Chapter 3: Decision Tree

Latreche Sara

Contents

1 Introduction

Decision Trees are supervised machine learning models used for both classification and regression tasks.They closely mimic how we humans naturally make decisions, often without even realizing it. Whether we're deciding what to eat, which route to take to work, or how to respond in a conversation, we mentally break problems down into if-then scenarios. For example, "If I'm hungry and there's pizza in the fridge, then I'll eat that. Otherwise, I'll cook something." This step-by-step logic, branching at each decision point, is exactly how a decision tree works.

Because of this, decision trees are not only powerful but also intuitive. They give us a structured way to approach problems — and their visual tree-like form helps us understand and explain our reasoning.

Types of Decision Trees: Classification vs Regression

Decision trees can be used for two main types of prediction tasks: **classification** and **regression**.

1. Classification Decision Tree

One of the most intuitive use cases for a classification decision tree is helping make a categorical decision — such as whether or not to learn a new programming language.

- Do I need it for work or a specific project?
- Am I interested in the domain it's used in (web development, data science, etc.)?
- Is it beginner-friendly?

Each of these questions can be seen as a branch in a decision tree. Based on the answers, the tree leads to a final recommendation like `Learn Python`, `Learn JavaScript`, or `Don't learn a new language now`. These outputs are distinct categories, which makes this a **classification problem**.

2. Regression Decision Tree

In contrast, regression decision trees are used when the prediction is a **continuous numeric value**.

For example, consider a decision-making process to recommend a house size (like number of rooms) based on criteria such as:

- Number of family members
- Salary
- Marital status

The decision tree evaluates these conditions and gives a final output like `28BHK`, `4BHK`, or `3BHK` — which in this context can be interpreted as a form of numeric quantity prediction (especially if generalized to house price, size in square meters, etc.).

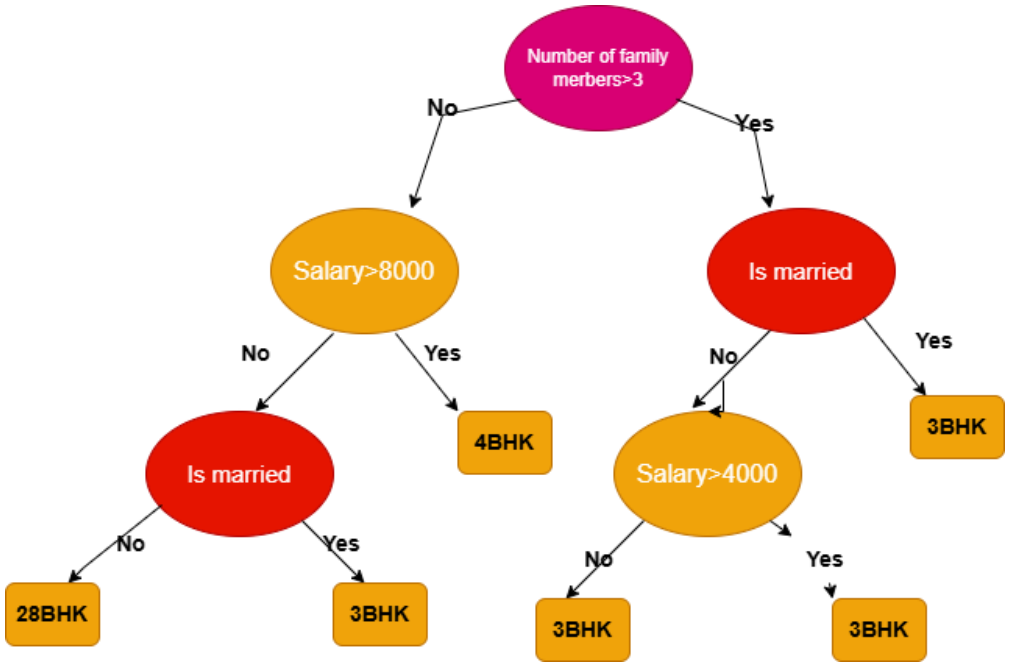


Figure 1: A regression decision tree predicting the number of rooms (e.g., 2BHK, 3BHK, 4BHK) based on family size, marital status, and salary.

Key Difference Between Classification and Regression Trees

- **Classification trees** predict discrete labels (e.g., `Yes/No`, `Apple/Cherry/Oak`, etc.).
- **Regression trees** predict continuous values (e.g., `House price = $120,000`, `Score = 8.5`, etc.).

Both types of trees use the same tree structure and splitting logic. The only difference is the type of value predicted at the **leaf nodes**.

2 Structure of a Decision Tree

A Decision Tree consists of:

- **Root Node**: The topmost node that starts the tree.
- **Internal Nodes**: Nodes that split the data.
- **Leaf Nodes**: Terminal nodes that contain the output class or value.

