Published in Towards Data Science · Follow

Rashida Nasrin Sucky · Follow
Feb 22 · 8 min read ★

Photo by Adarsh Kummur on Unsplash

Simple Explanation on How Decision Tree Algorithm Makes Decisions

Intuition Behind the Decision Tree Algorithm

The decision tree is a very popular machine learning algorithm. It works for both linear and non-linear data. Also, it can be used for both classification and regression. With great libraries and packages available in Python and R, anyone can easily use decision

In this article, I will try to give you an intuition on how a decision tree algorithm works.

> *This article will include:*

A high-level introduction to the Decision tree algorithm working process: Don't worry if this high-level introduction is not so clear. The next part is a detailed explanation with an example. That will be more clear.

An example and mathematical calculation behind the decisions in the decision tree: This part will show the real decision making by the decision tree algorithm using the mathematical calculation.

Introduction to the Decision Tree

The decision tree algorithm works based on the decision on the conditions of the features. Nodes are the conditions or tests on an attribute, branch represents the outcome of the tests, and leaf nodes are the decisions based on the conditions.

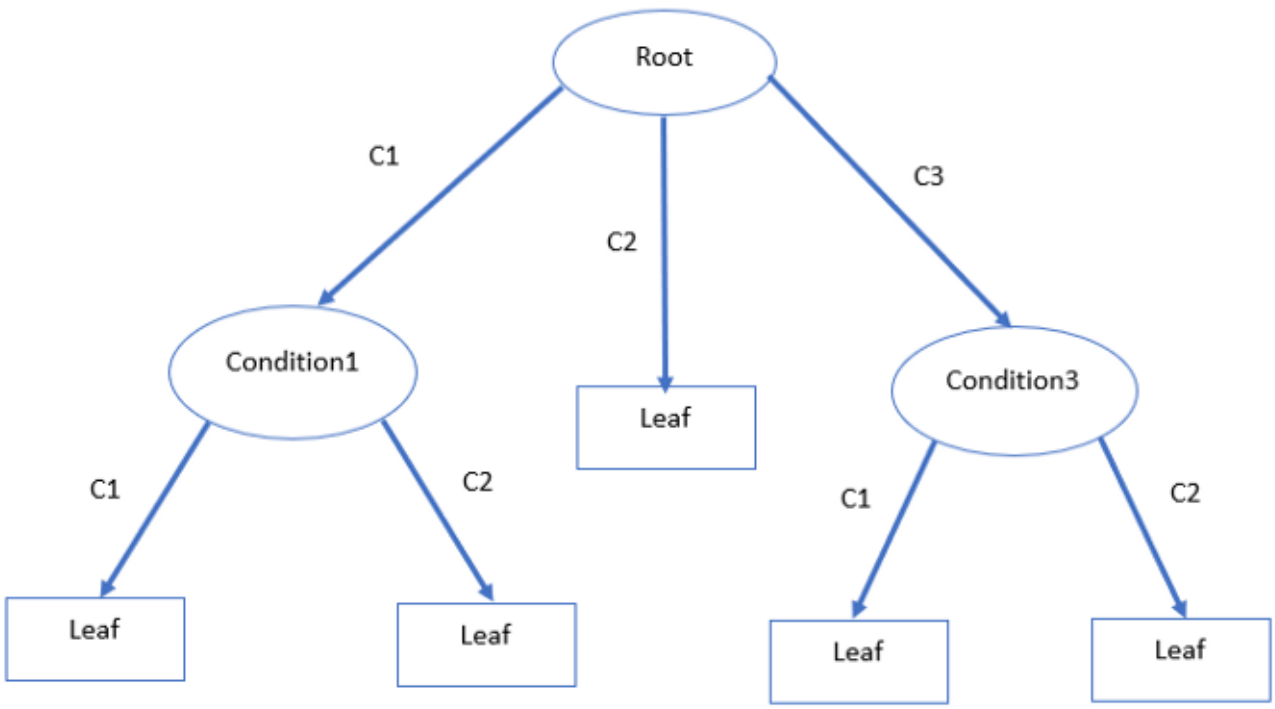


Image By Author

As you can see in the picture, It starts with a root condition, and based on the decision

decision or a prediction. C1 and C3 of the root node ended up with Confition1 and Condition2.

Condition1 and condition2 are two different features. We will further split the data based on the categories in condition1 and condition3. The picture above shows both condition1 and condition3 have two features each. As usual, the categories can be more than that. This is a high-level idea of how a decision tree makes predictions. We will see in some more detail how actually we chose features for the root node and other conditions.

These are some terms that need to be clear before diving deeper:

Root Nodes: As I already discussed, a root node is the beginning of the decision tree where the decision tree starts. This is the first feature that starts splitting the data.

Decision Nodes: Look at the 'condition1' and 'condition3' nodes above. We got those nodes after splitting based on the root node and we further split based on the decision node.

Leaf Nodes: These are the final nodes where the decision tree makes final decisions and no further splitting is possible. In the picture above Decision2 and Decision3 are the leaf nodes.

