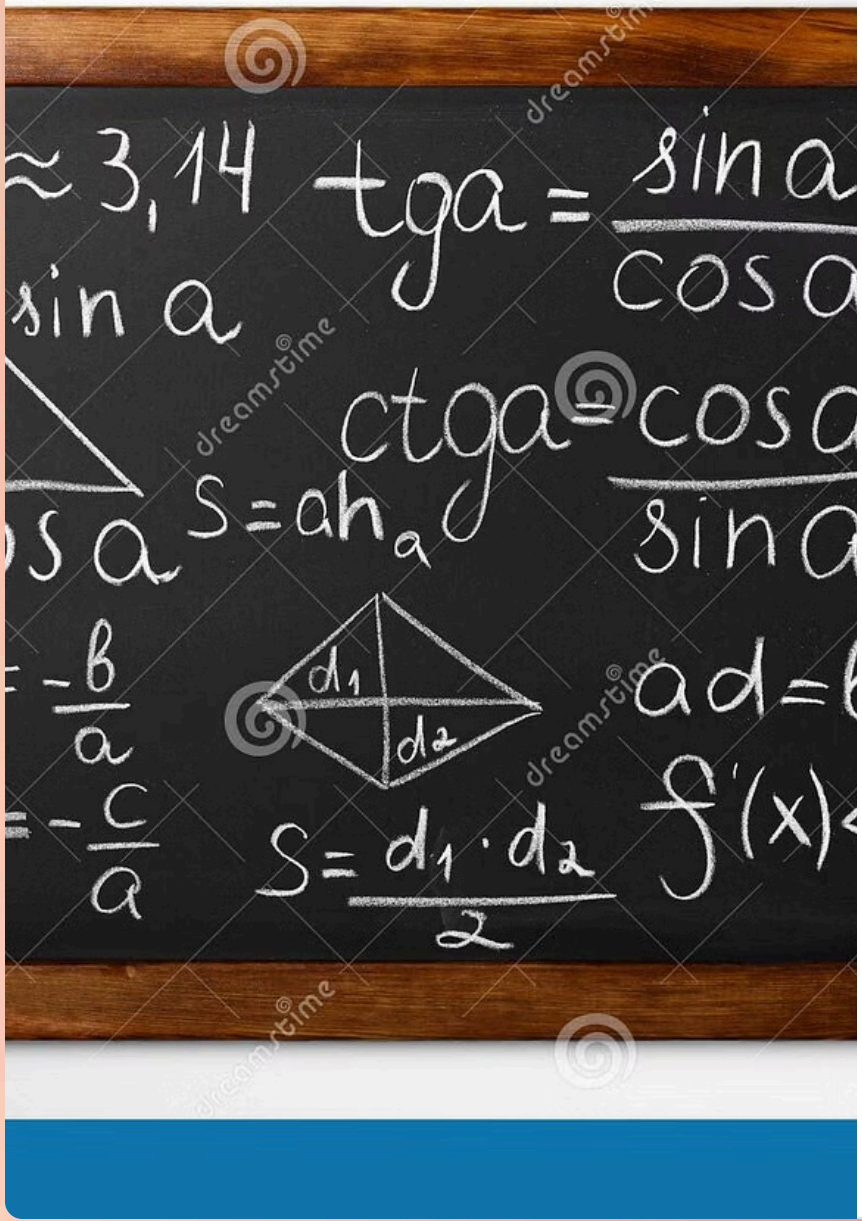
Non-homogeneous, mixed classes

C0: 5
C1: 5

Low Impurity

Homogeneous, single class dominates

C0: 9
C1: 1



Impurity Measures

Entropy

Information-theoretic measure of disorder

Gini Index

Probability of misclassification

Classification Error

Fraction of misclassified instances

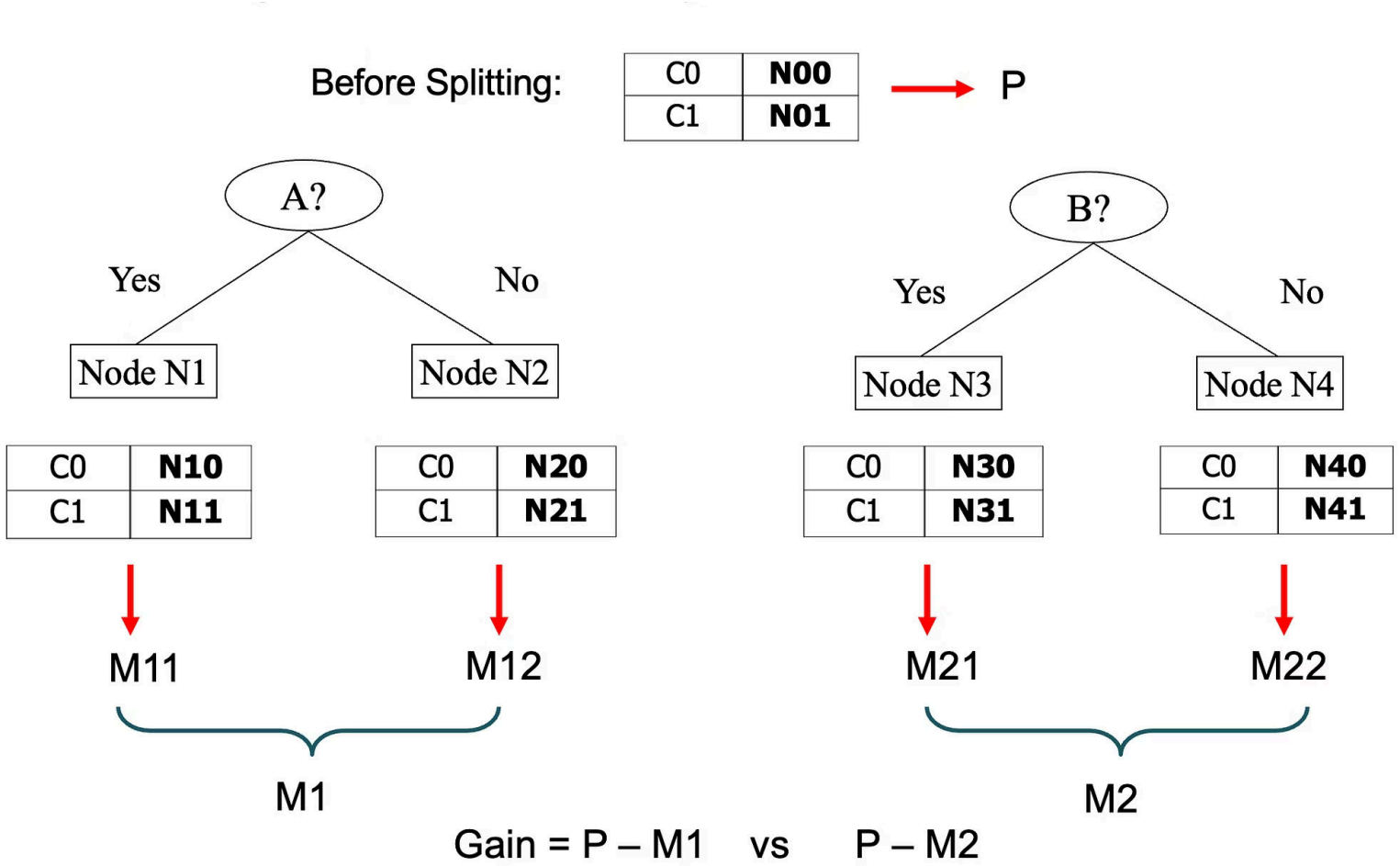
Each measure quantifies node purity differently, affecting tree structure