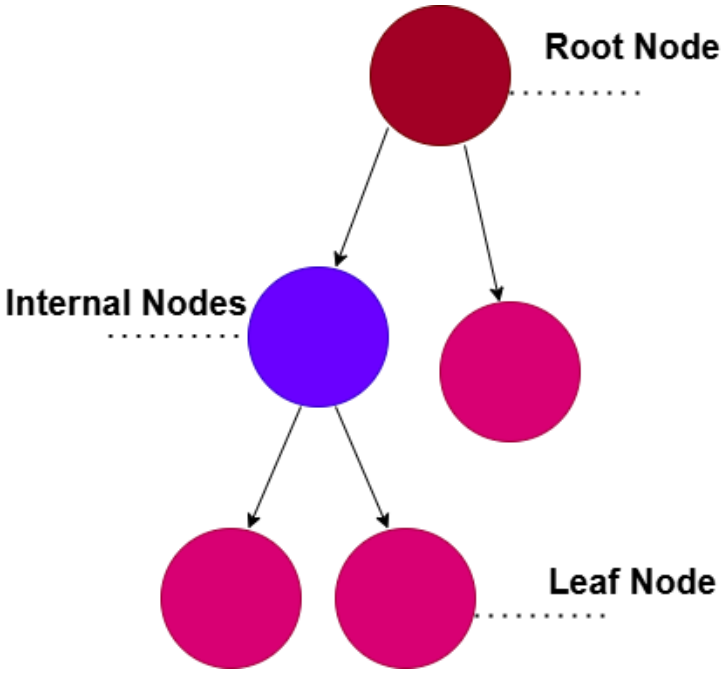


Figure 2: Basic structure of a decision tree.

3 Building a Decision Tree

Let's pretend you are a farmer with a new plot of land. Given only the Diameter and Height of a tree trunk, we must determine if it's an Apple, Cherry, or Oak tree. To do this, we'll use a Decision Tree.

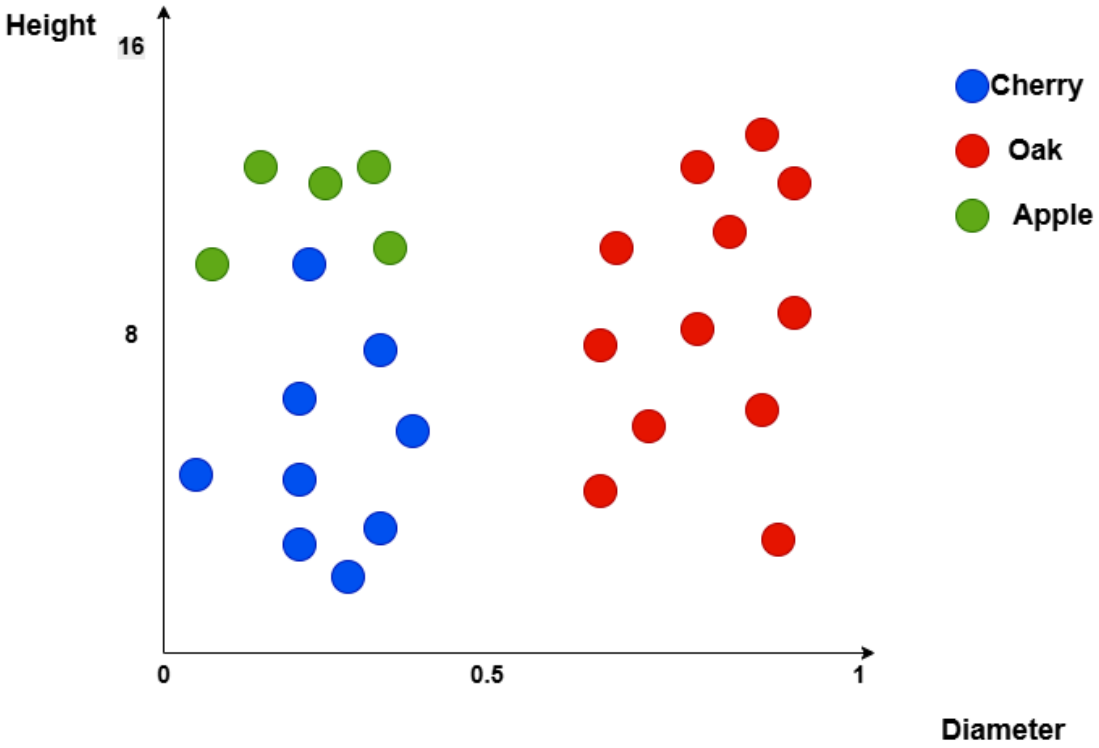


Figure 3: Scatter plot of trees by Diameter and Height. Apples (green), Cherries (blue), Oaks (red).

Start Splitting

Looking at the data, almost every tree with a Diameter greater than or equal to 0.45 meters is an Oak tree! This is a strong clue.

