Scribe

Moneesh B

March 4th, 2025

1 Supervised Learning Framework

1.1 Understanding the Learning Framework

Supervised learning is a type of machine learning where a model is trained on labeled data. The objective is to learn an **unknown target function** $f : X \to Y$, which maps input features X to output labels Y. This function is unknown because we do not have explicit knowledge of how the inputs are transformed into outputs; instead, we rely on learning from a dataset of training examples.

1.2 Key Components

- **Input space** X: Represents the set of all possible input values.
- **Output space** Y: Represents the set of all possible output values.
- **Training examples**: The dataset consists of a finite set of paired observations:

$$(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$$

  where each X_i is an input feature vector and each Y_i is the corresponding output label.

1.3 Learning Process

The goal of supervised learning is to generalize from training examples to make accurate predictions on unseen data. This process involves:

1. **Receiving training examples**: The learning algorithm is given labeled data consisting of feature vectors and their corresponding outputs.
2. **Selecting a hypothesis**: The algorithm chooses a function h from a predefined **hypothesis set** H, which contains all the possible models the algorithm can consider:

$$H = \{h_1, h_2, ..., h_n\}$$

   In the case of decision trees, these hypotheses are represented as Boolean formulas that define the splitting criteria.
3. **Choosing the final hypothesis**: The model that best fits the training data is selected as the final hypothesis h^*, which serves as an approximation of the unknown target function f.

1.4 Feasibility of Learning

A fundamental question in supervised learning is whether it is possible to learn a function that generalizes well from a finite number of training examples. The **Probably Approximately Correct (PAC) learning framework** provides a probabilistic bound on how well the chosen hypothesis approximates the target function.

This is quantified using **Hoeffding's Inequality**, which provides an upper bound on the probability that the empirical error (measured on the training data) significantly differs from the true error (measured on the entire distribution of possible inputs):

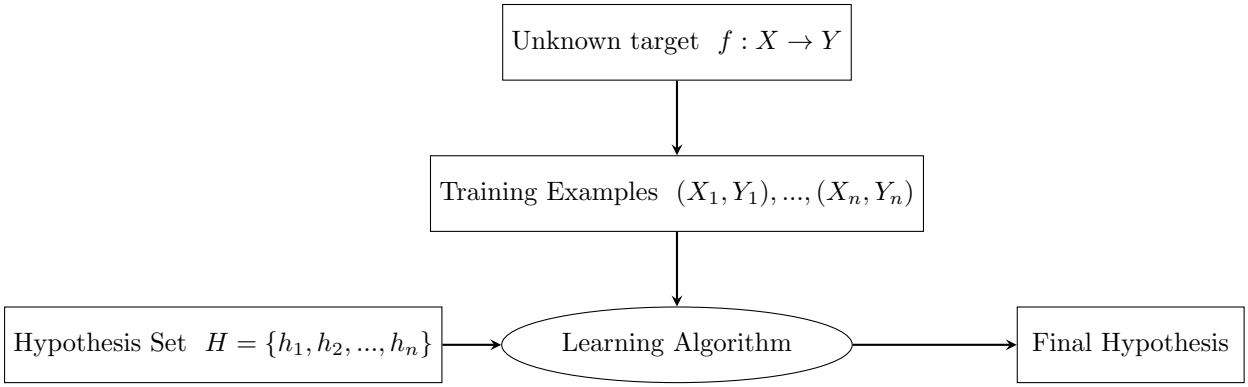
$$P\left[|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon\right] \leq 2me^{-2\epsilon^2 n}$$

where:

- $E_{\text{in}}(g)$ is the in-sample/training error of hypothesis g.
- $E_{\text{out}}(g)$ is the true error of g on out of sample/unseen data.
- ϵ is the error threshold.
- m is the number of hypotheses in the hypothesis set.
- n is the number of training examples.

