Understanding Decision Trees: From Medical Diagnosis to Machine Learning

Rishabh Iyer

March 19, 2024

1 Introduction

Visiting a doctor with symptoms, the diagnostic process involves a sequence of conditional questions about symptoms like fever, sore throat, cough, and back pain, guiding the doctor to a potential diagnosis, such as a cold, flu, or an infection like a urinary tract infection. This step-by-step method of narrowing down diagnoses based on symptom presence mirrors the functioning of decision trees in machine learning, where data attributes guide the path to a conclusion, as depicted in Figure 1.

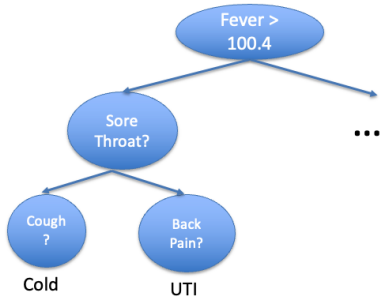


Figure 1: An example how a Doctor would ask questions to zero-in on a disease for a patient.

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g., whether a cough is present), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules or regression paths.

This analogy highlights the essence of decision trees - they are fundamental, interpretable machine learning models that mimic human decision-making processes. Just as a doctor uses medical knowledge and logical reasoning to deduce a patient's condition, a decision tree algorithm uses data-driven learning to make predictions or decisions. It systematically divides the data into smaller and smaller subsets based on specific criteria (features), which is akin to a doctor narrowing down diagnoses based on responses to questions.

The beauty of decision trees lies in their simplicity and interpretability. They do not require complex mathematical formulations to understand or visualize. Each decision or split in the tree is made based on straightforward logic, much

like the yes-or-no questions a doctor might ask. This makes decision trees an invaluable tool in the machine learning toolkit, offering clear insights into how predictions are made, which is particularly useful in fields where understanding the rationale behind a model's decision is crucial, such as in medicine.

Furthermore, decision trees serve as the foundation for more complex ensemble methods like Random Forests and Gradient Boosted Trees, where multiple trees are combined to improve predictive performance. Yet, at their core, these advanced models still rely on the simple, yet powerful, decision-making process that characterizes individual decision trees.

2 How Decision Trees Work

2.1 Partitioning the Feature Space

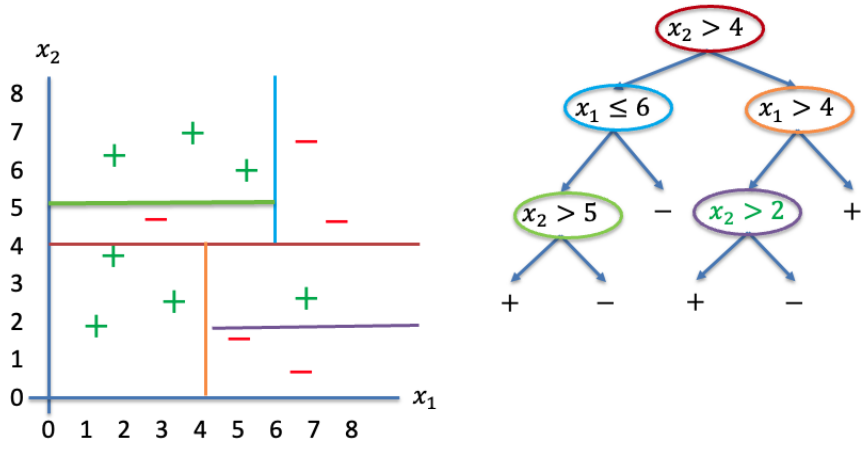


Figure 2: Illustrating an example Decision tree and the resulting splitting of the region. Each split in the feature space shown on the left (in the feature space) has the same color as the node in the decision tree, showing how the decision tree essentially partitions the feature space into axis-aligned regions.

Decision trees create a hierarchical division of the feature space into axis-aligned rectangles, which can be mathematically represented as a series of conditional splits. Each internal node n in the tree corresponds to a binary decision:

$$\text{if } x_i \leq \theta \text{ then go to left child, else go to right child}$$

where x_i is a feature, and θ is the threshold for splitting. The process of choosing x_i and θ at each node is determined by criteria, that we will describe in the section below. The partitioning can be represented as:

$$R_j = \{X | a_{j1} < x_i \leq a_{j2}\}$$

for each region R_j created by the tree, where a_{j1} and a_{j2} are the boundaries defined by splits on feature x_i across the tree.

