# Decision Tree Algorithm – A Complete Guide





Anshul Saini — August 29, 2021

Algorithm Beginner Classification Data Science Machine Learning Supervised

This article was published as a part of the Data Science Blogathon

## Introduction

Till now we have learned about linear regression, logistic regression, and they were pretty hard to understand. Let's now start with Decision tree's and I assure you this is probably the easiest algorithm in Machine Learning. There's not much mathematics involved here. Since it is very easy to use and interpret it is one of the most widely used and practical methods used in Machine Learning.

## Contents

# What is a Decision Tree?

It is a tool that has applications spanning several different areas. Decision trees can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves.

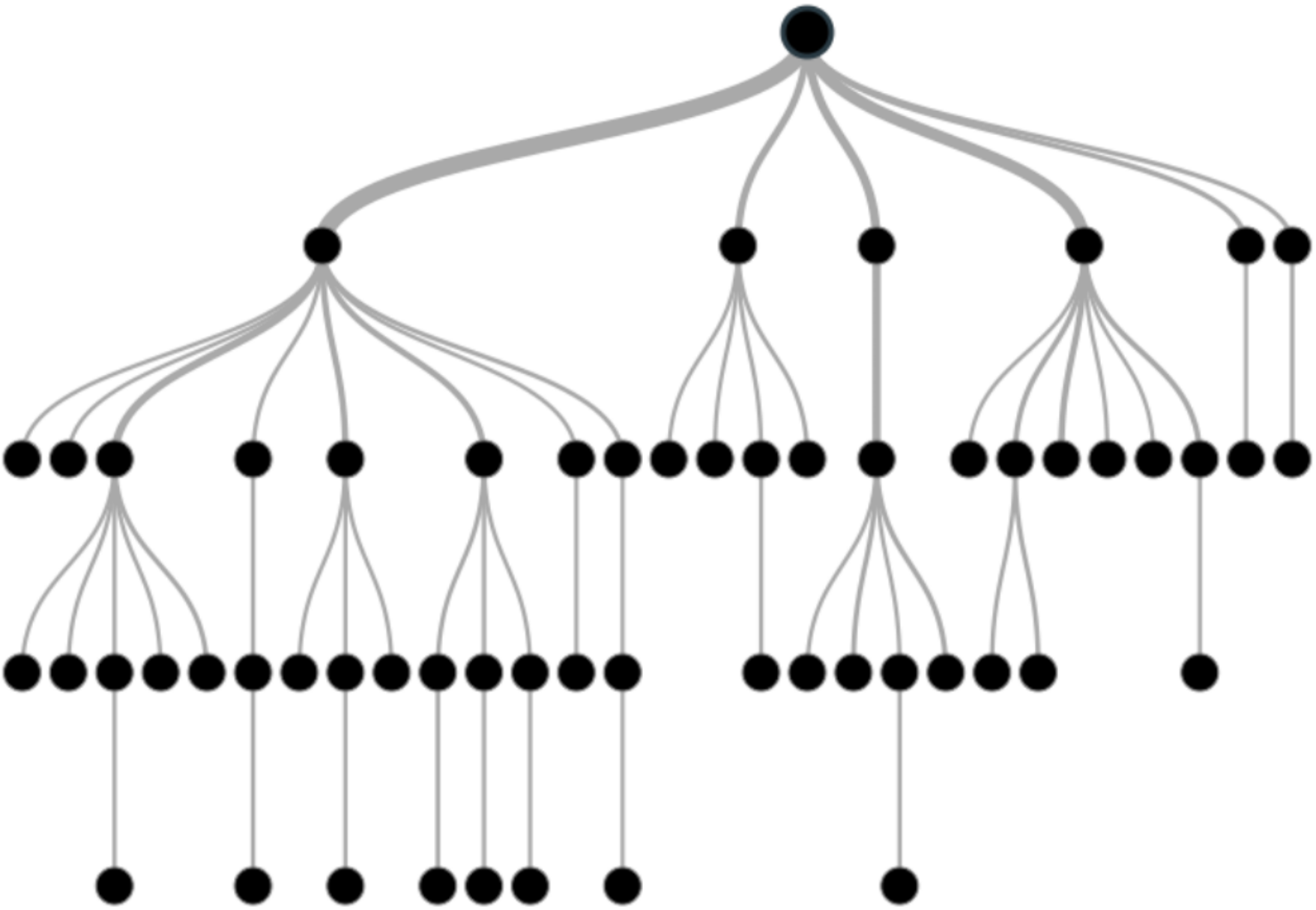


Image 1

Before learning more about decision trees let's get familiar with some of the terminologies.

***Root Nodes*** – It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features.

***Decision Nodes*** – the nodes we get after splitting the root nodes are called Decision Node

***Leaf Nodes*** – the nodes where further splitting is not possible are called leaf nodes or terminal nodes

***Sub-tree*** – just like a small portion of a graph is called sub-graph similarly a sub-section of this decision tree is called sub-tree.

