CHAPTER 4

DECISION TREE

4.1 Introduction

A Decision Tree is a supervised learning model used for both classification and regression tasks. It is tree-structured with:

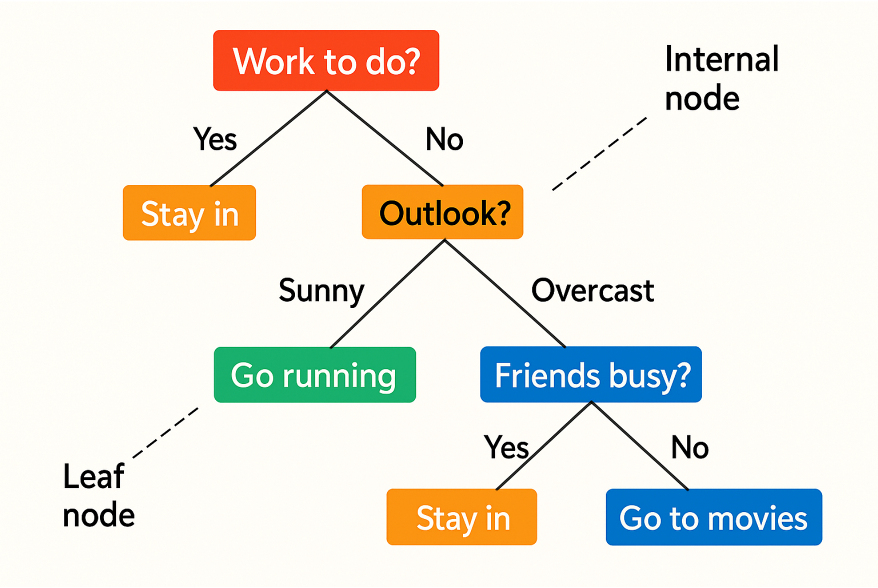


Figure 4.1: An Example of Decision Tree

- Internal nodes representing decisions or tests on features. Example: 4.1 Outlook, Friends Busy
- Branches representing the outcomes of those tests. Example: 4.1 Yes, No, Sunny, Overcast etc.
- Leaf nodes representing class labels (classification) or numerical predictions (regression). Example: 4.1 Stay in, Go running, Go to Movies.

The objective is to split the dataset in a way that reduces impurity or prediction error.

4.2 Entropy: A Measure of Uncertainty

Entropy, from Shannon's Information Theory, measures the impurity or disorder in a dataset D with k class labels, $C_1, C_2 \dots C_k$:

$$H(D) = -\sum_{i=1}^{k} P_i \log_2 P_i, \tag{4.1}$$

where P_i is the relative frequency (or probability) of class C_i.

This formula originates from Claude Shannon's foundational work on information theory. Entropy quantifies the expected amount of information (or surprise) from observing the outcome of a random variable, such as the class label of a data point.

If all classes are equally probable:

$$P_i = \frac{1}{k}$$

$$k = \frac{1}{P_i}$$

$$\log_2 k = -\log_2 P_i$$

Extending this idea to the general case where the classes $C_1, C_2, \dots, C_k$ have arbitrary probabilities $P_1, P_2, \dots, P_k$, we can measure the total entropy as:

$$H(D) = -\sum_{i=1}^{k} P_i \log_2 P_i$$

We are measuring information in units of **bits**, which is why we use base 2 in the logarithm.

Consider a binary classification problem with 2 class labels. To distinguish between these two labels, we need 1 bit. More generally, if we have k classes, we need $\log_2 k$ bits to represent them.

Other common units for measuring information include:

- **Nats**, which use the natural logarithm ($\log_e$ or ln), commonly used in physics and information theory when working with continuous distributions.
- **Hartleys**, which use base 10 logarithms ($\log_{10}$), sometimes used in communication systems.

In the context of classification, entropy reflects the unpredictability of the class label.

- High entropy (close to 1 bit for binary classification) means classes are equally mixed, making the outcome uncertain.
- Low entropy (0) means all samples belong to one class, hence fully predictable.