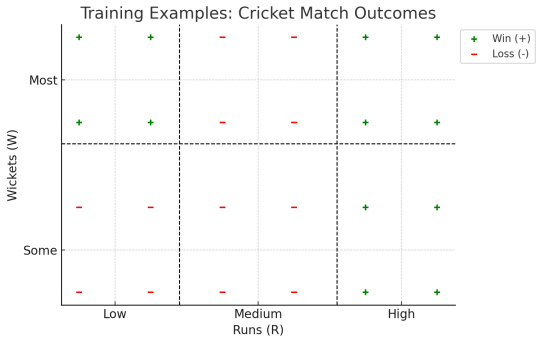
The bound shows that as the number of training examples n increases, the probability of a large difference between training error and true error decreases exponentially. This provides a theoretical foundation for why supervised learning works under suitable conditions, reinforcing the idea that with sufficient data, a good approximation of the target function can be learned. **If we can handle the in-sample errors well, the out of sample error takes care of itself according to Hoeffding's Inequality.**

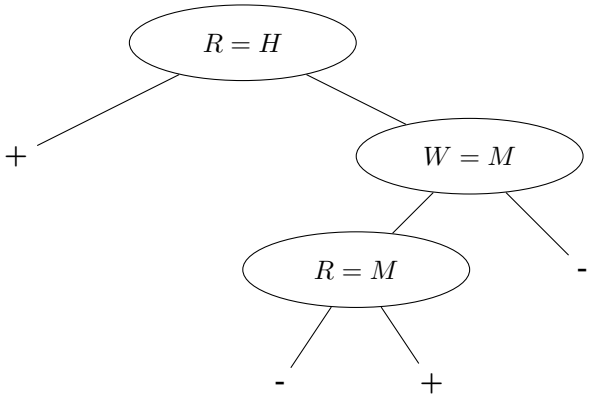
Supervised Learning Process



2 Decision Tree

Hypothesis Set: Boolean Expressions
Learning Algorithm: Decision Tree (ID3)





Boolean Expression

$$(R = H) \vee (\neg(R = H) \wedge (W = M) \wedge \neg(R = M))$$

Decision trees are a widely used method for making structured decisions based on historical data. In this section, we explore how a decision tree can classify match outcomes based on past data from the Indian cricket team's matches. The dataset consists of two key features:

- **Runs (R)** categorized into *Low, Medium, High*.
- **Wickets (W)** categorized into *Some* (≤ 7 wickets lost) and *Most* (> 7 wickets lost).

By analyzing these two features, we aim to determine whether the team won or lost the match. The scatter plot below represents the training data, where:

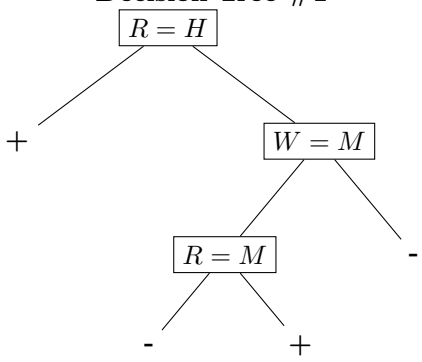
- Each + symbol represents a **win**.

- Each $-$ symbol represents a **loss**.
- The x-axis divides **Runs (R)** into *Low, Medium, and High* categories.
- The y-axis divides **Wickets (W)** into *Some* and *Most* categories.
- The dashed lines indicate the **decision boundaries**, helping visualize how different regions of the feature space correspond to wins and losses.

When constructing a decision tree, there can be multiple ways to split the data that result in the same classification of outcomes. The key challenge is not just to find a decision tree that correctly classifies the data but to determine the best possible tree among the many valid options.

A decision tree works by recursively splitting the dataset based on feature values until all points in a region belong to the same class. However, the order in which we make these splits is not fixed—there can be multiple ways to arrange the tree that still correctly classify the data.