# Decision Tree Algorithm – A Complete Guide

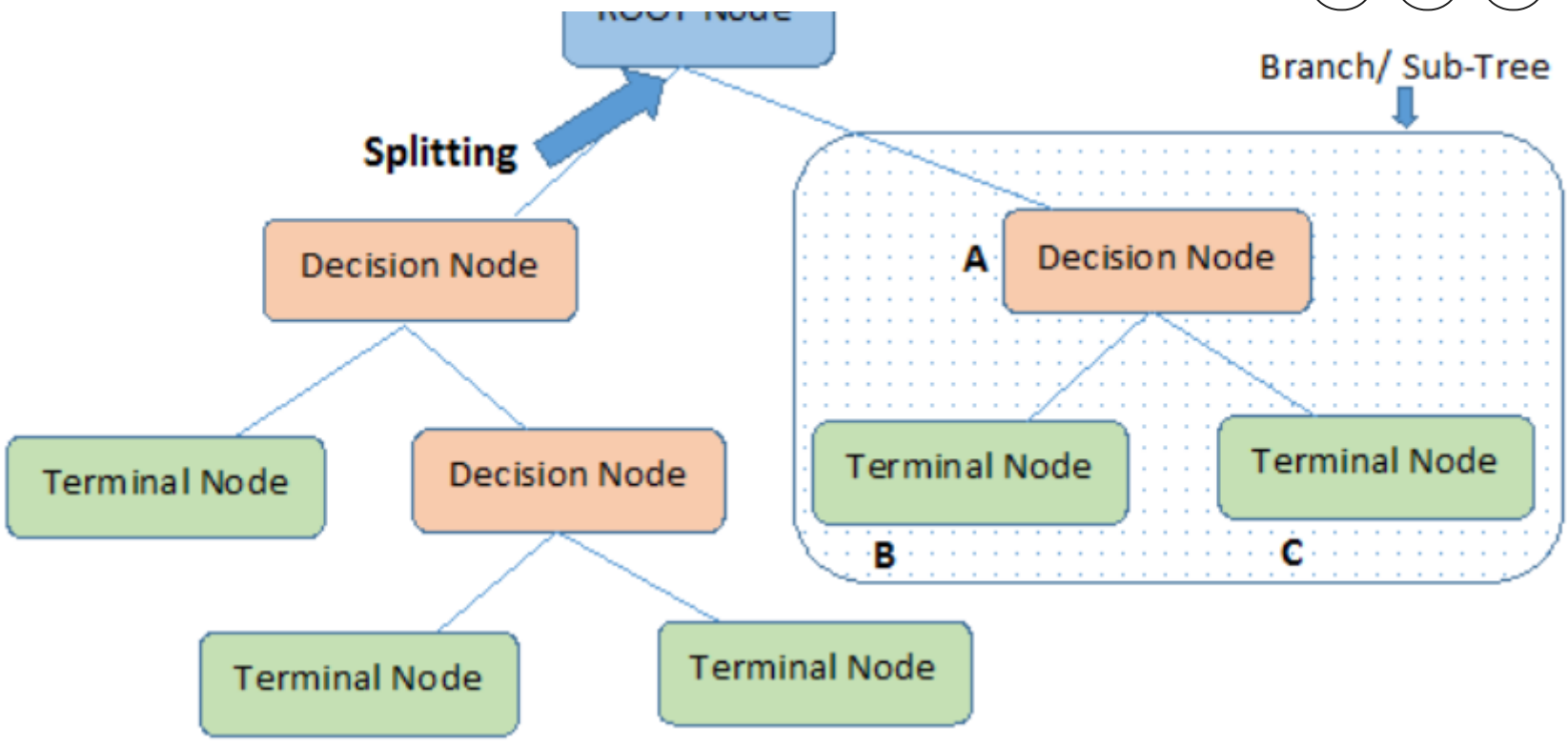



Image 2

## Example of a decision tree

Let's understand decision trees with the help of an example.

| Day | Weather | Temperature | Humidity | Wind | Play? |
|---|---|---|---|---|---|
| 1 | Sunny | Hot | High | Weak | No |
| 2 | Cloudy | Hot | High | Weak | Yes |
| 3 | Sunny | Mild | Normal | Strong | Yes |
| 4 | Cloudy | Mild | High | Strong | Yes |
| 5 | Rainy | Mild | High | Strong | No |
| 6 | Rainy | Cool | Normal | Strong | No |
| 7 | Rainy | Mild | High | Weak | Yes |
| 8 | Sunny | Hot | High | Strong | No |
| 9 | Cloudy | Hot | Normal | Weak | Yes |
| 10 | Rainy | Mild | High | Strong | No |

Image 3

Decision trees are upside down which means the root is at the top and then this root is split into various several nodes. Decision trees are nothing but a bunch of if-else statements in layman terms. It checks if the condition is true and if it is then it goes to the next node attached to that decision.

In the below diagram the tree will first ask what is the weather? Is it sunny, cloudy, or rainy? If yes then it will go to the next feature which is humidity and wind. It will again check if there is a strong wind or weak, if it's a weak wind and it's rainy then the person may go and play.