Finding the Best Split

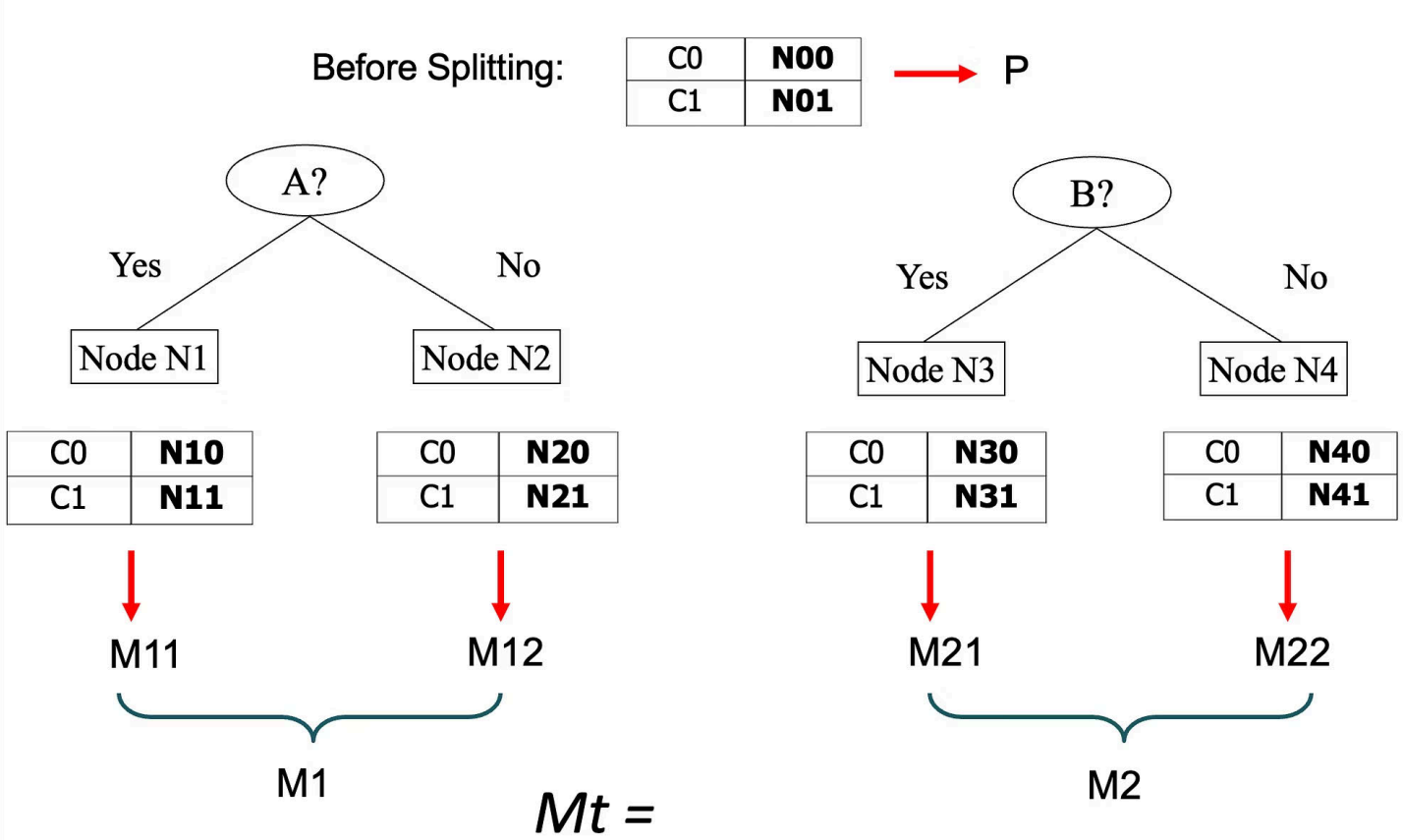
1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   a. Compute impurity measure of each child node
   b. M is the weighted impurity of children
3. Choose the attribute test condition that produces the highest gain or equivalently, lowest impurity measure after splitting (M)

$$\text{GAIN} = P - M$$

Finding the best split



Finding the best split



$$Mt = Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

Measure of Impurity: Entropy

Entropy at node t:

$$Entropy(t) = -\sum_{j} p(j|t) \log_2 p(j|t)$$

where p(j|t) is the relative frequency of class j at node t

Properties

- Maximum (1) when classes equally distributed
- Minimum (0) when all records in one class
- Similar computations to Gini Index