High entropy in the training data indicates a rich diversity of examples across different classes. This is good for training because it provides the model with sufficient information to learn meaningful decision boundaries.

However, when splitting data at a node, high entropy is undesirable—it means the split has not made the data more pure or homogeneous. The goal of each split in a decision tree is to reduce entropy (increase purity), so a good split is one that creates low-entropy subsets.

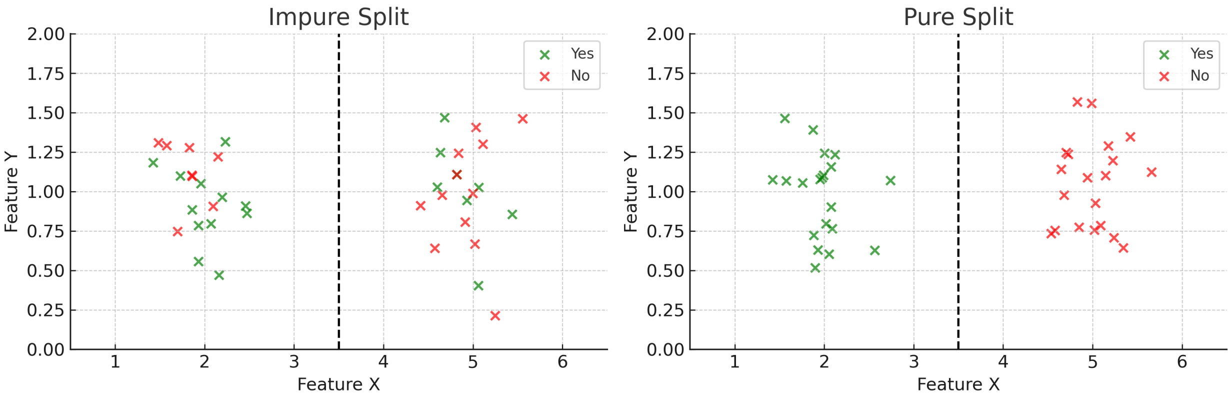


Figure 4.2: Visual comparison of an impure split (left) vs a pure split (right).

Example in fig: 4.2 we can see, in the impure split, both child nodes contain a mix of classes (Yes in green, No in red), indicating high entropy. In the pure split, the data is cleanly separated by class, resulting in low entropy and a more informative split.

- **High entropy in training data** ensures the model encounters all class types.
- **Low entropy in split subsets** ensures the decision tree is making clear distinctions that improve classification.

4.3 Conditional Entropy and Information Gain

Let's say a feature X can have m values (subsets) $X_1, X_2, ..., X_m$ after a split. We define the **Conditional Entropy** of the feature for the dataset as:

$$H(D|X) = \sum_{j=1}^{m} P(X = X_i) H(D|X = X_j) \tag{4.2}$$

$$H(D|X) = \sum_{j=1}^{m} \frac{\text{\#of instances where } X = X_j}{\text{Total instances in } X} H(D|X = X_j) \tag{4.3}$$

Here, $H(D \mid X = X_j)$ also written as $H(D \mid X_j)$ or $H(D_{X_j})$ is the **Specific Conditional Entropy** for the subset where the feature X has value X_j.

$$H(D \mid X_j) = -\sum_{i=1}^{k} P(C_i|X_j) \log_2 P(C_i|X_j). \tag{4.4}$$

Here, $P(C_i|X_j)$ is the probability of the i'th class, C_i in the subset where feature $X = X_j$.

Using all these, we can compute the **Information Gain (IG)** which measures reduction in entropy.

$$IG(D, X) = H(D) - H(D|X) \tag{4.5}$$

Features with higher IG are preferred for splitting.