# Decision Tree Algorithm – A Complete Guide

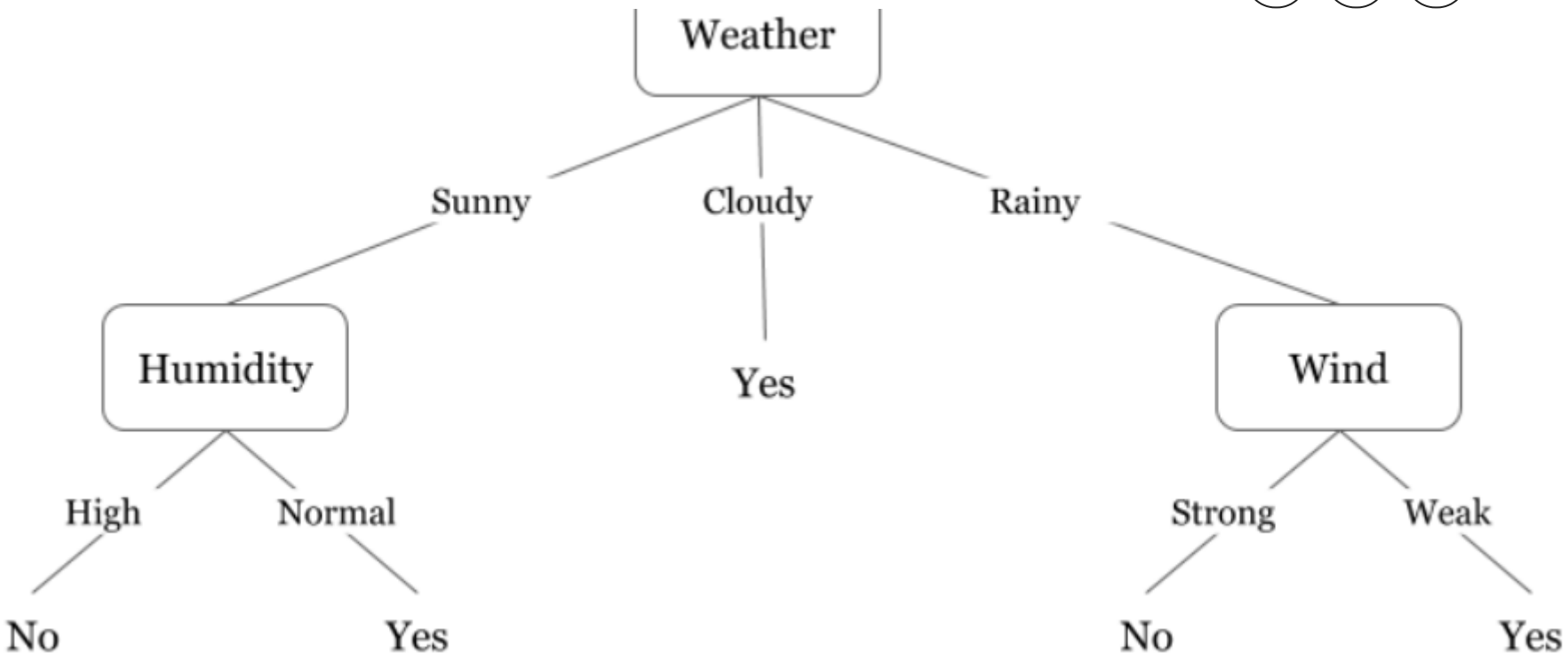



Image 4

Did you notice anything in the above flowchart? We see that if the *weather is cloudy* then we must go to play. Why didn't it split more? Why did it stop there?

To answer this question, we need to know about few more concepts like entropy, information gain, and Gini index. But in simple terms, I can say here that the output for the training dataset is always yes for cloudy weather, since there is no disorderliness here we don't need to split the node further.

The goal of machine learning is to decrease uncertainty or disorders from the dataset and for this, we use decision trees.

Now you must be thinking how do I know what should be the root node? what should be the decision node? when should I stop splitting? To decide this, there is a metric called "Entropy" which is the amount of uncertainty in the dataset.

## Entropy

Entropy is nothing but the uncertainty in our dataset or measure of disorder. Let me try to explain this with the help of an example.

Suppose you have a group of friends who decides which movie they can watch together on Sunday. There are 2 choices for movies, one is ***"Lucy"*** and the second is ***"Titanic"*** and now everyone has to tell their choice. After everyone gives their answer we see that *"Lucy" gets 4 votes* and *"Titanic" gets 5 votes*. Which movie do we watch now? Isn't it hard to choose 1 movie now because the votes for both the movies are somewhat equal.

This is exactly what we call disorderness, there is an equal number of votes for both the movies, and we can't really decide which movie we should watch. It would have been much easier if the votes for "Lucy" were 8 and for "Titanic" it was 2. Here we could easily say that the majority of votes are for "Lucy" hence everyone will be watching this movie.

In a decision tree, the output is mostly "yes" or "no"

The formula for Entropy is shown below:

$$E(S) = -p_{(+)}\log p_{(+)} - p_{(-)}\log p_{(-)}$$

S is the subset of the training example

# How do Decision Trees use Entropy?

Now we know what entropy is and what is its formula, Next, we need to know that how exactly does it work in this algorithm.

Entropy basically measures the impurity of a node. Impurity is the degree of randomness; it tells how random our data is. A **pure sub-split** means that either you should be getting "yes", or you should be getting "no".