Figure 2 illustrates an example decision tree on the right hand side. The left side shows the resulting partitioning. Each boundary split on the left figure (partitioning) has the same color as the node on the right hand side decision tree, thereby showing how each split in the decision tree results in a partitioning in the feature space. This is a simple 2D example with two features x_1, x_2.

2.2 Inference in Decision Trees

Given a new instance X_{new}, inference in a decision tree is a matter of applying the binary decisions at each node based on X_{new}'s features until a leaf node is reached. The prediction $\hat{y}$ for X_{new} is then given by:

$$\hat{y} = \begin{cases} \text{mode}\{y_{\text{leaf}}\} & \text{for classification} \\ \text{mean}\{y_{\text{leaf}}\} & \text{for regression} \end{cases}$$

where y_{leaf} represents the set of target values in the leaf node. For classification, the mode (most frequent label) of y_{leaf} is returned as the prediction. For regression, the mean of y_{leaf} provides the predicted value.

This mathematical framework underpins the operation of decision trees, from how they divide the feature space to make it simpler to understand the relationships within the data, to how they apply these divisions to make accurate predictions or decisions based on new data.

3 Training Decision Trees

The training of decision trees involves a recursive, greedy approach:

1. Identify the best feature and threshold to split the data at the current node.
2. Partition the data into two groups based on the chosen feature and threshold.
3. Recurse on each partition to build sub-trees.

3.1 Choosing the Best Feature and Threshold

The process of building a decision tree hinges on selecting the optimal feature and threshold for splitting the data at each node. This decision is guided by criteria designed to maximize the predictive power of the tree by creating the most informative partitions possible.

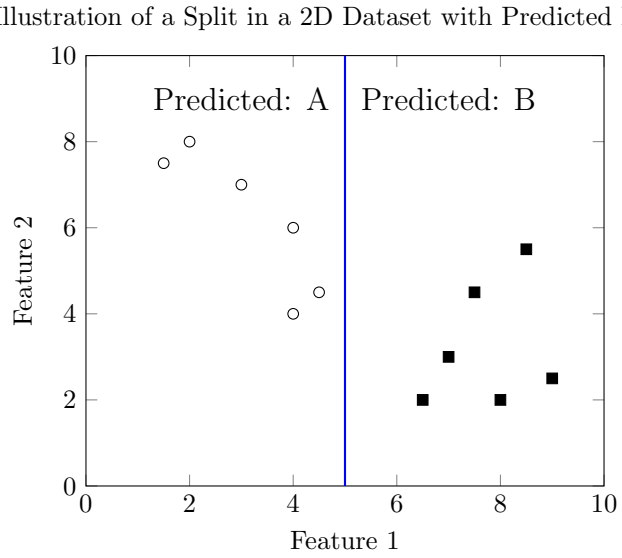


Figure 3: A simple 2D dataset with a vertical split at Feature 1 = 5, showing the predicted label for each partition (class A is the circles and class B are the Squares)

Fundamental Intuition Behind the Splitting Conditions: The underlying intuition behind any splitting criteria for classification problems is to achieve as much **class homogeneity as possible in each partition**. By doing so, the decision tree aims to isolate samples from different classes into separate partitions, thereby simplifying the classification problem. This approach ensures that the decision boundaries drawn by the tree align closely with the true distribution of classes in the feature space, leading to more accurate and reliable predictions. In the context of decision trees for regression, the splitting criteria differ from those used in classification. Instead of focusing on class homogeneity, **regression trees aim to reduce variance or the mean squared error within each node**.

3.1.1 Splitting Conditions for Classification

Figure 3 illustrates a simple 2D dataset with two classes and the split. The split on Feature 1 obtains a perfect split and homogeneous classes after split (the blue line split divides the dataset into one region being only circles (class A) and the other region only squares (class B). No split with feature 2 will obtain that and this shows that Feature 1 is the desired split. Below, we will will consider two criteria that will ensure this.

Information Gain: Information gain, on the other hand, measures the reduction in entropy before and after a dataset is split on an attribute (recall

entropy from this (**CITE**) blog article). It is calculated as the difference between the entropy of the labels in parent dataset (before the split) and the sum of the entropies of the label distributions in the two child datasets (after split), weighted by their sizes:

$$\text{Information Gain} = H(S) - \left(\frac{|S_1|}{|S|} H(S_1) + \frac{|S_2|}{|S|} H(S_2) \right)$$

where $|S|$ is the total number of samples in the parent set, and $|S_1|$ and $|S_2|$ are the numbers of samples in the two child sets created by the split. The goal is to find a split that maximizes the information gain, effectively reducing uncertainty with each split. Alternatively, we can also find a split that reduces the conditional entropy after the split, i.e., reduces $\frac{|S_1|}{|S|} H(S_1) + \frac{|S_2|}{|S|} H(S_2)$.