So, our first decision node (called the **root node**) checks:

$$\textit{Is the Diameter} \geq 0.5?$$

If yes, we classify it as an Oak and stop. If not, we continue to look further. This splits the plot vertically at Diameter = 0.5 (see Figure 3).

Split Some More

Next, we look at the remaining trees (Diameter < 0.5). We notice that those with Height 8 meters are mostly Cherry trees. Our next decision node asks Is Height ¿ 8 ?: If yes, label as Cherry; if no, label as apple.

Almost Done

Finally, a further split by height separates the last few points. Our decision tree now looks like this:

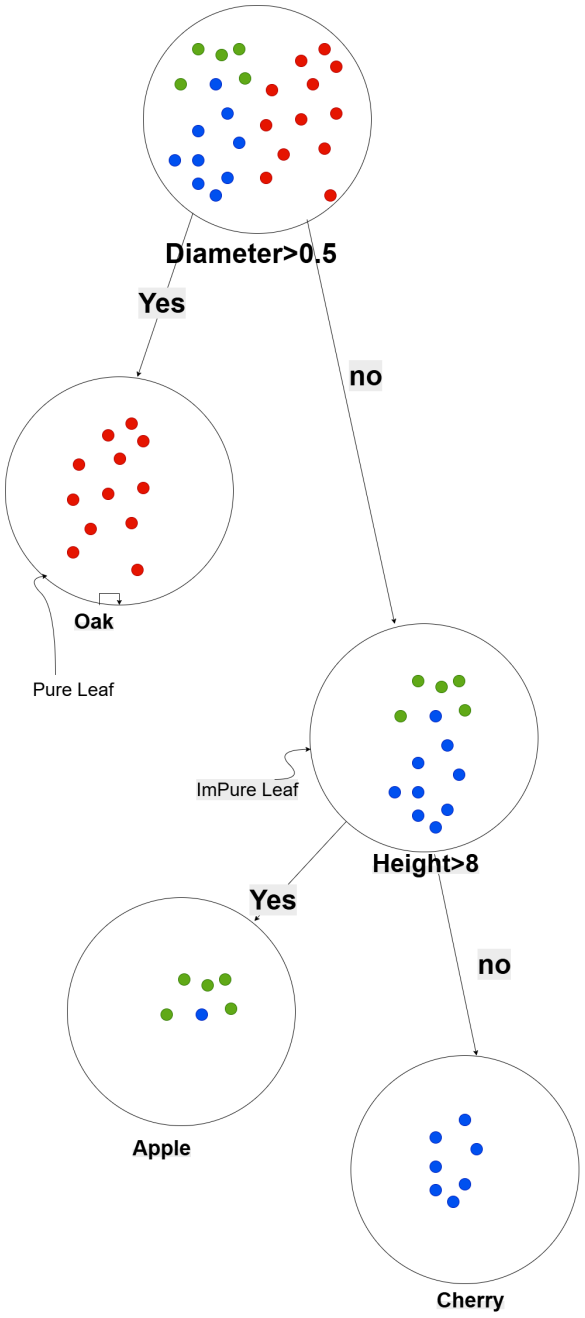


Figure 4: Constructed Decision Tree for classifying trees by Diameter and Height.

While this tree is quite accurate, some points might still be misclassified. Should we continue splitting and risk overfitting? That's the art of decision trees.

Your Turn: How to choose the best split?

Given this data, how would you decide where to split next? What attribute and what threshold would you pick? Think about what would best separate the classes.