Suppose a *feature* has 8 "yes" and 4 "no" initially, after the first split the left node *gets 5 'yes' and 2 'no'* whereas right node *gets 3 'yes' and 2 'no'.*

We see here the split is not pure, why? Because we can still see some negative classes in both the nodes. In order to make a decision tree, we need to calculate the impurity of each split, and when the purity is 100%, we make it as a leaf node.

To check the impurity of feature 2 and feature 3 we will take the help for Entropy formula.

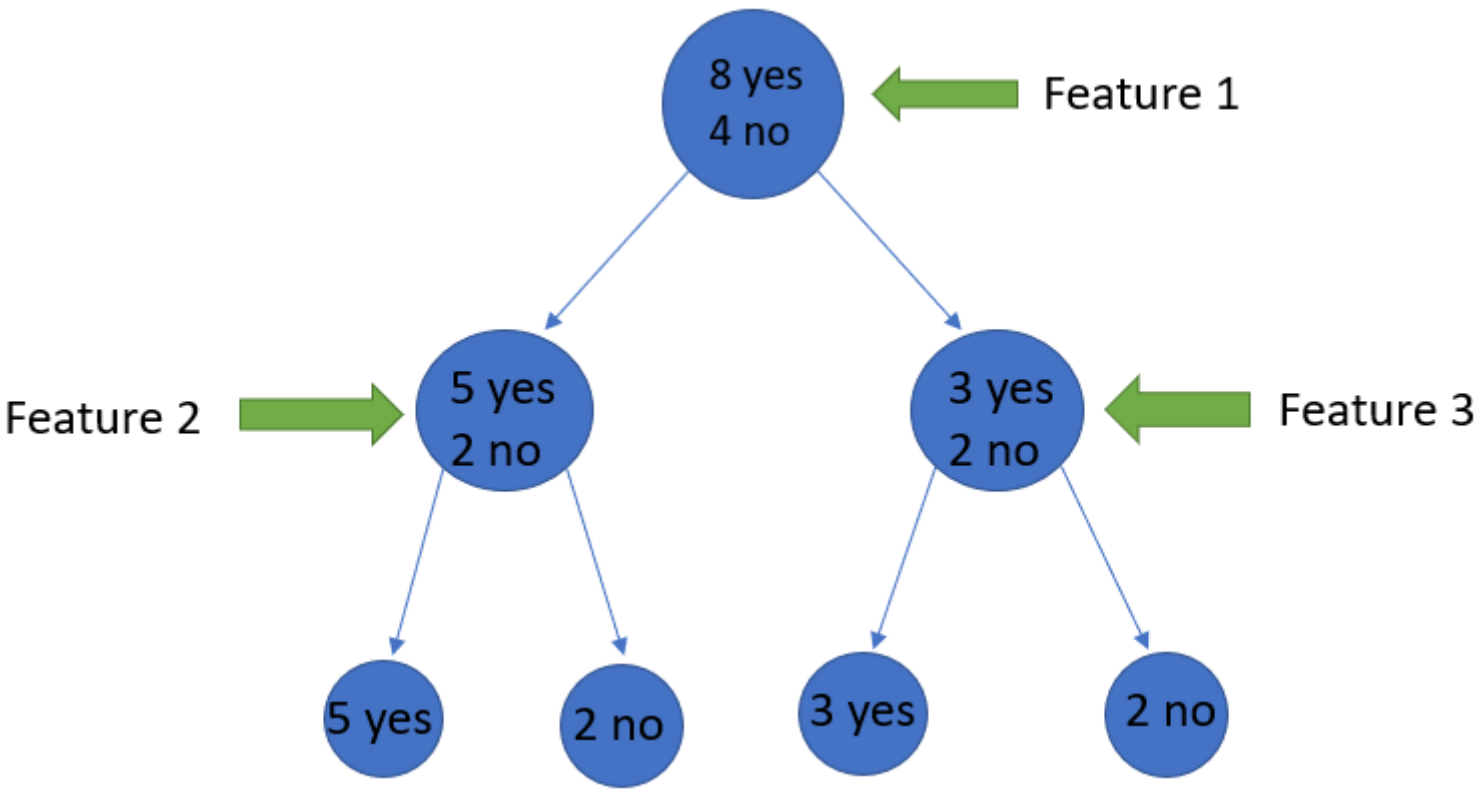



Image Source: Author

$$\Rightarrow \quad -\left(\frac{5}{7}\right)log_2\left(\frac{5}{7}\right) - \left(\frac{2}{7}\right)log_2\left(\frac{2}{7}\right)$$

$$\Rightarrow \quad -(0.71 * -0.49) - (0.28 * -1.83)$$

$$\Rightarrow \quad -(-0.34) - (-0.51)$$

$$\Rightarrow \quad 0.34 + 0.51$$

$$\Rightarrow \quad 0.85$$

For feature 3,

## Decision Tree Algorithm – A Complete Guide

$$\Rightarrow -\left(\frac{}{5}\right)log_2\left(\frac{}{5}\right) - \left(\frac{}{5}\right)log_2\left(\frac{}{5}\right)$$

$$\Rightarrow -(0.6 * -0.73) - (0.4 * -1.32)$$

$$\Rightarrow -(-0.438) - (-0.528)$$

$$\Rightarrow 0.438 + 0.528$$

$$\Rightarrow 0.966$$

We can clearly see from the tree itself that left node has low entropy or more purity than right node since left node has a greater number of “yes” and it is easy to decide here.

Always remember that the higher the Entropy, the lower will be the purity and the higher will be the impurity.

As mentioned earlier the goal of machine learning is to decrease the uncertainty or impurity in the dataset, here by using the entropy we are getting the impurity of a particular node, we don’t know if the parent entropy or the entropy of a particular node has decreased or not.

For this, we bring a new metric called “Information gain” which tells us how much the parent entropy has decreased after splitting it with some feature.

# Information Gain

Information gain measures the reduction of uncertainty given some feature and it is also a deciding factor for which attribute should be selected as a decision node or root node.

$$Information\ Gain = E(Y) - E(Y|X)$$

It is just entropy of the full dataset – entropy of the dataset given some feature.

To understand this better let’s consider an example:
Suppose our entire population has a total of 30 instances. The dataset is to predict whether the person will go to the gym or not. Let’s say 16 people go to the gym and 14 people don’t

Now we have two features to predict whether he/she will go to the gym or not.

Feature 1 is **“Energy”** which takes two values *“high” and “low”*

Feature 2 is **“Motivation”** which takes 3 values *“No motivation”, “Neutral” and “Highly motivated”.*

Let’s see how our decision tree will be made using these 2 features. We’ll use information gain to decide which feature should be the root node and which feature should be placed after the split.