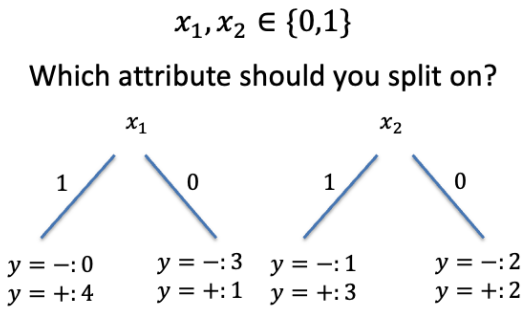
Gini Coefficient: The Gini coefficient is another popular measure used to evaluate splits, focusing on minimizing the probability of misclassification. It is defined for a set S as:

$$\text{Gini}(S) = 1 - \sum_{i=1}^{N} p_i^2$$

where, as before, p_i is the proportion of samples in S that belong to class i. The Gini coefficient measures the impurity of a dataset; a lower Gini coefficient indicates a purer dataset. When choosing where to split the data, the decision tree algorithm will seek the feature and threshold that result in child nodes with the lowest possible weighted Gini impurity. In other words, given a dataset S split into two subsets S_1 and S_2, we have $\text{Gini}(S_1)$ and $\text{Gini}(S_2)$ as the Gini-coefficients of the two subsets, the goal is to find a split that minimizes:

$$\text{Gini-Split} = \frac{|S_1|}{|S|} \text{Gini}(S_1) + \frac{|S_2|}{|S|} \text{Gini}(S_2)$$

Working out the Example from Figure 4: We now use the Information Gain and Gini coefficient to work out which feature x_1 or x_2 do we split on. Note that both features are categorical features so it is easy to calculate the information gain and gini coefficient. Let us start with information gain. Before split, the entropy of the labels is $H(S) = -3/8 \log(3/8) - 5/8 \log(5/8)$. Next, consider splitting on x_1. This results into two child datasets with label distributions shown in Figure 4. The value of $\frac{|S_1|}{|S|} H(S_1) + \frac{|S_2|}{|S|} H(S_2)$ with a split on x_1 is $1/2(0) + 1/2(-3/4 \log(3/4) - 1/4 \log(1/4))$, since the split on x_1 results in sets S_1, S_2 of the same size (note that when $x_1 = 1$ the entropy after the split of the child dataset is 0). You can similarly calculate the value of the conditional entropy for the split on x_2 and this value is $1/2(-1/2 \log(1/2) - 1/2 \log(1/2) + 1/2(-3/4 \log(3/4) - 1/4 \log(1/4)$. A quick check on the math will tell you that it is better to split on x_1 which also makes



x_1	x_2	y
1	1	+
1	0	+
1	1	+
1	0	+
0	1	+
0	0	−
0	1	−
0	0	−

Figure 4: Illustrating a simple dataset with two features. We would like to find out which feature do we split on using the entropy and gini coefficient.

logical sense since x_1 results in a pure homogeneous dataset in one of the child nodes when $x_1 = 1$. In a similar light, we can also use the Gini coefficient. The Gini Split with x_1 is $1/2((1-1-0)+(1-9/16-1/16)) = 5/32$. The Gini split with x_2 is $1/2((1-1/4-1/4)+(1-1/16-9/16)) = 13/32$. Clearly the gini coefficient with x_1 is lower which indicates that it is the better split. To summarize, using both the information gain and the gini coefficient will first split on x_1 rather than x_2.

In the provided example, both measures identified the same optimal split due to its straightforward nature. Generally, when multiple potential splits exist, these two measures might yield different results. Nonetheless, both criteria aim to create more homogeneous datasets following the split.

3.1.2 Splitting Criteria for Regression

In regression decision trees, where the goal is to predict a continuous value, the splitting criteria are based on minimizing the variance or mean squared error (MSE) within each partition. This approach aims to create partitions where the target values are as close to each other as possible, thereby ensuring more accurate predictions.

The variance reduction for a split is calculated as follows:

$$\text{Variance Reduction} = \text{Var}(S) - \left(\frac{|S_1|}{|S|}\text{Var}(S_1) + \frac{|S_2|}{|S|}\text{Var}(S_2) \right)$$

where $\text{Var}(S)$ is the variance of the target variable in the parent set S, and $\text{Var}(S_1)$ and $\text{Var}(S_2)$ are the variances in the two child sets. The sizes of the parent and child sets are denoted by $|S|$, $|S_1|$, and $|S_2|$, respectively. The objective is to maximize the variance reduction, which corresponds to finding

the split that results in the most homogeneous child nodes in terms of the target variable.