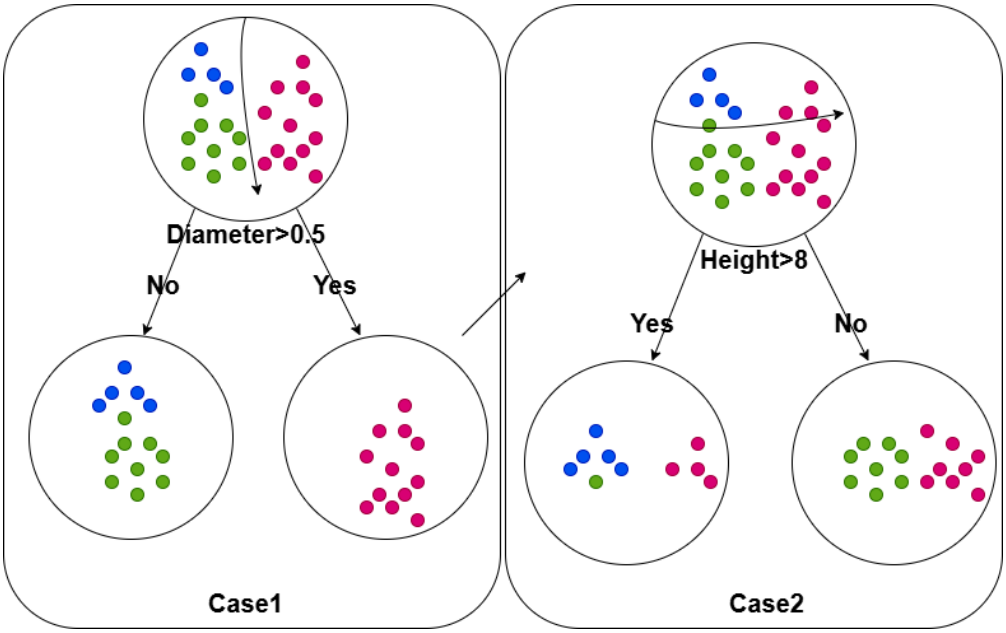


Figure 5: Two possible cases for splitting the mixed region: one based on Diameter, one based on Height.

4 Mathematics Behind Decision Trees: Making the Best Splits

To answer the previous question rigorously, we need to quantify how "good" a split is. This is where tree important concepts come in:

- **Entropy**
- **Gini Impurity**
- **Information Gain**

Let's dive in. —

What is Entropy? From Phone Lines to Decision Trees

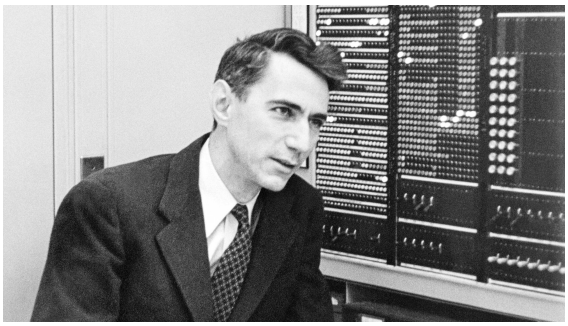


Figure 6: Claude Shannon (1916–2001), father of information theory

The Problem Shannon Was Solving

In the 1940s, Claude Shannon was working at Bell Labs, where he focused on improving the efficiency and reliability of telephone communications. At the time, phone lines were analog and noisy — signals were often distorted, and engineers needed to find a way to **quantify and manage this noise**.

Shannon asked:

> *"How much information can we transmit over a noisy communication channel, and what is the theoretical limit?"*

To answer this, he had to define what **"information"** actually means — in a mathematical sense. This led to the development of a new field: **information theory**. In 1948, Shannon published his groundbreaking paper: *"A Mathematical Theory of Communication"*, where he introduced the idea of **entropy** as a measure of uncertainty in a message source.

Von Neumann's Advice: Why "Entropy"?

Initially, Shannon considered calling his function "uncertainty." But when he discussed it with John von Neumann, he received this now-famous suggestion:

> *"You should call it entropy, for two reasons. In the first place, your uncertainty function has been used in statistical mechanics under that name, so it already has a name. In the second place, and more important, nobody knows what entropy really is, so in a debate you will always have the advantage."*

Entropy in Information Theory

Shannon defined entropy as the **average information content** of a random variable X, whose outcomes occur with probabilities $p(x_1), p(x_2), ..., p(x_n)$:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

Entropy reaches its maximum when all outcomes are equally likely, reflecting maximum uncertainty.

Entropy Beyond Communication: Thermodynamics

Although Shannon developed entropy for communications, it turned out that the formula mirrors one already used in **statistical mechanics**. In physics, entropy measures how many microscopic configurations correspond to a macroscopic state. The Boltzmann formula is:

$$S = k \ln W$$

where W is the number of microstates, and k is Boltzmann's constant.

One Concept

Shannon entropy measures **uncertainty in messages**. Boltzmann entropy measures **disorder in physical systems**. Same math. Different domains. Deep connections.

Relevance to Machine Learning

In decision trees, we use entropy to decide how to split data at each node. A node with only one class has zero entropy (pure). A node with an even class distribution has high entropy (uncertain). We aim to choose splits that **reduce entropy the most**, known as **information gain**.

- **Entropy** — measures the impurity or uncertainty of a dataset.
- **Information Gain** — measures the reduction in entropy after a split.

Why log base 2? Because entropy is measured in bits — the fundamental unit of information.

What is Gini Impurity? Simplicity Meets Performance

Motivation and Definition

While entropy comes from information theory, **Gini impurity** is a simpler, yet powerful, metric for measuring the **purity of a node** in a decision tree.

It is defined as:

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

Where:

- p_i is the probability of class i in the node t,
- C is the number of classes.

Like entropy:

- Gini impurity is 0 when the node is pure (all samples belong to one class).
- It reaches its maximum when the classes are perfectly mixed.

Intuition Behind Gini Impurity

Gini impurity measures the probability of *incorrectly classifying* a randomly chosen element from the set, if you randomly label it according to the distribution of labels in the node.