# Decision Tree Algorithm – A Complete Guide

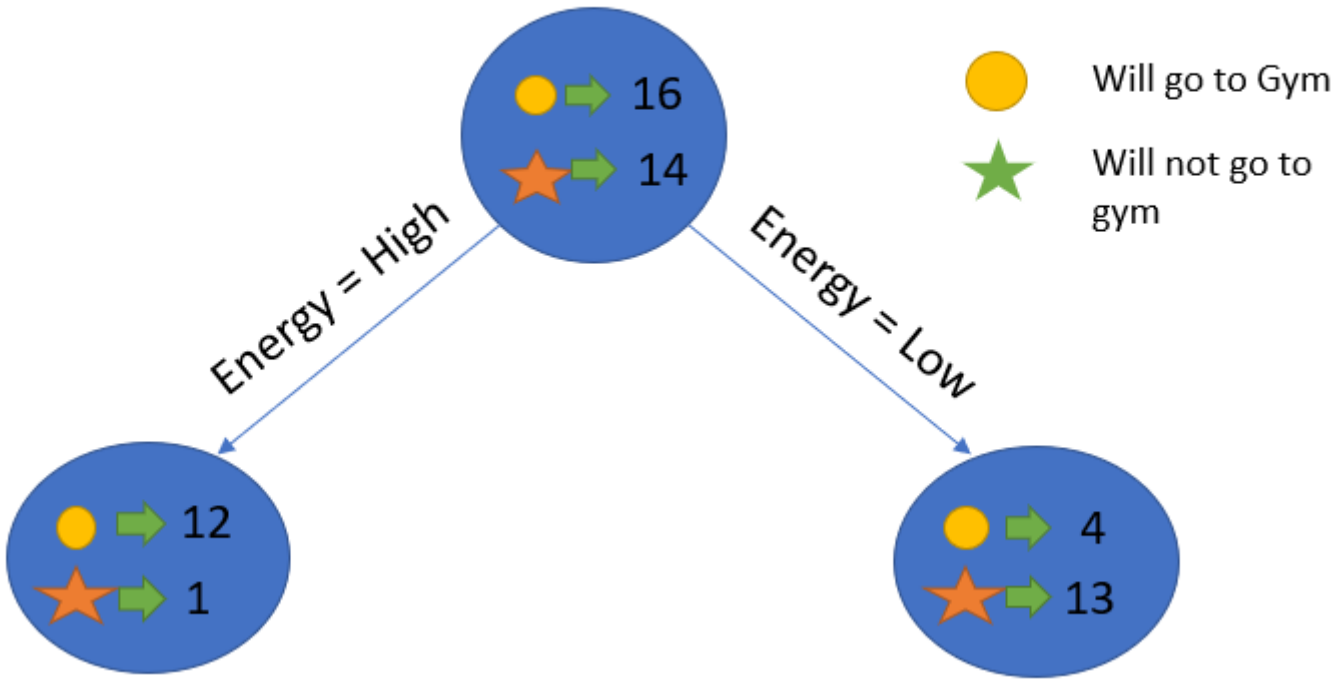



Image Source: Author

Let's calculate the entropy:

$$E(Parent) = -\left(\frac{16}{30}\right)\log_2\left(\frac{16}{30}\right) - \left(\frac{14}{30}\right)\log_2\left(\frac{14}{30}\right) \approx 0.99$$

$$E(Parent|Energy = "high") = -\left(\frac{12}{13}\right)\log_2\left(\frac{12}{13}\right) - \left(\frac{1}{13}\right)\log_2\left(\frac{1}{13}\right) \approx 0.39$$

$$E(Parent|Energy = "low") = -\left(\frac{4}{17}\right)\log_2\left(\frac{4}{17}\right) - \left(\frac{13}{17}\right)\log_2\left(\frac{13}{17}\right) \approx 0.79$$

To see the weighted average of entropy of each node we will do as follows:

$$E(Parent|Energy) = \frac{13}{30} * 0.39 + \frac{17}{30} * 0.79 = 0.62$$

Now we have the value of E(Parent) and E(Parent|Energy), information gain will be:

$$Information\ Gain = E(parent) - E(parent|energy)$$
$$= 0.99 - 0.62$$
$$= 0.37$$

Our parent entropy was near 0.99 and after looking at this value of information gain, we can say that the entropy of the dataset will decrease by 0.37 if we make "Energy" as our root node.

Similarly, we will do this with the other feature "Motivation" and calculate its information gain.

