CS-470: Machine Learning

Week 7 - Decision Trees

Instructor: Dr. Sajjad Hussain

Department of Electrical and Computer Engineering
SEECS, NUST

October 23rd, 2025

What are Decision Trees?

- **Tree-based models** for classification and regression
- **Mimic human decision-making** process
- **Interpretable** and **transparent** models
- Work by recursively partitioning feature space

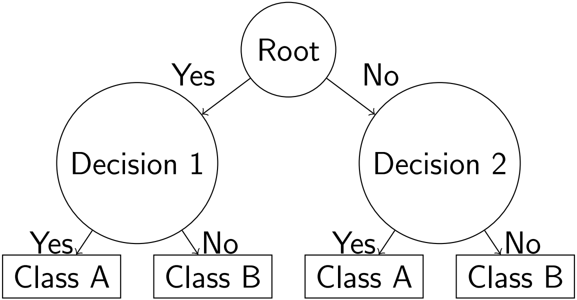


Figure: Structure of a Decision Tree

Why Decision Trees?

Advantages:

- Easy to understand and interpret
- Handle both numerical and categorical data
- Require little data preprocessing
- Can model non-linear relationships
- White box model

Applications:

- Medical diagnosis
- Credit risk analysis
- Customer segmentation
- Quality control
- Game playing (chess)

Key Insight

"Decision trees break down complex decisions into simple, interpretable rules"

Classification Tree Example

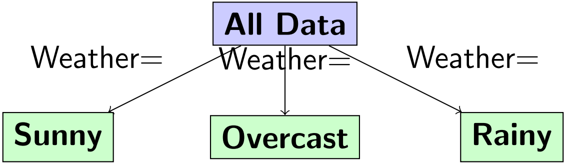
Weather Prediction Problem

Predict if outdoor activity will be enjoyable based on weather conditions

Outlook	**Temperature**	**Enjoy?**
Sunny	Hot	No
Sunny	Hot	No
Overcast	Hot	Yes
Rainy	Mild	Yes
Rainy	Cool	Yes
Rainy	Cool	No
Overcast	Cool	Yes
Sunny	Mild	Yes
Sunny	Cool	Yes
Rainy	Mild	Yes

Building the Tree: Key Questions

- **Which feature to split on first?**
- **What value to split at?**
- **When to stop splitting?**



Goal

Choose splits that create the **purest** child nodes

Measuring Impurity: Entropy

Entropy: Measure of disorder/uncertainty

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

- p_i: proportion of class i in node
- **Minimum (0):** Pure node (all same class)
- **Maximum ($\log_2 c$):** Equal distribution

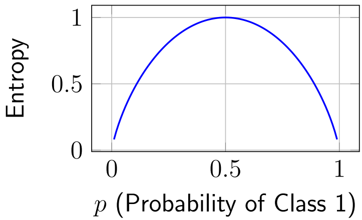


Figure: Binary Entropy Function

Example

Entropy Calculation Node with [3 Yes, 1 No]: $p_{yes} = 0.75$, $p_{no} = 0.25$, $H = -0.75\log_2(0.75) - 0.25\log_2(0.25) = 0.811$

Information Gain

Information Gain: Reduction in entropy after split

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

Before Split

Root: [7 Yes, 3 No]
$H_{before} = 0.881$

After Split

Weather=Sunny: [2Y, 2N],
$H = 1.0$
Weather=Overcast: [2Y, 0N],
$H = 0$
Weather=Rainy: [3Y, 1N],
$H = 0.811$
$H_{after} = \frac{4}{10}(1.0) + \frac{2}{10}(0.0) + \frac{4}{10}(0.8113) = 0.724$
$IG = 0.881 - 0.724 = 0.156$

GINI Impurity

GINI Impurity: Probability of misclassification

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

- **Range:** 0 (pure) to 0.5 (max impurity for binary)
- **Faster** to compute than entropy
- **Default** in many implementations

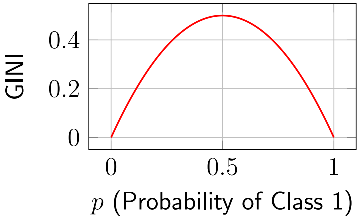


Figure: GINI Impurity

Example

GINI Calculation Node with [3 Yes, 1 No]: $p_{yes} = 0.75$, $p_{no} = 0.25$

$$GINI = 1 - (0.75^2 + 0.25^2) = 0.375$$

Comparison: Entropy vs GINI

Property	Entropy	GINI
Range	$[0, \log_2 c]$	$[0, 1 - \frac{1}{c}]$
Computation	Slower (log)	Faster
Sensitivity	More sensitive	Less sensitive
Popularity	ID3, C4.5	CART, scikit-learn
Results	Similar trees	Similar trees

Practical Choice

- GINI: Default choice for efficiency
- Entropy: When theoretical purity matters
- Both usually produce similar trees

CART Algorithm Overview

CART: Classification And Regression Trees

- **Binary splits only** (each node has 2 children)
- Works for both **classification** and **regression**