Gini in Plain Words

If you pick a sample at random and randomly assign a class based on class frequencies in the node, Gini tells you the chance you'll be wrong.

Gini vs Entropy

Property	Entropy	Gini Impurity
Origin	Information Theory	Probability Theory
Formula	$-\sum p_i \log_2 p_i$	$1 - \sum p_i^2$
Range (Binary)	[0, 1]	[0, 0.5]
Computational Cost	Higher (due to log)	Lower
Tends to Favor	Balanced splits	Larger class purity

Table 1: Comparison between Entropy and Gini Impurity

Which One Should You Use?

Most decision tree algorithms (like CART) use Gini impurity by default because it is:

- Faster to compute,
- Yields similar trees to those built with entropy,
- Empirically effective for many problems.

Still, entropy remains useful when you need a more fine-grained measure of impurity, especially when class distributions are close.

Quantifying Uncertainty: Coin Flip Example

Let's consider two sequences of coin flips where the outcomes are either 0 (tails) or 1 (heads):

- **Sequence 1:** 2 ones (1s) and 16 zeros (0s)
- **Sequence 2:** 10 ones (1s) and 8 zeros (0s)

Entropy Calculation:
For a sequence with class proportions p_0 and p_1, entropy is given by:

$$H(X) = -p_0 \log_2 p_0 - p_1 \log_2 p_1$$

Sequence 1:

$$p_0 = \frac{16}{18}, \quad p_1 = \frac{2}{18}$$

$$H_1 = -\frac{16}{18} \log_2 \frac{16}{18} - \frac{2}{18} \log_2 \frac{2}{18} \approx 0.503$$

Sequence 2:

$$p_0 = \frac{8}{18}, \quad p_1 = \frac{10}{18}$$

$$H_2 = -\frac{8}{18} \log_2 \frac{8}{18} - \frac{10}{18} \log_2 \frac{10}{18} \approx 0.992$$

Gini Impurity:

$$G = 1 - p_0^2 - p_1^2$$

Sequence 1:

$$G_1 = 1 - \left(\frac{16}{18}\right)^2 - \left(\frac{2}{18}\right)^2 \approx 0.198$$

Sequence 2:

$$G_2 = 1 - \left(\frac{8}{18}\right)^2 - \left(\frac{10}{18}\right)^2 \approx 0.494$$

Questions to Reflect On:

- How **surprising** is it to see a new value in each sequence?
- Which sequence conveys **more information** when a new result appears?
- Why does Sequence 2 have higher entropy and Gini impurity than Sequence 1?

Figure 7: Comparison of Entropy and Gini Impurity as functions of the probability of Heads in a coin flip (binary classifier).

Insight: Entropy and Information Gain

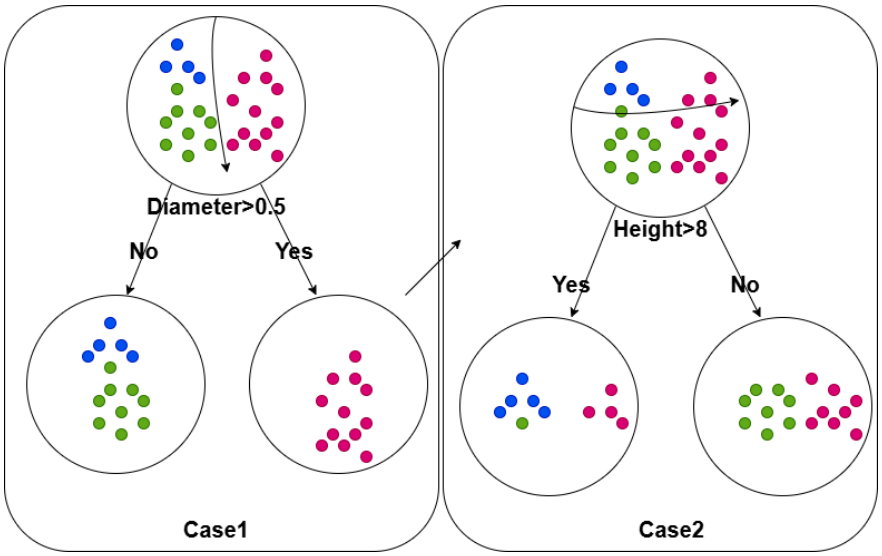
Entropy is a measure of **disorder or uncertainty**. As entropy increases, the system becomes more disordered — harder to predict.
But that also means the **information gain from a split decreases**, because it's harder to find attributes that make clean separations.
High entropy = high uncertainty = lower information gain.

Entropy, Gini Impurity, and Information Gain: Evaluating Two Splits

Now that we have become familiar with the rigorous measures of **entropy**, **Gini impurity**, and **information gain**, we can use these tools to objectively answer the fundamental question in decision tree learning: *Which split best separates the data?*

By quantifying the uncertainty and the effectiveness of a split in reducing this uncertainty, these measures guide us in selecting the attribute and split that maximizes information gain — thus yielding the most informative and discriminative partitions of the dataset.