Alternatively, the mean squared error (MSE) can be used, especially when the focus is on minimizing the average squared difference between the actual and predicted values. The MSE for a set S is defined as:

$$\text{MSE}(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \bar{y}_S)^2$$

where y_i are the target values in S, and $\bar{y}_S$ is the mean target value in S. The goal during splitting is to choose the feature and threshold that minimize the weighted sum of the MSE of the child nodes.

$$\text{MSE Reduction} = \text{MSE}(S) - \left(\frac{|S_1|}{|S|} \text{MSE}(S_1) + \frac{|S_2|}{|S|} \text{MSE}(S_2) \right)$$

The fundamental intuition behind these regression splitting criteria is to ensure that each partition (or leaf node) captures instances with similar target values as closely as possible. By doing so, the regression tree can make more precise predictions for new instances, as instances within the same leaf node are expected to have similar target values.

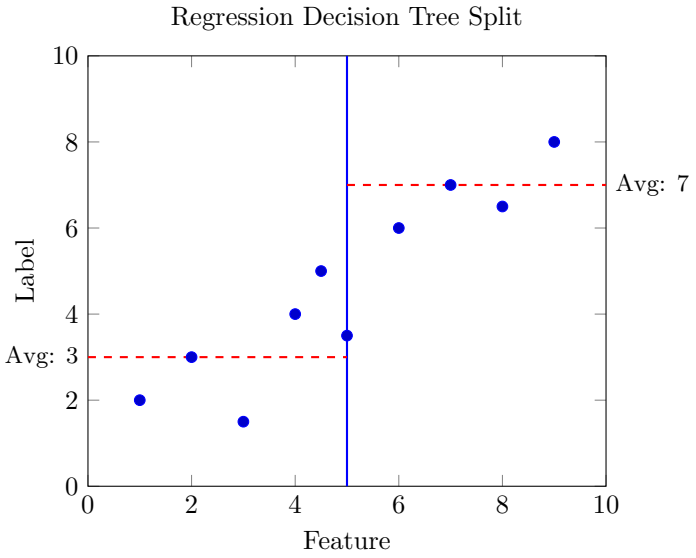


Figure 5: Illustration of a regression decision tree split with a single feature. The dataset is split at Feature = 5, with horizontal dashed lines representing the average label value for each partition.

Figure 5 illustrates how a decision tree would work with regression. It is a 1D dataset where the x-axis is the feature and y-axis is the label. The split is done at $x = 5$. On the left hand side, the predicted label is the average of all

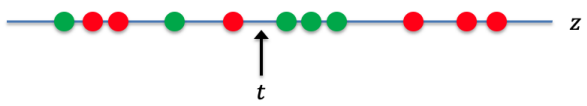


Figure 6: Illustrating the splitting condition for real-valued features.

the datapoints in that partition (in this case the average value is 3 on the left split and the average value is 7 for the right split).

3.2 Handling Different Types of Features

- **Categorical Features:** For categorical features, splits are based on whether a feature belongs to a category or not. Strategies include one-hot encoding or partitioning based on the category directly. The above example in Figure 4 was a categorical feature and the split is basically just simply splitting on the value of the categorical feature (for example, is $x_1 = 0$ or 1. In the case of more than two categories (say, $x_1 \in \{1, 2, 3\}$, we will simply split on $x_1 = 1$ or $x_1 \neq 1$.

- **Real-valued Features:** For continuous features, splits are made at a threshold that best separates the target classes, determined by evaluating the potential splits' effectiveness using the chosen criterion. But how do we pick the best threshold? We need to try every possible threshold, but since this is a continuous value, we cannot possibly search every possible value. The trick here is to sort the attribute z based on the feature values $z_1 > z_2 > \cdots > z_M$ (assuming there are M points in the dataset). We just need a finite number of thresholds then, and specifically, just split in between each consecutive pair of data points: $t = (z_i + z_{i+1})/2$. So we just need to compare at most M splits for d features. Figure 6 illustrates this.

4 Conclusion

Decision trees offer a powerful, interpretable framework for classification and regression tasks in machine learning. By systematically breaking down the decision-making process, much like a doctor diagnosing a patient, decision trees provide clear, logical reasoning for their predictions, making them invaluable tools in the machine learning toolkit.