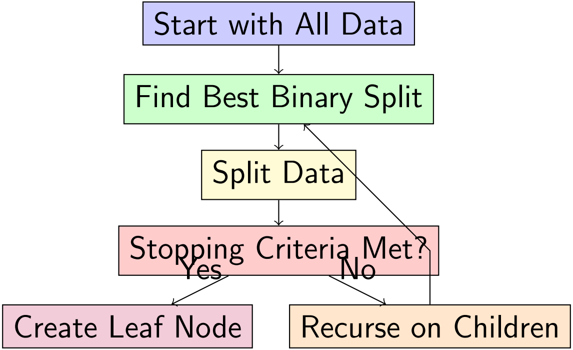


Figure: CART Algorithm Flow

Stopping Criteria

When to stop splitting:

- Node is pure (GINI $= 0$)
- Maximum depth reached
- Minimum samples per node
- No significant improvement
- All features identical

Preventing Overfitting

- **Pre-pruning:** Stop early
- **Post-pruning:** Grow full tree, then prune
- **Cross-validation** to find optimal parameters

Scikit-Learn Parameters

- `max_depth`
- `min_samples_split`
- `min_samples_leaf`
- `min_impurity_decrease`

Regression Trees

Key Differences from Classification

- **Predicts continuous values** instead of classes
- Uses **variance reduction** instead of GINI/entropy
- Leaf nodes predict **mean** of training samples

Splitting Criterion: Minimize Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

Variance Reduction:

$$\Delta_{var} = MSE_{parent} - \left(\frac{n_{left}}{n}MSE_{left} + \frac{n_{right}}{n}MSE_{right}\right)$$

Choose Split

Maximize variance reduction $\rightarrow$ Minimize weighted MSE

Regression Tree Example

Size (sqft)	Price ($)
1000	300,000
1200	320,000
1500	400,000
1800	420,000
2000	500,000
2200	520,000

Table: House Price Data

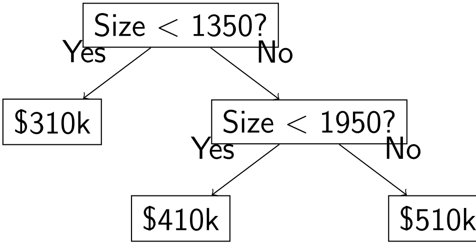


Figure: Regression Tree

- Size $<$ 1350: Predict $\$310,000$
- 1350 $<$ Size $<$ 1950: Predict $\$410,000$
- Size $>$ 1950: Predict $\$510,000$

Sensitivity to Data Orientation

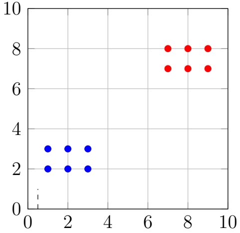


Figure: Easy Vertical Split

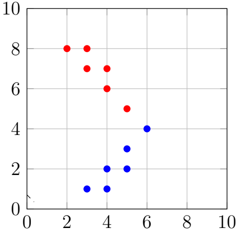


Figure: Hard Diagonal Split

Problem

Standard decision trees prefer to create **axis-parallel** splits

Instability: Small Data Changes

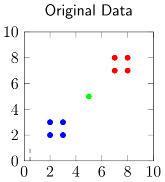


Figure: Tree 1: Split at x=4.5

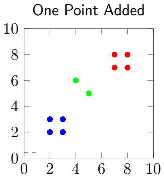


Figure: Tree 2: Split at y=4.5

High Variance

Small changes in training data can cause **completely different trees**

Other Limitations

Overfitting Tendency:

- Can create overly complex trees
- Memorize noise in training data
- Need careful pruning

Feature Selection Bias:

- Prefer features with more levels
- Can miss important interactions

Poor Extrapolation:

- Piecewise constant predictions
- Cannot extrapolate beyond training range
- Poor for trend prediction

Non-smooth Boundaries:

- Create hard decision boundaries
- Not suitable for probabilistic outputs

Solutions

- Ensemble methods (Random Forests, Gradient Boosting)

Summary

Strengths:

- Highly interpretable
- Handle mixed data types
- No feature scaling needed
- Non-linear relationships
- Feature importance

Weaknesses:

- Instability (high variance)
- Axis-parallel splits only
- Overfitting tendency
- Poor extrapolation

When to Use Decision Trees

- Need interpretable model
- Mixed data types
- Non-linear relationships
- As base learners for ensembles

Best Practices

Preprocessing:

- Handle missing values
- Encode categorical variables
- No need for scaling

Parameter Tuning:

- `max_depth`
- `min_samples_split`
- `min_samples_leaf`
- `min_impurity_decrease`

Validation:

- Use cross-validation
- Monitor train vs test performance
- Prune to avoid overfitting

Production:

- Use ensembles for better performance
- Export decision rules
- Monitor feature importance

Remember

"Decision trees are the building blocks for more powerful ensemble methods like Random Forests and Gradient Boosting Machines"