Let us revisit the example dataset and splits from the previous subsection to verify which split is indeed the best choice.



Original dataset: 12 Oak, 5 Apple, 10 Cherry (27 samples total)

Original Entropy and Gini Impurity:

The class proportions are:

$$p_{\text{Oak}} = \frac{12}{27} \approx 0.444, \quad p_{\text{Apple}} = \frac{5}{27} \approx 0.185, \quad p_{\text{Cherry}} = \frac{10}{27} \approx 0.370$$

Entropy is calculated as:

$$H_{\text{original}} = -\sum p_i \log_2 p_i \approx 1.50$$

Gini impurity is:

$$G_{\text{original}} = 1 - \sum p_i^2 \approx 0.632$$

Split 1:

Left group: 12 Oak, 5 Apple $(n_L = 17)$, Right group: 10 Cherry $(n_R = 10)$

- Left group proportions: $p^L_{\text{Oak}} \approx 0.706$, $p^L_{\text{Apple}} \approx 0.294$, $p^L_{\text{Cherry}} = 0$ - Left group entropy: $H_L \approx 0.873$ - Left group Gini impurity: $G_L \approx 0.415$ - Right group entropy and Gini impurity (pure class): $H_R = 0, G_R = 0$

Weighted entropy after split:

$$H_{\text{split 1}} = \frac{17}{27} \times 0.873 + \frac{10}{27} \times 0 = 0.549$$

Weighted Gini impurity after split:

$$G_{\text{split 1}} = \frac{17}{27} \times 0.415 + \frac{10}{27} \times 0 = 0.261$$

Information gain:

$$IG_1 = H_{\text{original}} - H_{\text{split 1}} = 1.50 - 0.549 = 0.951$$

Split 2:

Left group: 9 Oak, 9 Cherry $(n_L = 18)$, Right group: 3 Oak, 1 Apple, 5 Cherry $(n_R = 9)$

- Left group proportions: $p^L_{\text{Oak}} = 0.5$, $p^L_{\text{Apple}} = 0$, $p^L_{\text{Cherry}} = 0.5$ - Left group entropy: $H_L = 1.0$ - Left group Gini impurity: $G_L = 0.5$

- Right group proportions: $p^R_{\text{Oak}} \approx 0.333$, $p^R_{\text{Apple}} \approx 0.111$, $p^R_{\text{Cherry}} \approx 0.556$ - Right group entropy: $H_R \approx 1.353$ - Right group Gini impurity: $G_R \approx 0.568$

Weighted entropy after split:

$$H_{\text{split 2}} = \frac{18}{27} \times 1.0 + \frac{9}{27} \times 1.353 = 1.117$$

Weighted Gini impurity after split:

$$G_{\text{split 2}} = \frac{18}{27} \times 0.5 + \frac{9}{27} \times 0.568 = 0.522$$

Information gain:

$$IG_2 = H_{\text{original}} - H_{\text{split 2}} = 1.50 - 1.117 = 0.383$$

Conclusion:

Between the two splits, **Split 1** has a higher information gain (0.951) compared to Split 2 (0.383). This means Split 1 reduces uncertainty more effectively and is the better choice for splitting the dataset.

5 Iterative Dichotomiser 3 (ID3) Algorithm

Now that we have explored the key concepts such as entropy and information gain, we can present the complete pseudocode for the Iterative Dichotomiser 3 (ID3) algorithm, which builds a decision tree by recursively selecting the best attribute to split the data.

5.1 Pseudocode for the ID3 Algorithm

The ID3 algorithm follows a recursive approach to construct the decision tree, using information gain to decide the best feature at each step:

Algorithm 1 ID3(R: input attributes, C: class attribute, S: training set, ParentS: parent training set)

if S is empty **then**
    return `LeafNode` with class = `MajorityClass(ParentS)`
else if all examples in S have the same class c **then**
    return `LeafNode` with class = c
else if R is empty **then**
    return `LeafNode` with class = `MajorityClass(`S`)`
else
    $D \leftarrow \arg\max_{A \in R} \text{InformationGain}(A, S)$
    Create decision node with attribute D
    for each value d_j of attribute D **do**
        $S_j \leftarrow$ subset of S where $D = d_j$
        Add subtree: ID3($R \setminus \{D\}$, C, S_j, S)
    end for
    return decision node
end if

5.2 How ID3 Works

- If the training set S is empty, ID3 returns the majority class of the parent set to handle missing attribute combinations.
- If all examples in S belong to the same class, the algorithm returns a leaf node with that class.
- If there are no remaining attributes to split on, it returns the majority class in S.
- Otherwise, the attribute D with the highest information gain is chosen to split S.
- For each possible value of D, the algorithm recursively builds subtrees with the subset of data where D equals that value.