Decision Tree #1

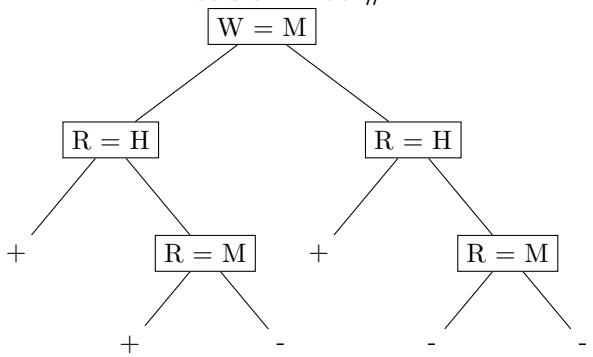


For an arbitrary point, average comparisons to decide:

$$\frac{1+3+3+2}{4} = \frac{9}{4}$$

This is computationally easier, so should be preferred.

Decision Tree #2



For an arbitrary point, average comparisons to decide:

$$\frac{2+3+3+2+2}{5} = \frac{12}{5}$$

Above, we present two potential decision trees that classify training data correctly. The first tree tends to be shallower on average, while the second tree may have a different structure that impacts its average comparisons.

Choosing the Best Tree:
While average comparisons are one way to gauge computational efficiency, a more rigorous approach to selecting the best attribute splits involves looking at *entropy*, the *Gini index*, and the corresponding *information gain*. By computing these metrics at each step, we can systematically determine which attribute should be chosen first, then second, and so on. This helps us build a tree that balances both depth and predictive accuracy.

3 Shannon Entropy

Shannon Entropy, introduced by Claude Shannon, measures the average uncertainty or information content in a random variable. In the context of decision trees, if a set S has class labels with probabilities

$p_1, p_2, \ldots, p_k$, the entropy $E(S)$ is given by:

$$E(S) = -\sum_{i=1}^{k} p_i \log_2(p_i).$$

For a binary classification (where $k = 2$ and $p_1 = p,\ p_2 = 1 - p$), the entropy simplifies to:

$$E(p) = -\,p \log_2(p)\ -\ (1-p)\ \log_2(1-p).$$

The plot below shows how $E(p)$ varies for p from 0 to 1, peaking at $p = 0.5$.

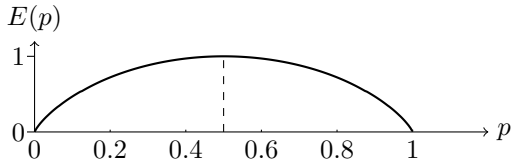


Figure 1: Plot of Shannon Entropy $E(p)$ for a binary distribution ($p \in [0, 1]$).

4 Comparing the two trees

4.1 Entropy of the Dataset

In a binary classification problem, the entropy for a node (or branch) is given by

$$E = -\Big(p_+ \log_2(p_+) + p_- \log_2(p_-)\Big),$$

where p_+ and p_- are the proportions of positive and negative instances, respectively. For example, if the dataset is perfectly balanced (i.e., $p_+ = p_- = 0.5$), then

$$E_{\text{base}} = 1 \text{ bit}.$$

When an attribute is used to split the dataset, it is partitioned into subsets. Suppose the dataset is split into two subsets: one with N_1 examples and the other with N_2 examples (with $N = N_1 + N_2$). Each subset will have its own entropy, say E_1 and E_2. The overall entropy after the split is computed as the weighted average of these entropies:

$$E_{\text{split}} = \frac{N_1}{N} E_1 + \frac{N_2}{N} E_2.$$

Here, the weights $\frac{N_1}{N}$ and $\frac{N_2}{N}$ represent the fraction of examples in each subset.

The *information gain* (IG) from the split is the reduction in entropy, and is calculated as:

$$\text{IG} = E_{\text{base}} - E_{\text{split}}.$$

A higher information gain indicates that the split has effectively reduced uncertainty, making it more useful for classification.

This procedure of computing the entropy for each node, weighting the entropies of the splits, and calculating the information gain is repeated at every stage of the decision tree construction until the data is perfectly classified or no further improvement can be achieved.