4.4 Example: Toy Dataset

Instance	Weather	Play Tennis?
1	Sunny	Yes
2	Sunny	No
3	Rainy	Yes
4	Sunny	Yes
5	Rainy	No

Table 4.1: Example Dataset

Entropy of Entire Dataset

$$H(D) = -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right) \approx 0.971$$

Conditional Entropy given Weather

- Sunny: 3 instances (2 Yes, 1 No)

$$H(D_{Sunny}) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) \approx 0.918$$

- Rainy: 2 instances (1 Yes, 1 No)

$$H(D_{Rainy}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(D|\text{Weather}) = P(\text{sunny}) \cdot H(D_{\text{sunny}}) + P(\text{rainy})H(D_{\text{rainy}})$$

$$H(D|\text{Weather}) = \frac{3}{5} \cdot 0.918 + \frac{2}{5} \cdot 1 = 0.9508$$

Information Gain

$$IG(D, Weather) = 0.971 - 0.9508 = 0.0202$$

4.5 The ID3 Algorithm

Pseudocode for ID3 Decision Tree Construction

```
Input: Dataset $D$, Feature Set $F$
Output: Decision Tree T
Procedure ID3(D, F):
    1. Compute entropy $H(D)$
    2. For each feature $X \in F$:
        a. Compute Information Gain $IG(D, X)$
    3. Select feature $X^*$ with highest IG
    4. Split $D$ into subsets $D_1, D_2, \ldots, D_m$ based on the values $v_1, v_2, \ldots, v_m$ of $X^*$
    5. For each subset $D_j$ corresponding to $X^* = v_j$:
        a. If $D_j$ is pure (all same class) or $F$ is empty:
            - Return leaf node with majority class label in $D_j$
        b. Else: Recurse: ID3($D_j$, $F - \{X^*\}$)
```

4.6 Example: Full ID3 Walkthrough

Dataset: Play Tennis Example

Day	Outlook	Temp	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Table 4.2: Play Tennis Dataset

Step 1: Compute Entropy of the Entire Dataset

9 instances are labeled `Yes`, and 5 are labeled `No`.

$$\begin{aligned} H(D) &= -\left(\frac{9}{14}\log_2\frac{9}{14} + \frac{5}{14}\log_2\frac{5}{14}\right) \\ &= -\left(0.643 \cdot (-0.643) + 0.357 \cdot (-1.485)\right) \\ &\approx 0.940 \end{aligned}$$

Step 2: Compute Information Gain for Each Attribute

Compute Information Gain For Outlook

Compute Conditional Entropy $H(D|\textbf{Outlook})$

Sunny: 5 instances (2 Yes, 3 No)

$$\begin{aligned} H(D_{Sunny}) &= -\left(\frac{2}{5}\log_2\frac{2}{5} + \frac{3}{5}\log_2\frac{3}{5}\right) \\ &= -\left(0.4 \cdot (-1.322) + 0.6 \cdot (-0.737)\right) \\ &\approx 0.971 \end{aligned}$$

Overcast: 4 instances (4 Yes, 0 No) $\Rightarrow H = 0$.
Rain: 5 instances (3 Yes, 2 No)

$$\begin{aligned}
H(D_{Rain}) &= -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right) \\
&= -\left(0.6 \cdot (-0.737) + 0.4 \cdot (-1.322)\right) \\
&\approx 0.971
\end{aligned}$$

Weighted Conditional Entropy:

$$\begin{aligned}
H(D|\text{Outlook}) &= \frac{5}{14} \cdot 0.971 + \frac{4}{14} \cdot 0 + \frac{5}{14} \cdot 0.971 \\
&= \frac{10}{14} \cdot 0.971 \\
&\approx 0.694
\end{aligned}$$

Compute Information Gain for Outlook

$$\begin{aligned}
IG(D, \text{Outlook}) &= H(D) - H(D|\text{Outlook}) \\
&= 0.940 - 0.694 = 0.246
\end{aligned}$$

We do similar computation to find $IG(D, \text{Temp})$, $IG(D, \text{Wind})$, $IG(D, \text{Humidity})$.