Higher entropy = more disorder = less information

Lower entropy = more order = more information

Computing Entropy: Examples

Pure Node

P(C1) = 0/6 = 0, P(C2) = 6/6 = 1

Entropy = -0 log 0 - 1 log 1
= -0-0=0

C1	**0**
C2	**6**

Low Impurity

P(C1) = 1/6, P(C2) = 5/6

Entropy = – (1/6) log2 (1/6) – (5/6) log2 (5/6)
= 0.65

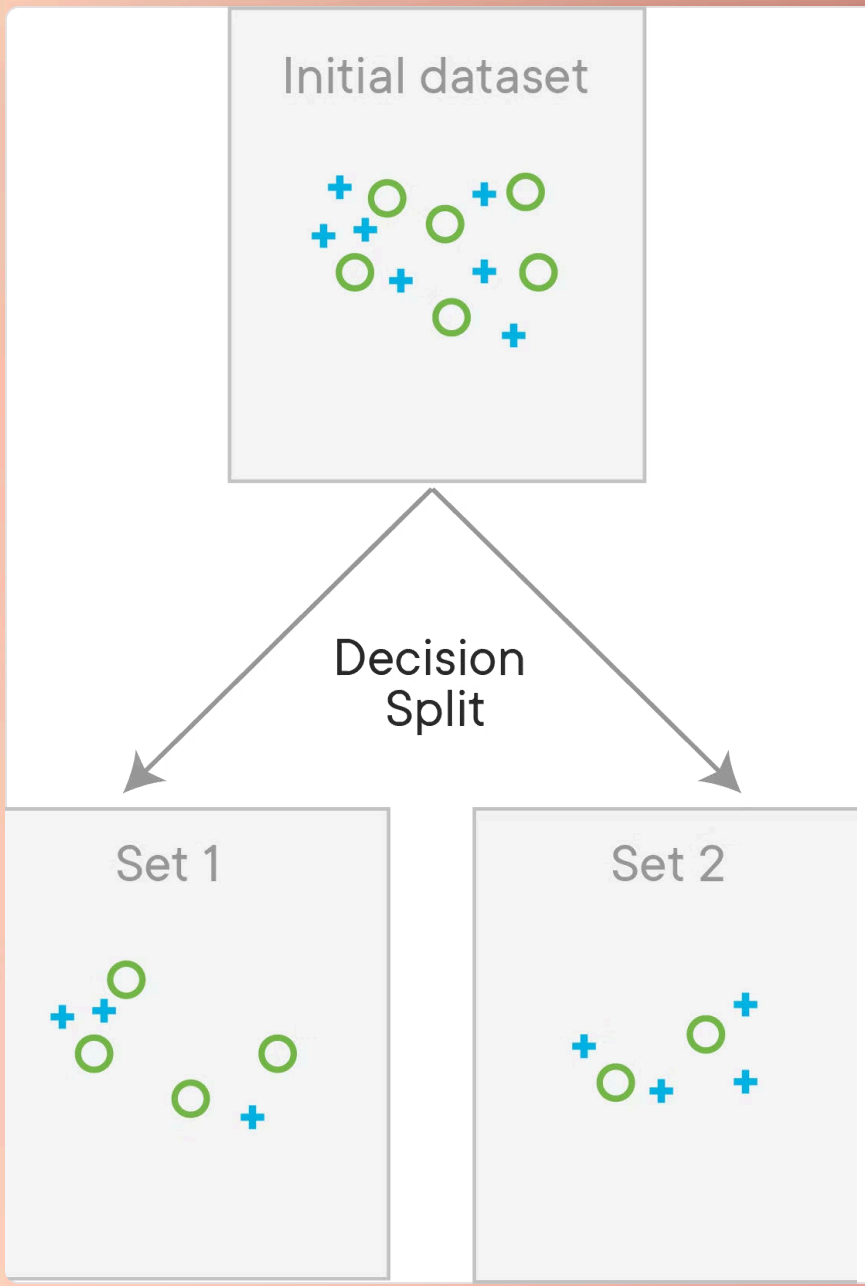
C1	**1**
C2	**5**

Higher Impurity

P(C1) = 2/6, P(C2) = 4/6

Entropy = – (2/6) log2 (2/6) – (4/6) log2 (4/6)
= 0.92

C1	**2**
C2	**4**



Information Gain

$$GAIN_{split} = Entropy(p) - \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)$$

- Parent Node, p is split into k partitions;
- n_i is number of records in partition i

Measures: Reduction in Entropy

Measures how much disorder decreases after split

Used for: Split Selection

Choose split that maximizes GAIN

Used in ID3 and C4.5

Foundation of classic algorithms

Disadvantage: Bias toward attributes with many distinct values

- Large trees with many branches are preferred
- What happens if there is an ID attribute?

Gain Ratio

$$GainRATIO_{split} = \frac{GAIN_{split}}{SplitINFO}$$

where

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

- Parent Node, *p* is split into *k* partitions
- n_i is the number of records in partition *i*

Purpose

Adjusts Information Gain by entropy of the partitioning (SplitINFO) itself

Higher entropy partitioning (large number of small partitions) is penalized!

Benefits

- Used in C4.5 algorithm
- Overcomes Information Gain bias
- More balanced attribute selection

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

A = 1	A= 2	A = 3	A = 4	*SplitINFO*
32	0	0	0	0
16	16	0	0	1
16	8	8	0	1.5
16	8	4	4	1.75
8	8	8	8	2

Split Information

SplitINFO measures the breadth and uniformity of a split

- Higher values when creating many partitions
- Normalizes Information Gain
- Prevents bias toward high-cardinality attributes

Other measures of impurity: Gini Index

Gini Index for node t:

$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

where p(j|t) is the relative frequency of class j at node t

Used in CART

Classification And Regression Trees algorithm

Computational Efficiency

Faster than entropy (no logarithms)

Probability-based

Measures expected classification error

$Error(t) = 1 - \max_i P(i|t)$

Computing Gini Index of a single node: Examples

$GINI(t) = 1 - \sum_j [p(j|t)]^2$

Pure Node

P(C1) = 0, P(C2) = 1

Gini = 1 – P(C1)2 – P(C2)2 = 1 – 0 – 1 = 0

C1	**0**
C2	**6**

Low Impurity

P(C1) = 1/6, P(C2) = 5/6

Gini = 1 – (1/6)2 – (5/6)2 = 0.278

C1	**1**
C2	**5**

Higher Impurity

P(C1) = 2/6, P(C2) = 4/6

Gini = 1 – (2/6)2 – (4/6)2 = 0.444

C1	**2**
C2	**4**

Computing Error of a Single Node

Error at node t:

$$Error(t) = 1 - \max_i [p(i|t)]$$

Pure Node

P(C1) = 0, P(C2) = 1

Error = 1 – max (0, 1) = 0

C1	**0**
C2	**6**

Low Error

P(C1) = 1/6, P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1/6

C1	**1**
C2	**5**

Higher Error

P(C1) = 2/6, P(C2) = 4/6

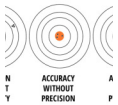
Error = 1 – max (2/6, 4/6) = 1/3

C1	**2**
C2	**4**

Choosing an Impurity Measure

Fast Computation

Gini Index (CART) or Classification Error



Precise Reduction

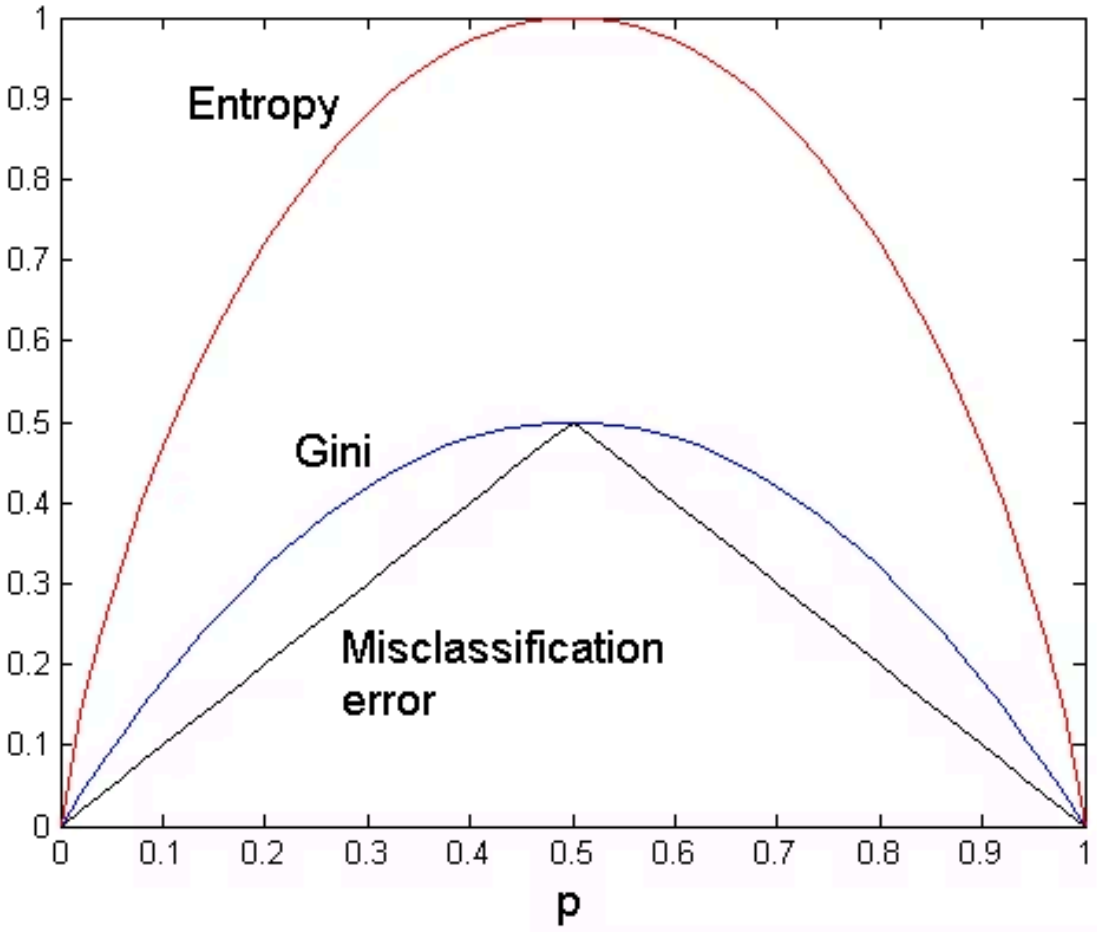
Entropy & Information Gain (ID3, C4.5)

Many Categories

Gain Ratio (C4.5) reduces bias

Simple Rule

Classification Error for straightforward splits



Comparing Impurity Measures

Two-Class Problem

All three measures show similar trends

- Entropy and Gini closely aligned
- Classification error less sensitive
- Maximum at 50-50 class distribution
- Minimum at pure nodes

Practical Implementation Issues

Decision trees are built by greedy search algorithms, guided by some heuristic that measures “impurity”.

In real-word applications we need also consider:

Continuous Attributes

Require discretization or threshold selection

Missing Values

Handled during preprocessing or imputation

Computational Efficiency

Optimization for large datasets

Overfitting

Trees that memorize training data

Handling Continuous Attributes

Discretization Approaches

Static

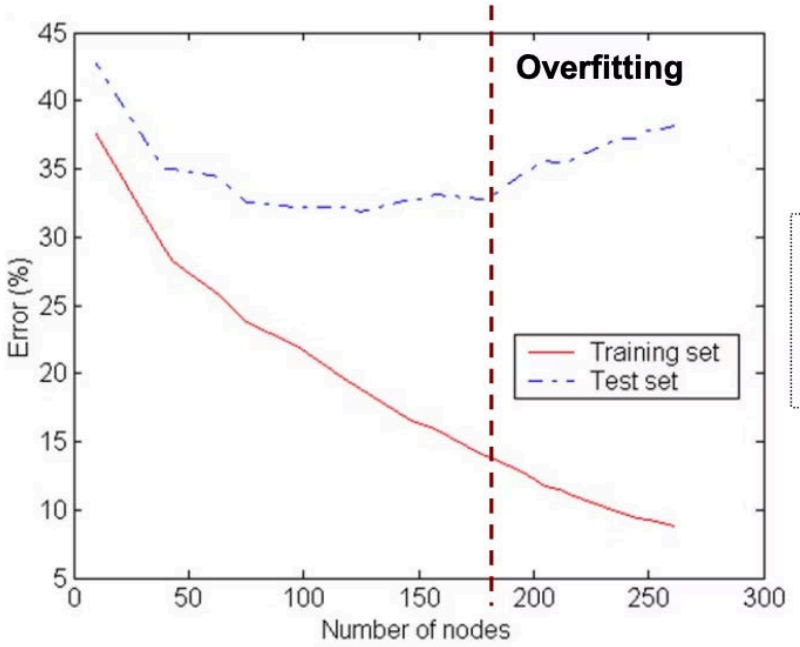
Discretize once at beginning (faster)

Dynamic

Repeat at each node (more accurate, computationally intensive)

Methods

- Find ranges using:
  - Equal interval bucketing
  - Equal frequency bucketing (percentiles)
  - Clustering
- Binary decision: $(A < v)$ or $(A \geq v)$:
  - consider **all** possible splits and finds the best cut
  - can be more computationally intensive



When the tree becomes too large, its test error rate begins increasing while its training error rate continues too decrease.

What causes overfitting?

