# Decision Trees

Shreeyesh Menon

## 1 Introduction

Decision trees are simple yet powerful models used for classification. They operate by recursively partitioning the feature space into regions that are increasingly homogeneous with respect to the target variable. The algorithm is quite intuitive: we come up with a sequence of yes-or-no questions and give an answer based on that. The trick is to come up with the most efficient sequence of such questions such that we are able to classify objects correctly. Decision trees are particularly valuable in situations where inference needs to be quick and computationally inexpensive.

**Intuition:** At each step, we ask a question about one of the input features. Based on the answer, we branch left or right (or along multiple branches in some variants). The goal is to ask questions that meaningfully split the data into "purer" subsets.

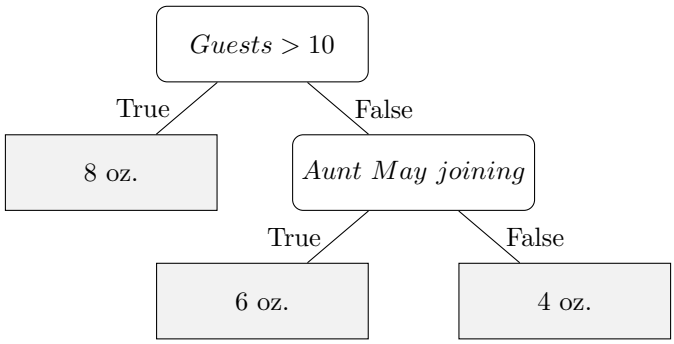


Figure 1: Illustration of a simple decision tree to decide how much chocolate syrup to buy for the dinner party

**Nodes, Branches, and Leaves**

- **Root Node**: The starting point; contains the entire dataset.
- **Internal Nodes**: Represent decisions based on feature values.
- **Leaves**: Terminal nodes that assign a class label.

Each split aims to maximize the separation of classes.

## 2 Impurity

To choose the best split, the algorithm evaluates possible partitions using impurity measures.

### 2.1 Gini Impurity

For classification tasks, Gini impurity is defined as

$$G = 1 - \sum_{k=1}^{K} p_k^2,$$

where $p_k$ is the proportion of class $k$ in the node. A lower Gini value means higher purity.

### 2.2 Entropy

Another common impurity measure is entropy:

$$H = -\sum_{k=1}^{K} p_k \log p_k.$$

Entropy is highest when classes are equally mixed.

## 3 Building the Tree

Decision trees grow through a process called *recursive binary splitting*. At each step, we:

1. Evaluate all possible feature thresholds.
2. Compute impurity for each split.
3. Choose the split that yields the largest reduction in impurity.

This continues until no further improvement is possible or until we reach a user-defined stopping criterion.

This is called a **greedy** algorithm, since we are making the best choice at each step without regard for the future. This approach is only *approximately* optimal.

## 4 Tree Complexity

Decision trees can easily overfit if allowed to grow without restriction. Several parameters control complexity.

### 4.1 Tree Depth

Depth refers to the maximum number of splits on any path from root to leaf. A deeper tree can model more complex patterns but risks overfitting.

### 4.2 Number of Leaves

The number of terminal nodes affects both interpretability and flexibility. Too many leaves indicate an overly complex model.

### 4.3 Minimum Samples for Split and Leaf

We may require a minimum number of samples to perform a split or to declare a leaf, preventing overly fine partitions.

## 5 Tree Pruning

Pruning reduces tree complexity after the tree has been fully grown.

### 5.1 Cost-Complexity Pruning

A common approach adds a penalty for tree size:

$$R_\alpha(T) = R(T) + \alpha|T|,$$

where $R(T)$ is the misclassification error, $|T|$ is the number of leaves, and $\alpha$ controls the trade-off.

$\alpha$ is usually selected based on cross-validation. Pruning finds a subtree that minimizes $R_\alpha(T)$, preventing overfitting.

### 5.2 Pre-Pruning vs Post-Pruning

- **Pre-pruning**: Stop tree growth early using constraints like max depth.
- **Post-pruning**: Grow the full tree and prune afterward.

## 6 In summary

Decision trees offer:

- Clarity in decision logic.
- No need for feature scaling.
- Handling of mixed feature types.

They serve as building blocks for advanced ensemble methods like Random Forests and Gradient Boosted Trees.