Computing Information Gain of Temperature

- Hot: 4 instances → 2 Yes, 2 No → $H = 1$
- Mild: 6 instances → 4 Yes, 2 No → $H = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} \approx 0.918$
- Cool: 4 instances → 3 Yes, 1 No → $H = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} \approx 0.811$

$$\begin{aligned}
H(D|\text{Temp}) &= \frac{4}{14} \cdot 1 + \frac{6}{14} \cdot 0.918 + \frac{4}{14} \cdot 0.811 \\
&\approx 0.286 + 0.393 + 0.232 = 0.911 \\
IG(D, \text{Temp}) &= 0.940 - 0.911 = \boxed{0.029}
\end{aligned}$$

Computing Information Gain for Humidity

- High: 7 instances → 3 Yes, 4 No → $H = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} \approx 0.985$
- Normal: 7 instances → 6 Yes, 1 No → $H = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} \approx 0.592$

$$\begin{aligned} H(D|\text{Humidity}) &= \frac{7}{14} \cdot 0.985 + \frac{7}{14} \cdot 0.592 \\ &= 0.5 \cdot (0.985 + 0.592) \\ &= 0.789 \\ IG(D, \text{Humidity}) &= 0.940 - 0.789 = \boxed{0.151} \end{aligned}$$

Computing Information Gain of Wind

- Weak: 8 instances → 6 Yes, 2 No → $H = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} \approx 0.811$
- Strong: 6 instances → 3 Yes, 3 No → $H = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1$

$$\begin{aligned} H(D|\text{Wind}) &= \frac{8}{14} \cdot 0.811 + \frac{6}{14} \cdot 1 \\ &= 0.463 + 0.429 = 0.892 \\ IG(D, \text{Wind}) &= 0.940 - 0.892 \\ &= \boxed{0.048} \end{aligned}$$

Summary of Information Gain

- IG(Outlook) = 0.246
- IG(Humidity) = 0.151
- IG(Wind) = 0.048
- IG(Temperature) = 0.029

Therefore, **Outlook** has the highest information gain and is selected as the **root node**.

Next Step: ID3 on Subset with Outlook = Sunny

Subset with Outlook = Sunny:

Day	Temp	Humidity	Wind	Play?
1	Hot	High	Weak	No
2	Hot	High	Strong	No
8	Mild	High	Weak	No
9	Cool	Normal	Weak	Yes
11	Mild	Normal	Strong	Yes

Table 4.3: Subset D_{Sunny}

Class counts: 2 Yes, 3 No $\Rightarrow H(D_{Sunny}) \approx 0.971$.

Compute IG for remaining features (Temp, Humidity, Wind):

- **Humidity:**
  - High (3 instances, all No): $H = 0$
  - Normal (2 instances, both Yes): $H = 0$
  - $H(D_{Sunny}|Humidity) = \frac{3}{5} \cdot 0 + \frac{2}{5} \cdot 0 = 0$
  - IG = 0.971 - 0 = 0.971
- **Temp and Wind** can be computed similarly.

ID3 on Subset with Outlook = Rainy

Subset D_{Rain}:

Day	Temp	Humidity	Wind	Play?
4	Mild	High	Weak	Yes
5	Cool	Normal	Weak	Yes
6	Cool	Normal	Strong	No
10	Mild	Normal	Weak	Yes
14	Mild	High	Strong	No

Table 4.4: Subset D_{Rain}

Class distribution: 3 Yes, 2 No $\Rightarrow H(D_{\text{Rain}}) = 0.971$ (previously computed).