- Noise

Underfitting: when model is too simple, both training and test errors are large

Overfitting and Underfitting

Underfitting

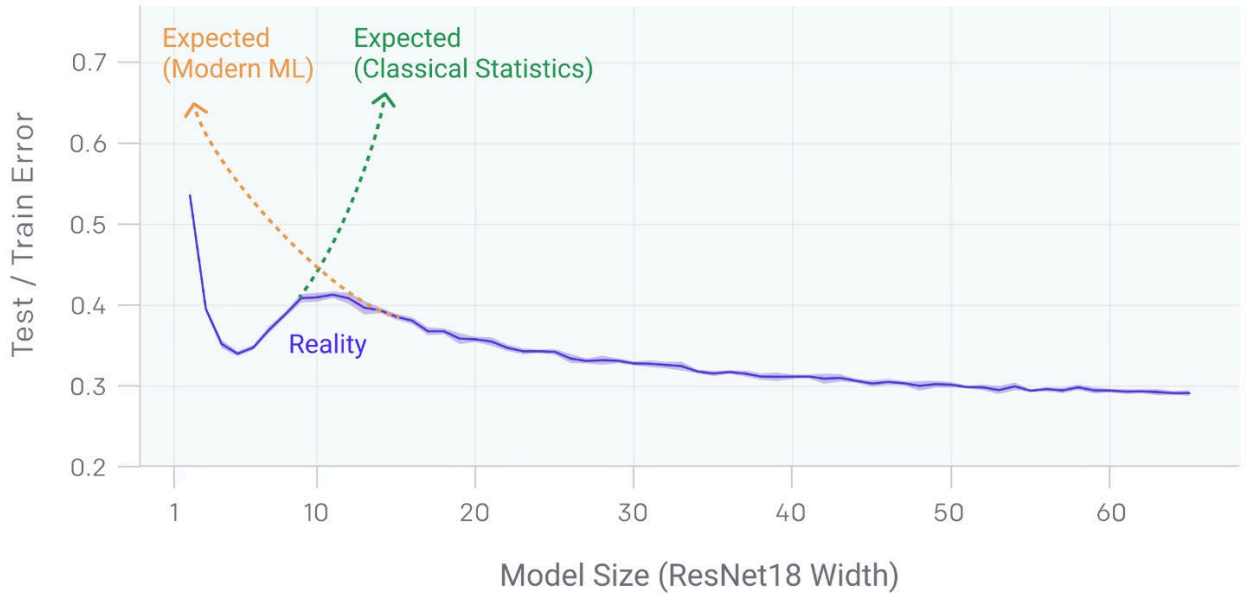
Model too simple, misses patterns

Good Fit

Balanced complexity, generalizes well

Overfitting

Model too complex, memorizes noise



Deep Double Descent

Recent discovery in model complexity

Error decreases, then increases, then decreases again as model capacity grows

Challenges classical bias-variance tradeoff

Source: OpenAI Research

Addressing Overfitting: Pre-Pruning

Early stopping prevents fully-grown trees

Class Purity

Stop if all instances belong to same class

→ fully grown tree

Attribute Exhaustion

Stop if all attribute values identical

→ fully grown tree

Instance Threshold

Stop if fewer than 5-10 instances per node

→ early stopping

Statistical Test

Stop if ◦ class distribution of instances are independent of the available attributes (e.g., using Chi Sq test)

→ early stopping

Impurity Threshold

Stop if splitting the tree improves the gain below minimum threshold

→ early stopping

Addressing Overfitting: Post-Pruning

Process

1. Grow full decision tree
2. Trim nodes bottom-up
   Reduced-error pruning *[remove subtrees if error does not increase]*
3. Use separate pruning dataset
4. Evaluate error improvement