This recursive process continues until all training examples are perfectly classified or no further attributes remain, producing a decision tree that models the data based on maximum information gain splits.

5.3 ID3 Algorithm Example: Weather and Temperature

We have a dataset with three columns — **Weather**, **Temperature**, and **PlayOutside** (target), with the following 6 records:

Weather	Temperature	PlayOutside
sunny	Hot	no
sunny	Mild	yes
rainy	Cool	no
rainy	Mild	no
cloudy	Cool	yes
cloudy	Mild	yes

Our goal is to build a decision tree to predict **PlayOutside** based on the features **Weather** and **Temperature**.

Step 1: Entropy of Parent Node

The target distribution is:

$$\text{Yes} = 3, \quad \text{No} = 3$$

The entropy is computed by:

$$H(X) = -p_{\text{yes}} \log_2 p_{\text{yes}} - p_{\text{no}} \log_2 p_{\text{no}}$$

with

$$p_{\text{yes}} = \frac{3}{6} = 0.5, \quad p_{\text{no}} = \frac{3}{6} = 0.5$$

Calculating entropy:

$$H_{\text{parent}} = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1.0$$

Step 2: Calculate Entropy and Information Gain for Each Feature

Feature: Weather

Weather	Count	Yes	No	Entropy
sunny	2	1	1	$-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1.0$
rainy	2	0	2	0 (pure no)
cloudy	2	2	0	0 (pure yes)

Weighted entropy for Weather:

$$H_{\text{Weather}} = \frac{2}{6} \times 1.0 + \frac{2}{6} \times 0 + \frac{2}{6} \times 0 = 0.3333$$

Information Gain for Weather:

$$IG_{\text{Weather}} = H_{\text{parent}} - H_{\text{Weather}} = 1.0 - 0.3333 = 0.6667$$

Feature: Temperature

Temperature	Count	Yes	No	Entropy
Hot	1	0	1	0 (pure no)
Mild	3	2	1	$-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} \approx 0.9183$
Cool	2	1	1	1.0

Weighted entropy for Temperature:

$$H_{\text{Temperature}} = \frac{1}{6} \times 0 + \frac{3}{6} \times 0.9183 + \frac{2}{6} \times 1.0 = 0 + 0.45915 + 0.3333 = 0.7925$$

Information Gain for Temperature:

$$IG_{\text{Temperature}} = 1.0 - 0.7925 = 0.2075$$

Step 3: Choose the Best Feature to Split On

Since

$$IG_{\text{Weather}} = 0.6667 > IG_{\text{Temperature}} = 0.2075,$$

we split on **Weather** first.

Step 4: Split Dataset by Weather

Branch: Weather = sunny

Temperature	PlayOutside
Hot	no
Mild	yes

Entropy of this subset:

$$H = 1.0 \quad \text{(since 1 yes, 1 no)}$$

Branch: Weather = rainy

Temperature	PlayOutside
Cool	no
Mild	no

Entropy of this subset:

$$H = 0 \quad \text{(pure no)}$$

Branch: Weather = cloudy

Temperature	PlayOutside
Cool	yes
Mild	yes

Entropy of this subset:

$$H = 0 \quad \text{(pure yes)}$$

Step 5: Further Split Sunny Branch on Temperature

Since entropy in the sunny branch is 1.0, we split it on Temperature:

Temperature	PlayOutside
Hot	no
Mild	yes

Each leaf is pure, so entropy is 0 for both.

—

Summary:

- First split on **Weather**.
- If Weather = rainy, predict **no**.
- If Weather = cloudy, predict **yes**.
- If Weather = sunny, split on **Temperature**.
- If Temperature = Hot, predict **no**.
- If Temperature = Mild, predict **yes**.

This completes the decision tree building using the ID3 algorithm for the given dataset.

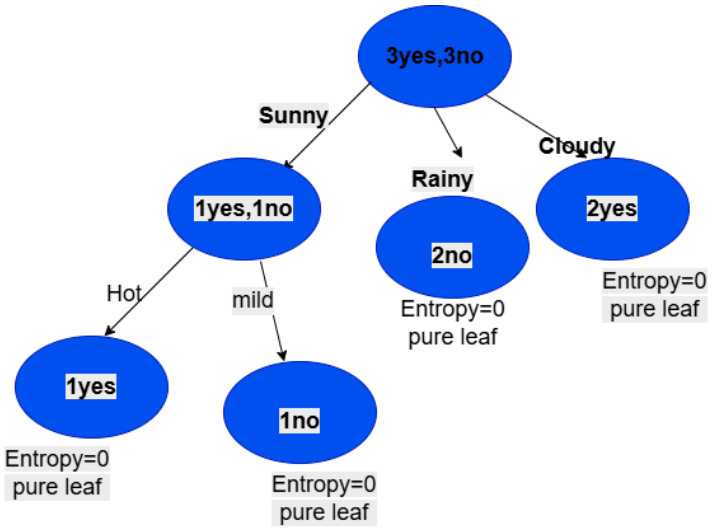


Figure 8: Decision tree built from the weather-temperature example

6 Regression Trees: Predicting Continuous Values

While decision trees are commonly used for classification tasks (predicting discrete labels), a variant known as **Regression Trees** is designed to predict *continuous* values.