Compute IG for remaining features:

- **Wind:**
  - Weak: 3 instances (all Yes) $\Rightarrow H = 0$
  - Strong: 2 instances (both No) $\Rightarrow H = 0$
  - $H(D_{\text{Rain}}|\text{Wind}) = \frac{3}{5} \cdot 0 + \frac{2}{5} \cdot 0 = 0$
  - IG $= 0.971 - 0 = 0.971$
- **Humidity:**
  - High: 2 instances (1 Yes, 1 No) $\Rightarrow H = 1$

- Normal: 3 instances (2 Yes, 1 No)

$$H = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) \approx 0.918$$

- Weighted entropy:

$$H = \frac{2}{5}\cdot 1 + \frac{3}{5}\cdot 0.918 \approx 0.951$$

- IG = 0.971 − 0.951 = 0.020

- **Temp:**

  - Mild: 3 instances (2 Yes, 1 No) ⇒ $H \approx 0.918$
  - Cool: 2 instances (1 Yes, 1 No) ⇒ $H = 1$
  - Weighted entropy:

$$H = \frac{3}{5}\cdot 0.918 + \frac{2}{5}\cdot 1 \approx 0.951$$

  - IG = 0.971 − 0.951 = 0.020

Conclusion: Wind gives the highest information gain. So we split on **Wind**:

- Wind = Weak ⇒ 3 Yes ⇒ Leaf Node: Yes
- Wind = Strong ⇒ 2 No ⇒ Leaf Node: No

ID3 on Subset with Outlook = Overcast

Subset D_{Overcast}:

Day	Temp	Humidity	Wind	Play?
3	Hot	High	Weak	Yes
7	Cool	Normal	Strong	Yes
12	Mild	High	Strong	Yes
13	Hot	Normal	Weak	Yes

Table 4.5: Subset D_{Overcast}

All instances are labeled `Yes` ⇒ $H(D_{\text{Overcast}}) = 0$

This is a pure leaf node and doesn't require further splitting.

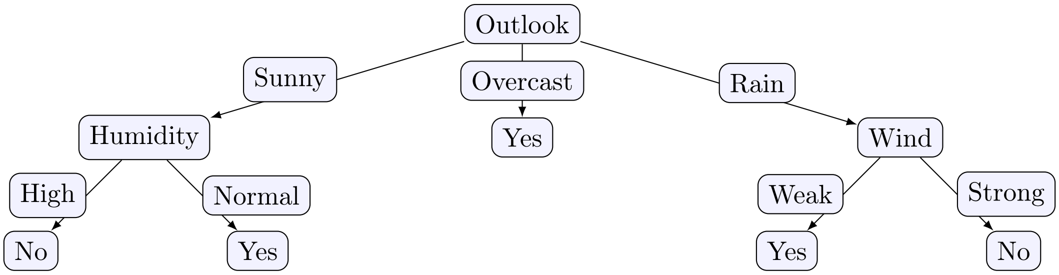


Figure 4.3: Final ID3 Decision Tree for Play Tennis Dataset

Special Example: Computing Entropy With Non-Binary Class

Consider a small dataset with 3 class labels: A, B, and C. The class distribution is as follows:

Class	Frequency
A	3
B	2
C	1

Table 4.6: Small Dataset with 3 Classes

Total instances: $N = 6$

Class probabilities:

$$P_A = \frac{3}{6} = 0.5$$
$$P_B = \frac{2}{6} \approx 0.333$$
$$P_C = \frac{1}{6} \approx 0.167$$

Entropy is calculated as:

$$\begin{aligned} H(D) &= -\left(P_A \log_2 P_A + P_B \log_2 P_B + P_C \log_2 P_C\right) \\ &= -\left(0.5 \cdot \log_2 0.5 + 0.333 \cdot \log_2 0.333 + 0.167 \cdot \log_2 0.167\right) \\ &= 1.46 \text{ bits (approx)} \end{aligned}$$