An Example and Mathematical Calculation Behind the Decisions in the Decision Tree

Here is a synthetic dataset I created for this example. This dataset has a binary class and three variables N1, N2, and N3. All three variables are categorical. I took all the categorical variables because it is easier to explain. **Once you get the intuition, it will be easier to understand the use of continuous variables as well.**

1	Rainy	Medium	East	Y
2	Cloudy	High	West	N
3	Cloudy	High	East	Y
4	Sunny	Medium	East	Y
5	Sunny	High	West	Y
6	Sunny	High	East	N
7	Rainy	Low	East	N
8	Sunny	Low	West	Y
9	Cloudy	Medium	West	N
10	Cloudy	Medium	East	Y
11	Sunny	High	East	Y
12	Rainy	High	East	N
13	Rainy	Low	West	Y
14	Sunny	Medium	East	N

Image By Author

We have the dataset. As per the previous discussion, we need to have a **root node** based on what we will start splitting the data. But we have three variables. **How to decide root node? Should I simply start with N1 because it comes first in the dataset?**

Not really. To understand how to make a good decision on which feature should be the root node, you need to learn about information gain.

The root node or first test attribute is selected based on a statistical measure called **information gain**.

Overall the selection of test attributes depends on the “purity measure”. These are the purity measures:

1. Information gain
2. Gain Ratio
3. Gini index

Imagine 10 friends are hanging out. 5 of them want to watch a movie and 5 of them want to go out. Making a decision is really hard in this case. If 8 or 9 of them would want to watch a movie and only 1 or 2 would want to go out, it will be easier to make a decision. Right? The best situation is if all 10 of them want the same thing. Decision-making is very easy. No confusion at all!.

In the decision tree also, if all the class belongs to either yes or no, the dataset is purest. On the other hand, if 50% of the class belongs to yes and the other 50% belongs to no, then the dataset is extremely impure. Because it is hard to make a decision. The decision is very uncertain!

Information gain helps to measure the reduction of uncertainty of a certain feature. **It also helps decide which feature is good as a root node.**

The calculation of information gain depends on **Info**. That is also called **entropy**.

The formula for Info:

$$I = -\sum p_i log_2(p_i) \, , where \, p_i = \frac{x_i}{|D|}$$

Let's work through an example.

I will work through the dummy dataset above to find out which feature will be the root node for this dataset.

Now, let's calculate the Info for our dummy dataset. This dataset has a total of 14 rows of data, 8 yes classes and 6 no classes.

So, the Info:

$$Info(D) = -\frac{8}{14} log_2\left(\frac{8}{14}\right) - \frac{6}{14} log_2\left(\frac{6}{14}\right)$$

Starting with N1:

We need to calculate the amount of information needed to classify the samples by splitting the dataset on N1:

First, let's dissect the data in N1. There are:

4 'Rainy' where 2 'Y's and 2 'N's class

4 'Sunny' where 2 'Y's and 2 'N's class

6 'Cloudy' where 4'Y's and 2'N's class

The information necessary to classify the dataset based on N1 is:

$$Info(D_{N1}) =$$

$$\frac{4}{14} * \left(-\frac{2}{4} log_2\left(\frac{2}{4}\right) - \frac{2}{4} log_2\left(\frac{2}{4}\right)\right)$$

$$+$$

$$\frac{4}{14} * \left(-\frac{2}{4} log_2\left(\frac{2}{4}\right) - \frac{2}{4} log_2\left(\frac{2}{4}\right)\right)$$

$$+$$

$$\frac{6}{14} * \left(-\frac{4}{6} log_2\left(\frac{4}{6}\right) - \frac{2}{6} log_2\left(\frac{2}{6}\right)\right)$$

It gives 0.964.

Info_gain(N1) = 0.985–0.964 = 0.02

So the **entropy** will decrease by 0.02 if N1 becomes the **root node**.

In the same way, we can calculate the Info(DN2) and Info (DN3) as 0.972 and 0.983 respectively.

So, the information gain for N2, and N3 are:




The **entropy** will decrease by 0.012 and 0.001 respectively if we make N2 or N3 our root node.

> *From the calculation above, the attribute with the highest information gain is N1. So, N1 will be the root.*

At this stage with our root node fixed, the decision tree will look like this:

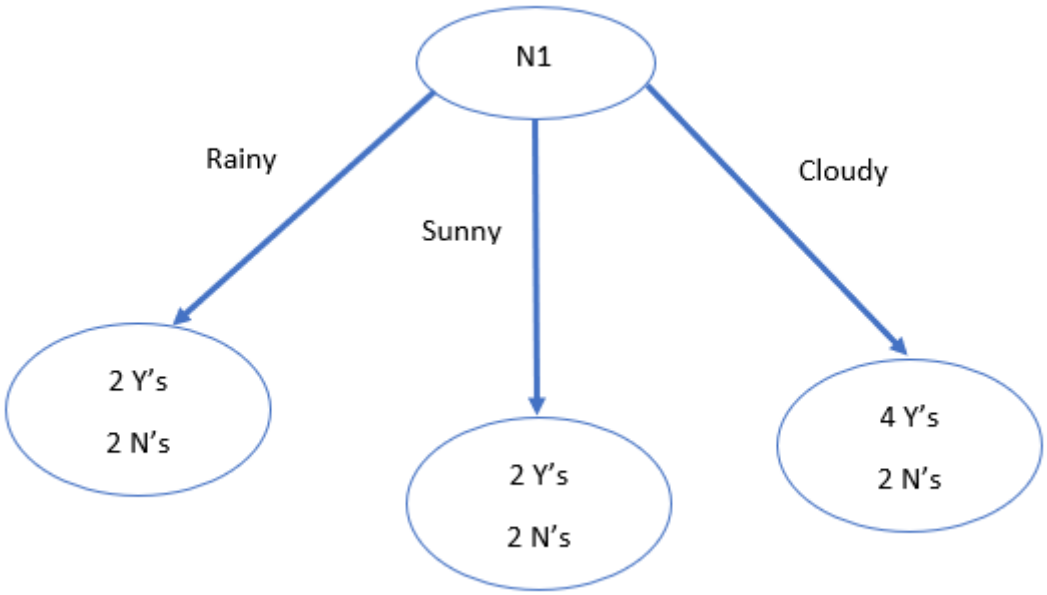


Image By Author

We chose the root node and split the data based on the root node. The picture shows the exact division of the data with corresponding classes.

What next?

As you can see when it is 'Rainy' or 'Sunny' entropy is very high. Because there is the same number of Y's and N's. But Entropy is lower when it is 'cloudy'. So, the class can be Y when it is 'cloudy'. We can split the data again based on N2 and N3. For that, we have to calculate the information gain for each of the subnodes 'Rainy', 'Cloudy', and 'Sunny' to decide which feature comes next and where. As you can see, for each division we have much smaller datasets now.

I am not showing any further calculation here. The intention of this article is to give an

features. But real-world datasets can have a lot more features to deal with. If we have a huge number of features and we want to use all the features to build the tree, the tree will become too large. That may cause overfitting and long computation time.

To handle this, there is a **max_depth** parameter in the decision tree function of the scikit-learn library. If the max depth parameter is large, the tree will be larger. So, we can choose the depth of the tree. **Though you can always use any feature selection method of your choice to select a fewer number of features for the algorithm.**

There is another parameter, **max_features**. The name of the parameter says what it does. You can specify the maximum number of features you want to use for your tree.

Please see the documentation for other parameters.

Major Advantages of Decision Tree algorithm

1. As you can see from the example we worked on to some extent, it gives you a clear visualization of how the prediction happens. So, it is easier to explain to the stakeholders or customers.
2. No feature scaling is required.
3. The non-linearity relationship between the dependent and independent variables does not affect the performance of the decision tree algorithms.
4. A decision tree algorithm can handle outliers automatically.
5. Handles missing values automatically.

Major Disadvantages of Decision Tree Algorithm

1. As discussed earlier, there is a major risk of overfitting. That can also cause high variance which may lead to many errors in the prediction.
2. Decision tree algorithms can be unstable easily. A little bit of noise in the data or adding one extra piece of data may make the whole tree unstable or may build the tree all over again.
3. A decision tree algorithm is not suitable for a large dataset. As discussed in the

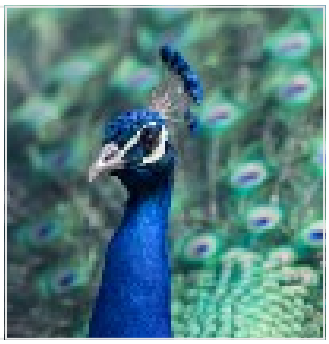
algorithms. So, it is very easy to work now. But knowing the mechanisms behind makes it much better for your decision-making on choosing the right kind of machine learning algorithm for your project.

More Reading

Multiclass Classification Algorithm from Scratch with a Project in Python: Step by Step Guide

This article explains two methods: The gradient descent method and the optimization function method

towardsdatascience.com



Details of Violinplot and Relplot in Seaborn

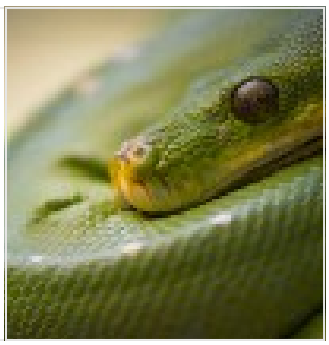
Using Violinplots and Relplots in Seaborn with Full Potential

towardsdatascience.com

A Complete K Mean Clustering Algorithm From Scratch in Python: Step by Step Guide

Also, How to Use K Mean Clustering Algorithm for Dimensionality Reduction of an Image

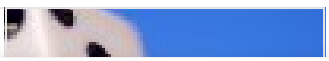
towardsdatascience.com



Stochastic Gradient Descent: Explanation and Complete Implementation from Scratch

Using a Single Perceptron

towardsdatascience.com



Dissecting 1-Way ANOVA and ANCOVA with Examples in R

Differences in Means by Analyzing the Variance

pub.towardsai.net

A Complete Sentiment Analysis Algorithm in Python with Amazon Product Review Data: Step by Step

An NLP Project Using Python's Scikit_learn library

towardsdatascience.com