4.2 Base Entropy of the Dataset

Suppose our entire training set has an equal number of positive (+) and negative (−) examples. That means $p_+ = p_- = \frac{1}{2}$. Hence, the *base entropy* E_{base} of the full dataset is

$$E_{\text{base}} \ = \ -\Big(\tfrac{1}{2} \log_2 \tfrac{1}{2} \ + \ \tfrac{1}{2} \log_2 \tfrac{1}{2}\Big) \ = \ -\Big(\tfrac{1}{2} \times (-1) + \tfrac{1}{2} \times (-1)\Big) \ = \ 1 \text{ bit}.$$

We will use $E_{\text{base}} = 1$ as the starting entropy to measure *information gain*.

4.3 Decision Tree #1 Entropy

The first split is on the attribute R.

- **Left branch ($R = H$):** This branch yields a pure positive outcome (denoted by $+$). Hence,

$$p(+) = 1, \quad p(-) = 0 \quad \Longrightarrow \quad E_{\text{left}} = -\Big(1 \cdot \log_2(1) + 0 \cdot \log_2(0)\Big) = 0.$$

- **Right branch ($R \neq H$):** In this branch (which continues with $W = M$), the subsequent outcomes are:
  - One outcome yields $+$ (from one of the leaves of the split on $R = M$).
  - Two outcomes yield $-$ (one directly from $W = M$ and one from the other branch of $R = M$).

  Thus, the class probabilities are

$$p(+) = \frac{1}{4}, \quad p(-) = \frac{3}{4}.$$

  and the entropy for the right branch is

$$E_{\text{right}} = -\left(\frac{1}{3}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right) \approx 0.978 \text{ bits.}$$

Assuming that the branch probabilities correspond to the number of outcomes (1 outcome from the left branch and 3 from the right branch, totaling 4 outcomes), the weights are:

$$w_{\text{left}} = \frac{1}{3}, \quad w_{\text{right}} = \frac{2}{3}.$$

Therefore, the overall entropy after the first split is:

$$E_{\text{split}} = \frac{1}{3} \cdot 0 + \frac{2}{3} \cdot 0.9183 \approx 0.6122 \text{ bits.}$$

Information Gain (IG) for Tree #1.

$$\text{IG}_1 = E_{\text{base}} - E_{\text{split}} = 1.0 - 0.6122 \;=\; 0.3878 \text{ bits (approx.).}$$

4.4 Decision Tree #2 Entropy

For Decision Tree #2, the first split is on the attribute W.

- **Branch 1:** In this branch, the outcomes (after a subsequent split on R) are:
  - Two outcomes yield $+$.
  - One outcome yields $-$.

  Thus, the class probabilities are:

$$p(+) = \frac{2}{3}, \quad p(-) = \frac{1}{3},$$

  and the entropy is

$$E_1 = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) \approx 0.9183 \text{ bits.}$$

- **Branch 2:** In this branch, the outcomes are:
  - One outcome yields $+$.
  - Two outcomes yield $-$.

  Hence,

$$p(+) = \frac{1}{3}, \quad p(-) = \frac{2}{3},$$

  and the entropy is

$$E_2 = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) \approx 0.9183 \text{ bits.}$$

Since each branch has equal weight ($w_1 = w_2 = \frac{1}{2}$), the overall entropy after the first split is:

$$E_{\text{split}} = \frac{1}{2}E_1 + \frac{1}{2}E_2 = \frac{1}{2}(0.9183) + \frac{1}{2}(0.9183) = 0.9183 \text{ bits.}$$

Information Gain (IG) for Tree #2.

$$\mathrm{IG}_2 = E_{\text{base}} - E_{\text{split}} = 1.0 - 0.9183 \;=\; 0.0817 \text{ bits (approx.)}.$$

Comparison

Since $\mathrm{IG}_1 \approx 0.3878$ bits is greater than $\mathrm{IG}_2 \approx 0.0817$ bits, the first split on R (Tree #1) provides a larger reduction in entropy than the first split on W (Tree #2). Therefore, based on *information gain*, *Decision Tree #1* is preferable at this stage.

Conclusion

The process of computing entropy and information gain, as demonstrated above for the first split, must be repeated at every stage of the decision tree construction. At each node, the attribute that provides the highest information gain is chosen for splitting, and the corresponding entropy is computed for the resulting branches. This recursive procedure continues until every record is correctly classified (i.e., until the nodes become pure) or until no further improvement can be achieved by additional splits.