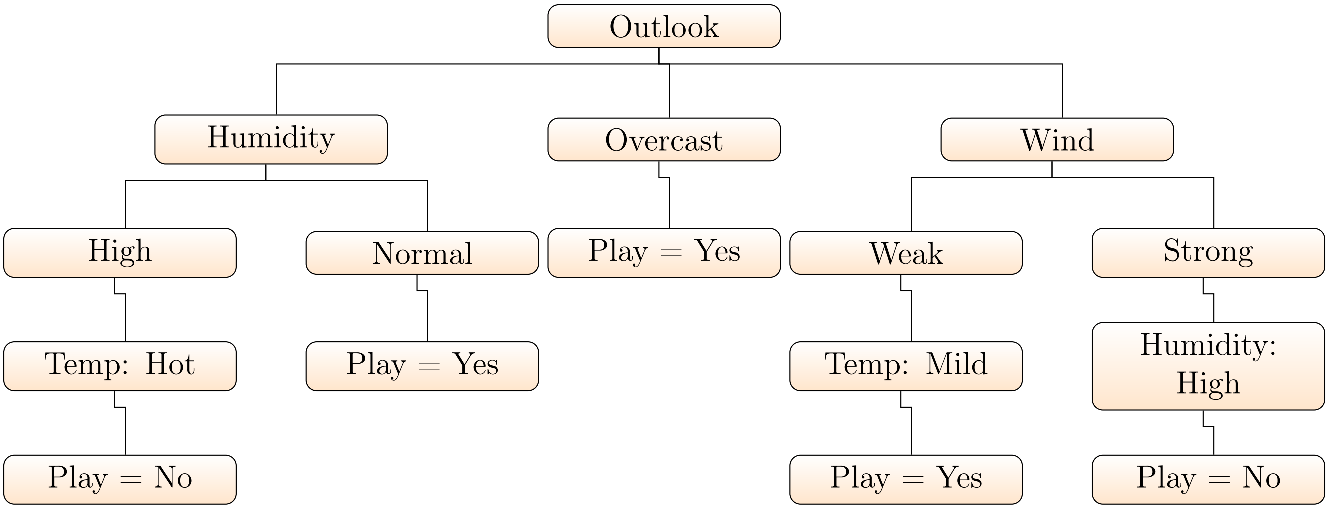
For, k number of class labels, we will have to use probability distribution $[P_1 \ldots P_k]$, P_i representing the probability of the i'th class label.

4.7 Overfitting and Pruning

Overfitting

A decision tree becomes overfitted when it grows too deep and begins to memorize noise or minor fluctuations in the training data, rather than learning the true underlying patterns. This often leads to excellent performance on the training set but poor generalization to unseen data.

Example: Overfit Tree



In this tree, the model makes decisions based on very specific combinations of attributes such as *Temp* = Hot and *Humidity* = High or *Wind* = Strong and *Humidity* = High, rather than broader, generalizable patterns. This level of detail may capture noise in the training dataset rather than meaningful trends.

Suppose a training dataset has a few instances where

- When *Outlook* = *Sunny*, *Humidity* = *High*, and *Temp* = *Hot*, the player didn't play.
- But for *Outlook* = *Sunny*, *Humidity* = *High*, and *Temp* = *Mild*, the player did play.

If the tree tries to fit such fine distinctions, it may become too sensitive to slight variations and overfit.

Underfitting

Underfitting occurs when a decision tree is too shallow or too simple to capture the underlying structure of the data. It fails to learn the relationships between features and the target class, resulting in poor performance on both the training and test sets.

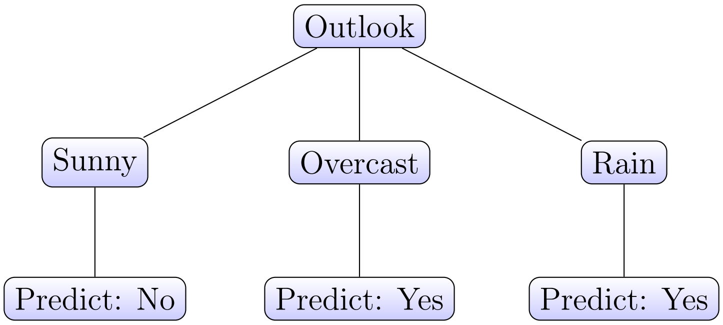
An underfit model makes overly broad generalizations and may return the same prediction across many different inputs. This can happen due to early stopping, excessive pruning, or not including enough splits to isolate relevant patterns.