Requires three datasets: training, testing, pruning

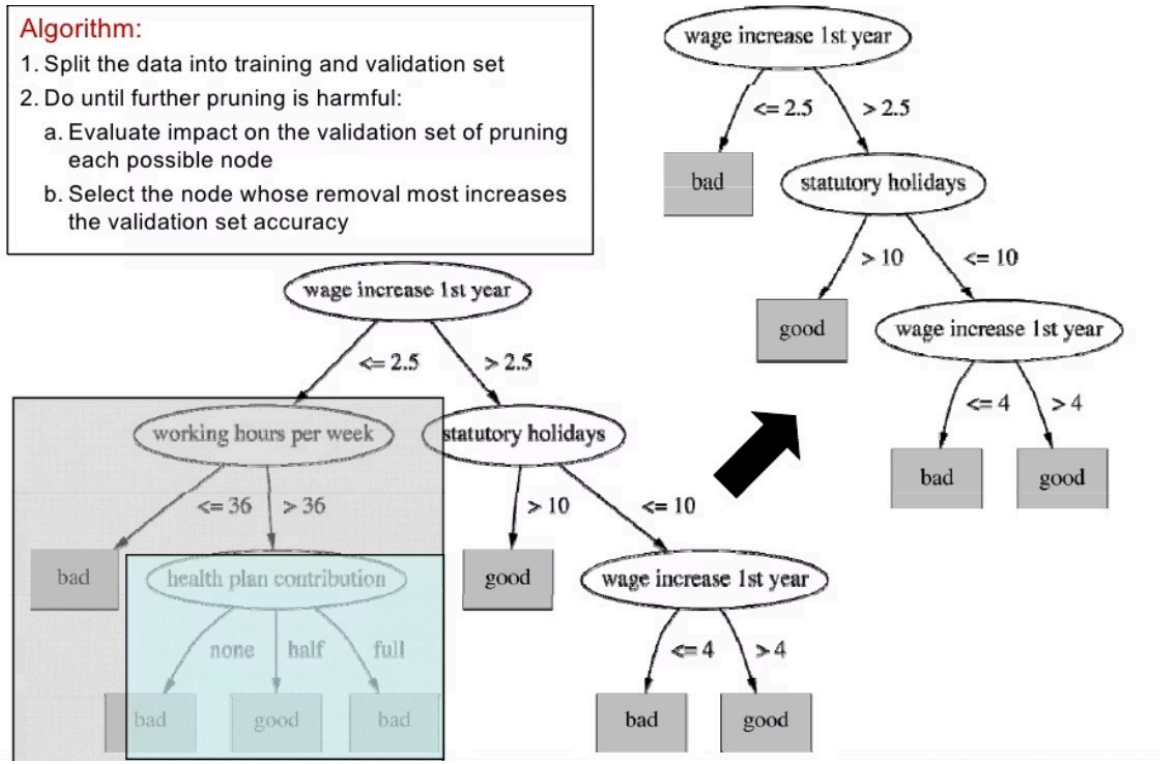
Two Post-Pruning Methods

Subtree Replacement

Replace subtree with single leaf

Subtree Raising

Replace subtree with child subtree

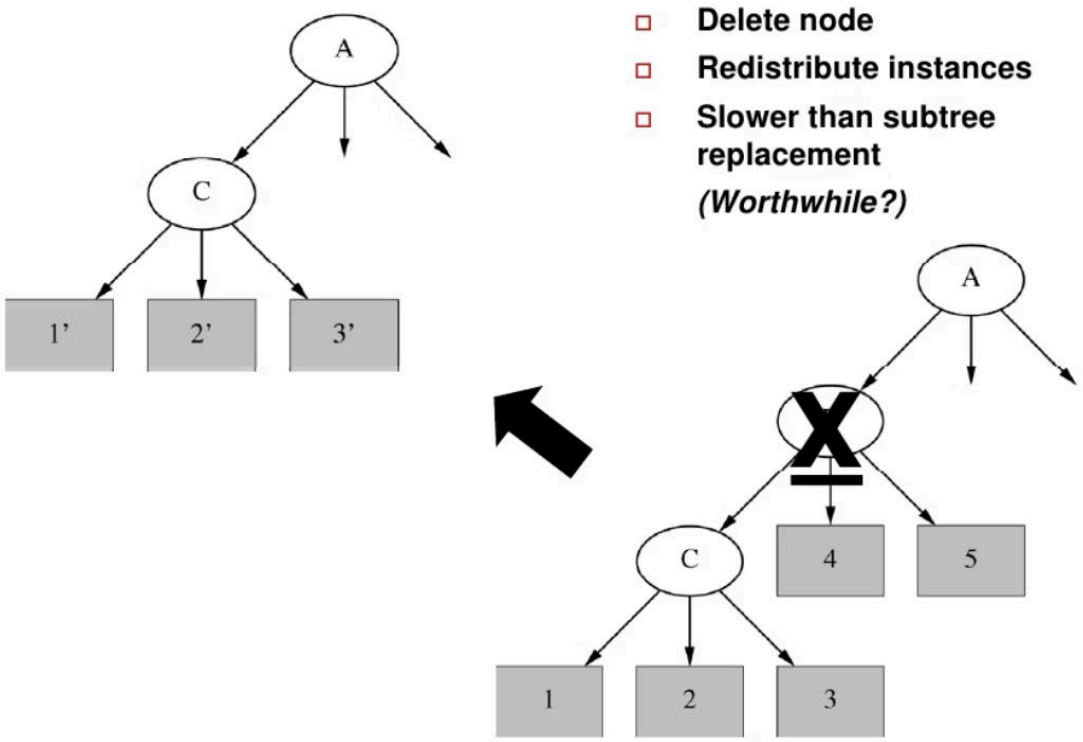


Subtree Replacement

Selects a subtree and replaces it with a single leaf node (most common label)

- Simplifies complex branches
- Tests error on pruning set
- Keeps replacement if error doesn't increase

Source: SlideShare



Subtree Raising

Selects a subtree and replaces it with one of its child subtrees

- Promotes promising branches
- Removes less informative siblings
- More aggressive than replacement

Source: SlideShare

Major Decision Tree Algorithms

1. **ID3**

   Information gain criterion, Ross Quinlan's foundational work

2. **C4.5 (1994)**

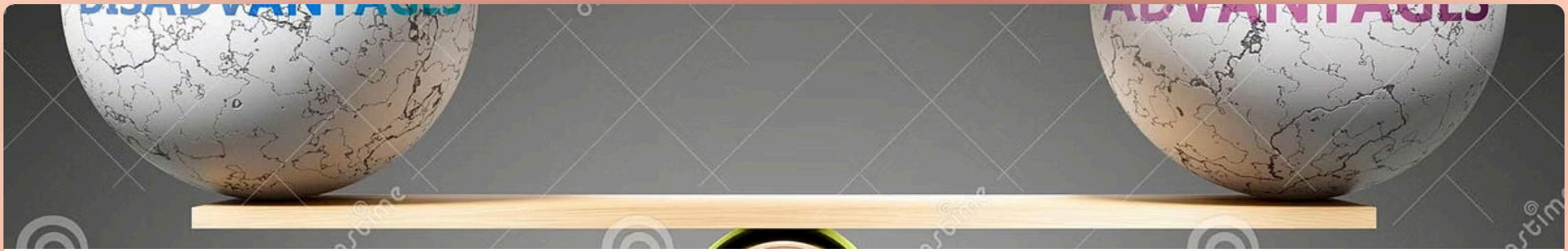
   Improves ID3: handles missing values, continuous attributes, post-pruning

3. **C5.0**

   Commercial successor with enhanced performance

4. **CART**

   Breiman et al.: binary trees, supports regression, uses Gini Index



Decision Tree Classification: Summary

Advantages

- Inexpensive to construct
- Extremely fast classification
- Easy to interpret (small trees)
- Robust to noise with pruning
- Handles redundant attributes

Disadvantages

- Exponential search space
- Greedy approach limitations - sometimes cannot find best tree
- Ignores attribute interactions
- Single-attribute boundaries only

RULE-BASED METHODS

Rule-Based Classification

IF-THEN rules that explicitly define decision logic

Example Rule Set

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Rule Covering

A rule **r** covers instance **x** if x satisfies the rule's condition (LHS)

Examples

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds

R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes

R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals

R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles

R5: (Live in Water = sometimes) → Amphibians

The rule R1 covers a hawk ⇒ Class = Bird

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

Consider the rule set

- Attributes A, B, C, and D can have values 1, 2, and 3

$$A = 1 \wedge B = 1 \rightarrow Class = Y$$
$$C = 1 \wedge D = 1 \rightarrow Class = Y$$
$$Otherwise, Class = N$$

How to represent it as a decision tree?

- The rules need a common attribute

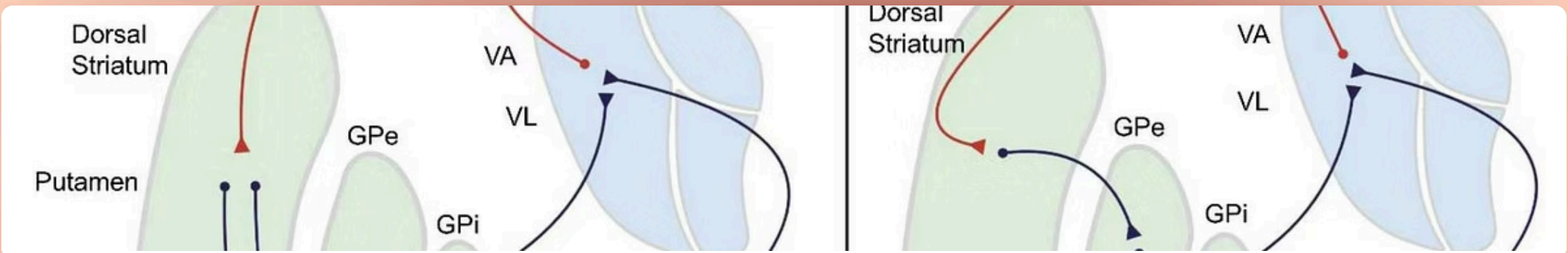
$$A = 1 \wedge B = 1 \rightarrow Class = Y$$
$$A = 1 \wedge B = 2 \wedge C = 1 \wedge D = 1 \rightarrow Class = Y$$
$$A = 1 \wedge B = 3 \wedge C = 1 \wedge D = 1 \rightarrow Class = Y$$
$$A = 2 \wedge C = 1 \wedge D = 1 \rightarrow Class = Y$$
$$A = 3 \wedge C = 1 \wedge D = 1 \rightarrow Class = Y$$
$$Otherwise, Class = N$$