This approach is the foundation of the **ID3 algorithm** developed by Ross Quinlan in 1987, which builds the complete decision tree by iteratively selecting the most informative attribute at each stage. In doing so, the algorithm effectively reduces uncertainty in the data, resulting in a model that classifies the records with increasing precision at each level.

5 Gini Index

The *Gini index* is an impurity measure used in decision trees. For a classification problem with k classes, if p_i is the probability of class i (with $\sum_{i=1}^{k} p_i = 1$), then the Gini index is defined as:

$$\text{Gini} \;=\; 1 - \sum_{i=1}^{k} p_i^2.$$

For the special case of binary classification ($k = 2$), let p be the probability of the positive class and $1 - p$ the probability of the negative class. The formula simplifies to:

$$\text{Gini}(p) \;=\; 1 - \Big(p^2 + (1-p)^2\Big) \;=\; 2p(1-p).$$

Advantages over Entropy

One key advantage of the Gini index over entropy is its computational simplicity. Entropy involves logarithmic calculations, while the Gini index is computed using only basic arithmetic (squaring and subtraction). This makes the Gini index faster to compute, which can be especially beneficial when handling large datasets.

Graph of the Gini Index for Binary Classification

Below is the function $\text{Gini}(p) = 2p(1-p)$ for $p \in [0, 1]$.

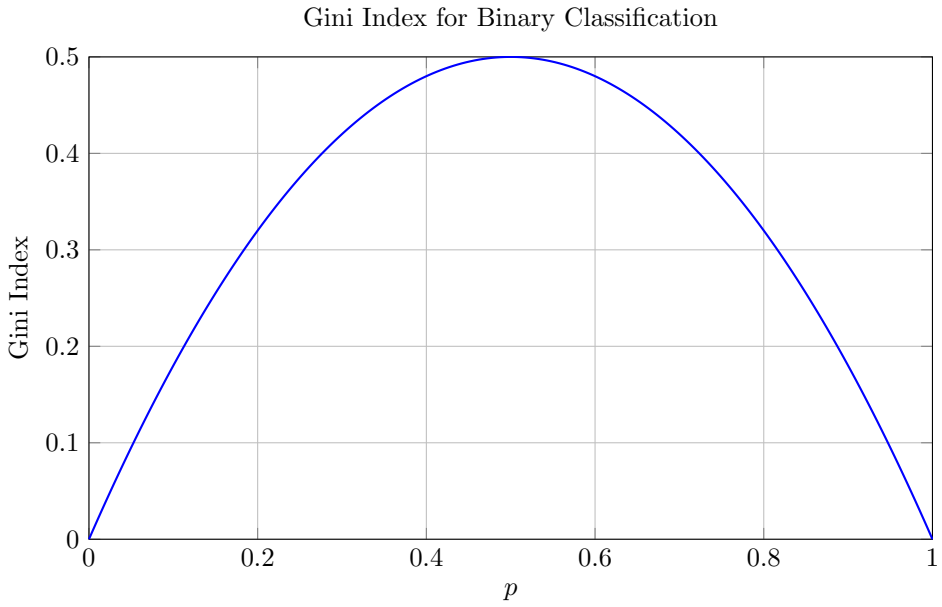


Figure 2: Gini Index as a function of p for binary classification.

6 Conclusion and Final Remarks

A guiding principle in model selection is *Occam's Razor*, which states that the simplest hypothesis that explains the data is generally preferable. In the context of decision trees, this translates to preferring smaller (shallower) trees that still achieve high accuracy, thereby avoiding overfitting and improving generalization.

Notably, the process we described—selecting the attribute that yields the highest information gain (or lowest impurity)—is inherently *greedy* or *best-first*. Once a split is chosen, the algorithm does not backtrack to change it later, introducing an *inductive bias* toward splits that locally reduce entropy or Gini index the most. While this often leads to effective trees in practice, it also means that if an early split is suboptimal, there is no mechanism to revisit and correct it.