Example of Underfitting

Imagine trying to model a dataset using only the root node feature, without allowing the tree to explore deeper levels. For instance, if we split solely on *Outlook*, but ignore important distinctions made by *Humidity* or *Wind*, the tree might generalize:

- All "Overcast" days result in "Yes"
- All "Sunny" days result in "No"
- All "Rainy" days result in "Yes"

While this may cover dominant patterns, it ignores the nuances (e.g., differences based on humidity or wind strength), leading to high error on both known and new data.



Common Causes of Underfitting

- Tree depth restricted too early (pre-pruning)
- High minimum sample split thresholds
- Limited features available or used

Solution

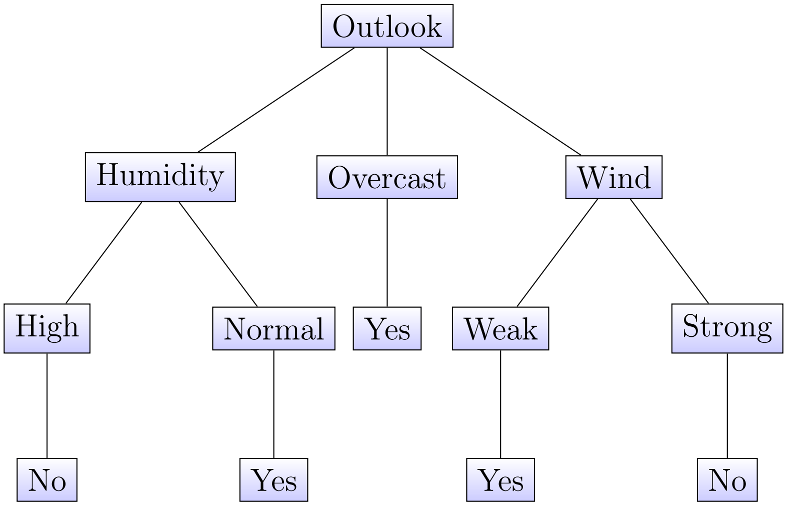
Allow the tree to grow deeper and use more features until training performance improves, followed by pruning to improve generalization.

Post-Pruning

To combat overfitting, we apply pruning strategies:

- **Post-pruning:** Build the full tree and then remove branches that do not improve accuracy on a validation set.

- **Pre-pruning (early stopping):** Stop tree construction early if further splits do not add significant value.



4.8 Handling Special Cases

4.8.1 Missing Data

Handling missing data is essential for building robust decision trees. There are several strategies to deal with missing attribute values:

- **Ignore the instance:** Remove records with missing values, especially if they form a small portion of the dataset.
- **Impute the missing value:** Replace with the mean (for numerical attributes), mode (for categorical attributes), or a more advanced method like KNN or regression-based imputation.

4.8.2 Continuous Attributes

Decision trees can handle continuous (numeric) features by choosing optimal split points:

- Determine possible thresholds (e.g., average of adjacent values).
- Evaluate each threshold using information gain (or Gini impurity).
- Select the threshold that maximizes the split quality.

Example: Given temperatures [60, 70, 80] and their associated play outcomes, try a threshold like Temp ≤ 65 to split. If this improves the purity of child nodes, the threshold is retained.

4.8.3 Regression Trees: Handling Numerical Target Variables

When the target variable is numerical rather than categorical, we use regression trees instead of classification trees.

- These are suitable for tasks like predicting house prices, exam scores, or temperature.
- The tree is constructed by splitting the data at each node based on a feature and a threshold that minimizes prediction error — typically measured using metrics like Mean Squared Error (MSE) or Mean Absolute Error (MAE).
- Unlike classification trees that output a class label, regression tree leaves output a real-valued prediction (usually the average value of target variables in that node).

Example: Predicting House Prices

- Suppose we're predicting house prices based on area, number of bedrooms, and location score.
- A node might split on the rule "Area $\leq$ 2000 sq ft", separating smaller and larger homes.
- The leaf nodes would return values such as \$150,000 for smaller homes and \$320,000 for larger ones, based on the average price within each group.