6.1 How Regression Trees Work

A regression tree follows the same recursive splitting strategy as classification trees, but instead of maximizing information gain or minimizing Gini impurity, it aims to minimize the **sum of squared errors** (SSE) for predictions at each split. Formally, the objective is to minimize:

$$\sum_{n \in \text{node}} \left(f_{\text{node}}(x_n) - y_n\right)^2$$

Here, $f_{\text{node}}(x_n)$ typically returns the mean of the output values y_n of the training samples that fall into the same leaf node as x_n.

6.2 Key Characteristics and Variants

- **Splitting Criterion**: Instead of entropy or Gini impurity, the split is chosen to minimize prediction error (usually SSE).
- **Leaf Prediction**: Each leaf node predicts the average (mean) target value of the training examples it contains.

- **Stopping Criteria**: To avoid overfitting, training may stop early:
  - If the number of samples in a node is less than a threshold $N_{\min}$
  - If the maximum depth of the tree is reached
- **Advanced Splits**: While typical regression trees split on a single feature at a time, some variants can split on linear combinations of features—especially in high-dimensional continuous input spaces.
- **Multi-output Support**: Regression trees can be extended to predict multiple continuous values or even handle multi-class classification tasks.

6.3 Comparison with Classification Trees

Regression trees differ mainly in the nature of the output variable and the splitting criterion:

Tree Type	**Output**	**Splitting Criterion**
Classification Tree	Discrete labels	Information Gain / Gini Impurity
Regression Tree	Continuous values	Sum of Squared Errors (SSE)

Regression trees are a powerful tool for numerical prediction and are often used in ensemble methods like *Random Forest Regressors* and *Gradient Boosted Trees*.

7 The Problem of Perturbations

Despite their many strengths—such as interpretability, fast training, and robustness to outliers—Decision Trees suffer from a major drawback: **instability**. Specifically, they are highly sensitive to small changes in the training data. Even a slight perturbation can cause the structure of a Decision Tree to change significantly.

Example of Instability

Imagine applying small Gaussian perturbations to just 5% of the training examples. If we train multiple Decision Trees on these slightly modified datasets, we may find that each tree is structurally very different. This is because:

- Decision Trees are greedy; they make splitting decisions based on local criteria.
- A minor change in the data distribution can shift the perceived best feature to split on.

- Once the first split changes, the rest of the tree can evolve in a completely different direction.

This makes individual Decision Trees poor candidates for high-stakes decision-making, where stability and reproducibility are crucial.

A Robust Solution: Random Forests

To overcome this instability, one effective solution is to use **Random Forests**. A Random Forest builds an ensemble of Decision Trees, each trained on a slightly different subset of the training data and with a random subset of the features at each split. The predictions are then averaged (for regression) or voted (for classification).

- This ensemble approach reduces variance.
- Randomness ensures that individual trees do not overfit to the same patterns.
- The overall model is much more stable and robust to perturbations.

In short, while individual Decision Trees are fragile learners, Random Forests combine their strengths while mitigating their weaknesses. This makes them one of the most popular and reliable models in machine learning.

8 Advantages and Limitations

Advantages

- Easy to interpret and visualize
- No need for feature scaling
- Handles both numerical and categorical data

Limitations

- Prone to overfitting
- Unstable with small data changes
- Greedy algorithm may miss optimal tree

9 Applications

- Fraud detection
- Medical diagnosis
- Loan approval systems

10 Conclusion

Decision Trees are a core part of machine learning due to their simplicity and power. By understanding their logic, structure, and math, you are equipped to use them effectively in your own data science projects.

References

- Claude E. Shannon, *A Mathematical Theory Of Communication*, Bell System Technical Journal, 1948.
- John Ross Quinlan, *Induction of Decision Trees*, Machine Learning, 1986.
- Luis Torgo, *A Study on End-Cut Preference in Least Squares Regression Trees*, University of Porto, 2001.
- Lidia Ceriani and Paolo Verne, *The Origins Of The Gini Index: Extracts From Variabilità e Mutabilità (Corrado Gini, 1912)*, The Journal of Economic Inequality, 2012.
- Alice Gao, *Lecture 7: Decision Trees*, November 2, 2021.
- *MLU: Machine Learning, Explained Visually*, `https://mlu-explained.github.io`