From Rules to Decision Trees

Rules can be converted to tree representation

Decision trees are more verbose equivalents of rule sets

Each path from root to leaf represents one rule



Building Classification Rules

Direct Methods

Extract rules directly from data

- RIPPER algorithm
- Holte's 1R (OneR)
- Sequential covering

Indirect Methods

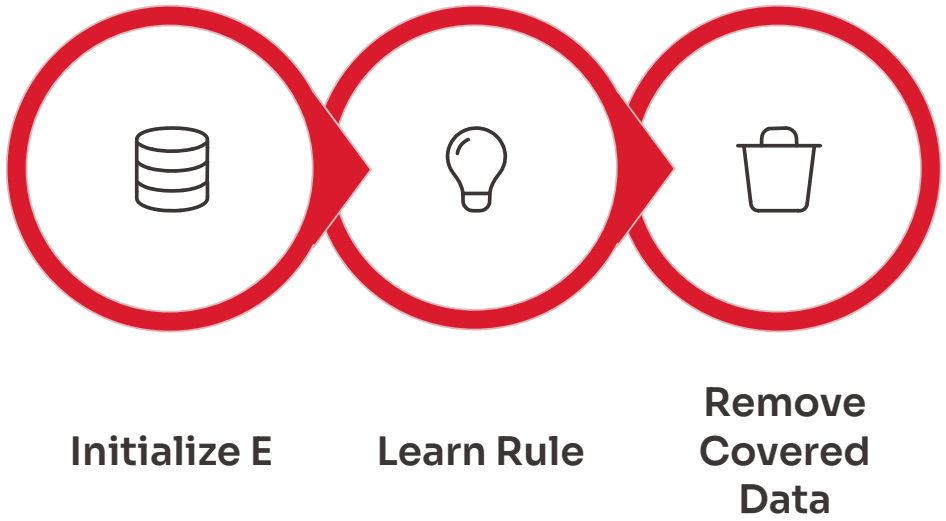
Extract rules from other models

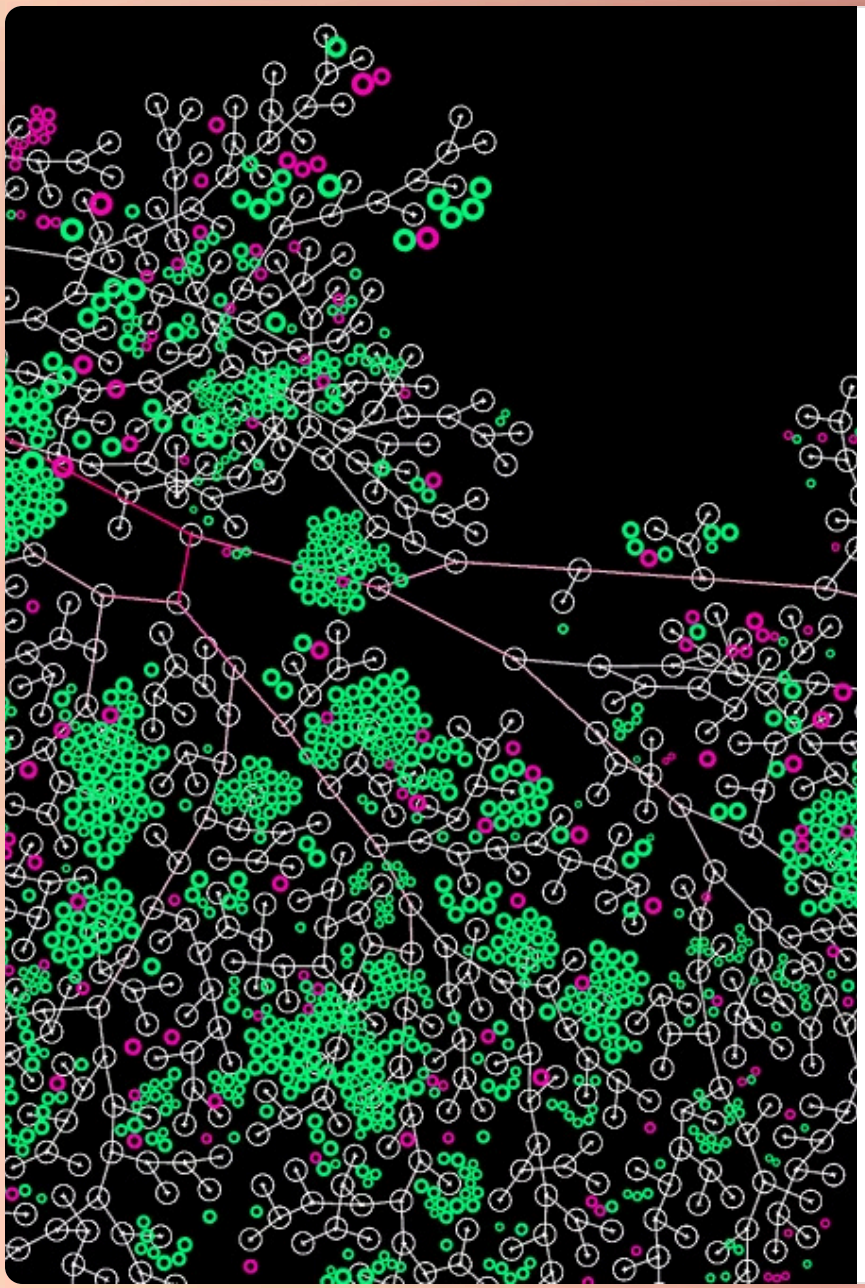
- C4.5rules (from decision trees)
- Convert tree paths to rules
- Simplify and optimize

Sequential Covering Algorithm

- Let E be the training set
- Extract rules one class at a time
  - For each class C
    - Initialize set S with E
    - While S contains instances in class C
      - Learn one rule R for class C
      - Remove training records covered by the rule R

Goal: to create rules that cover many examples of a class C and none (or very few) of other classes





RIPPER Algorithm

Repeated Incremental Pruning to Reduce Error

01

For a Two-Class Setup:

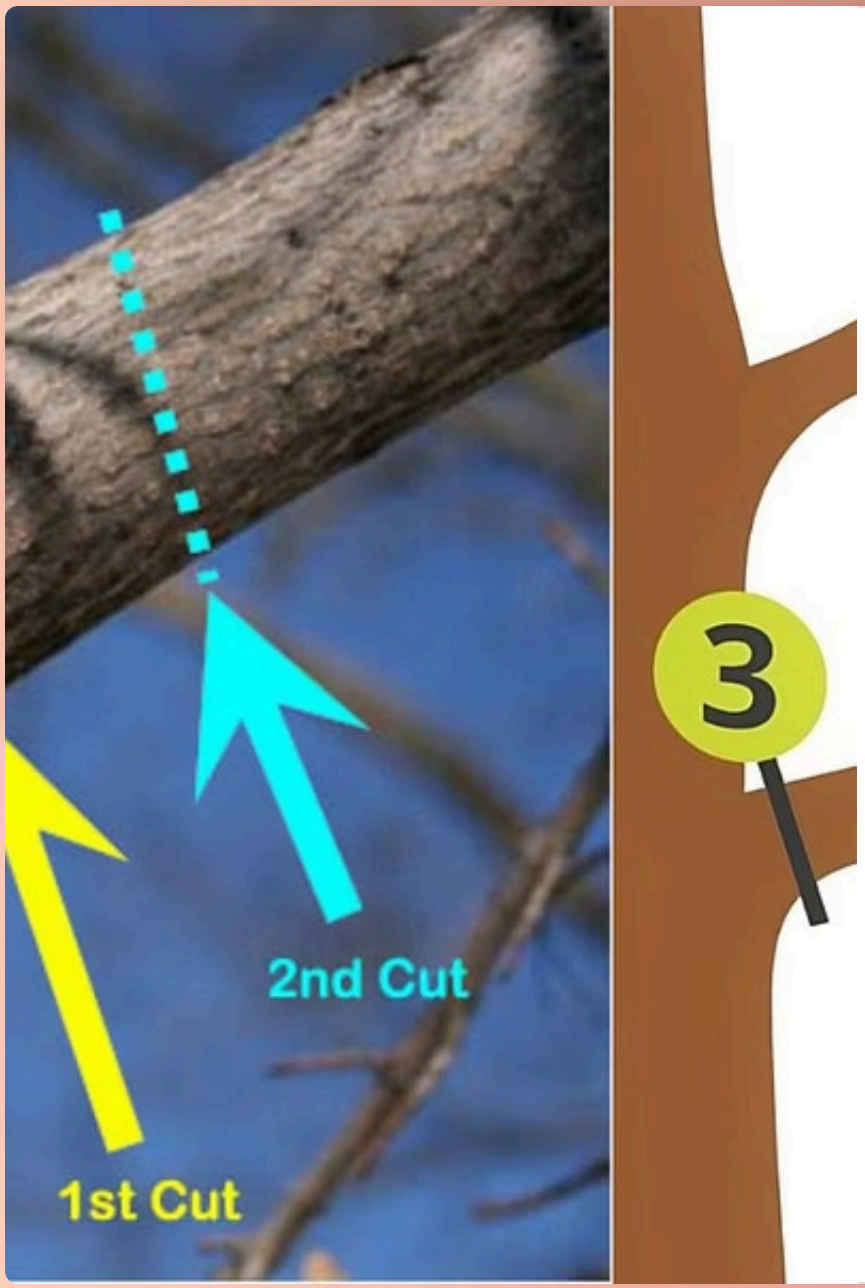
Choose one of the classes as positive class, and the other as negative class

- Learn rules for positive class
- **Negative class will be default class**

02

For a Multi-Class Strategy:

- Order classes by increasing prevalence (fraction of instances that belong to a particular class)
- Learn the rule set for smallest class first, treat the rest as negative class
- Repeat with next smallest class as positive class



RIPPER: Rule Construction

Start Empty
Begin with empty rule

Add Conjuncts
Add conditions improving FOIL information gain

Stop Condition
Stop when no longer covering negative examples

Prune Immediately
Apply reduced error pruning using validation set

Pruning measure: **W(R) = (p-n)/(p+n)** where:

- p: number of positive examples covered by the rule in the validation set
- n: number of negative examples covered by the rule in the validation set

Pruning starts from the last test added to the rule

- May create rules that cover some negative examples (accuracy < 1)
- A global optimization (pruning) strategy is also applied

RIPPER: Optimization Strategy

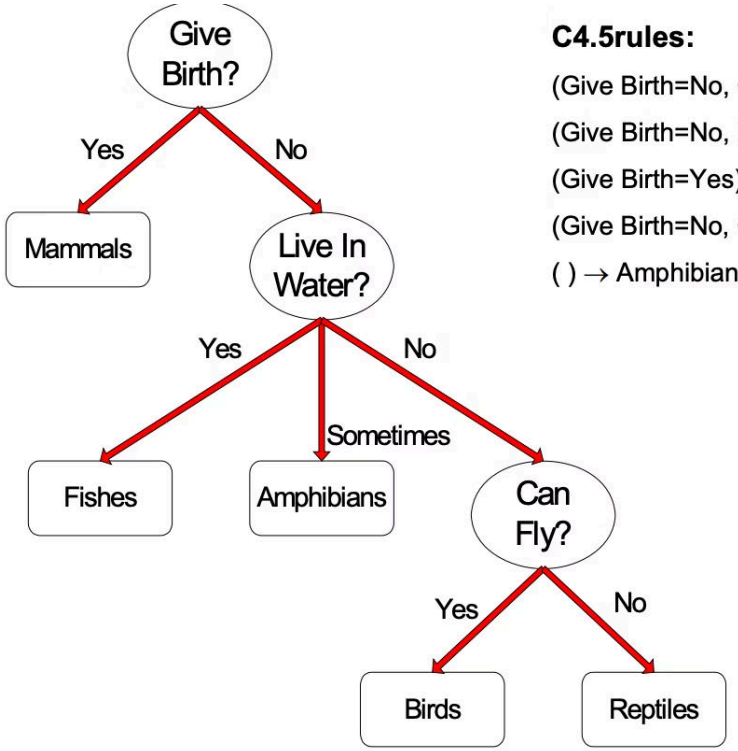
Sequential Covering

1. Find best rule that covers the current set of positive examples
2. Eliminate both positive and negative samples covered by the rule from the dataset
3. Compute description length after adding rule

Stopping Criterion

Stop when new description length is **d** bits longer than smallest obtained

Balances accuracy with model complexity



C4.5rules:

(Give Birth=No, Can Fly=Yes) → Birds

(Give Birth=No, Live in Water=Yes) → Fishes

(Give Birth=Yes) → Mammals

(Give Birth=No, Can Fly=No, Live in Water=No) → Reptiles

( ) → Amphibians

RIPPER:

(Live in Water=Yes) → Fishes

(Have Legs=No) → Reptiles

(Give Birth=No, Can Fly=No, Live In Water=No) → Reptiles

(Can Fly=Yes,Give Birth=No) → Birds

() → Mammals

C4.5rules vs RIPPER

C4.5rules: Tree-based, globally partitions into homogeneous regions

RIPPER: Sequential covering, biased toward smaller classes generated first

Different optimization strategies lead to different rule structures

Rule-Based Methods: Advantages

Highly Expressive

As powerful as decision trees

Easy to Interpret

Clear IF-THEN logic

Simple Generation

Straightforward algorithms

Rapid Classification

Fast prediction on new instances

Handles Imperfect Data

Missing values and numeric attributes

Comparable Performance

Similar accuracy to decision trees