# Decision Tree Algorithm – A Complete Guide

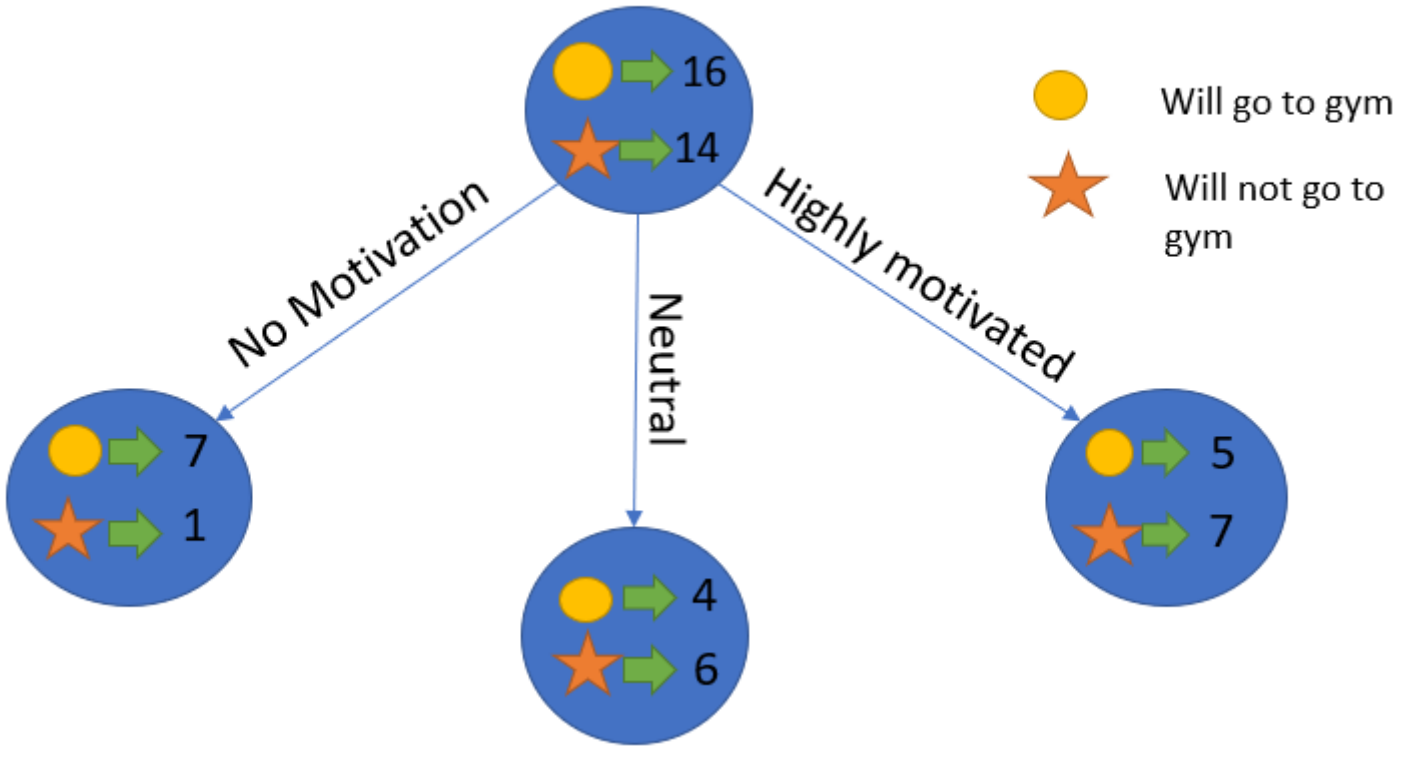



Image Source: Author

Let's calculate the entropy here:

$$E(Parent) = 0.99$$

$$E\Big(Parent|Motivation = "No\ motivation"\Big) = -\Big(\frac{7}{8}\Big)log_2\Big(\frac{7}{8}\Big) - \frac{1}{8}\log_2\Big(\frac{1}{8}\Big) = 0.54$$

$$E\big(Parent|Motivation = "Neutral"\big) = -\Big(\frac{4}{10}\Big)log_2\Big(\frac{4}{10}\Big) - (\frac{6}{10})\log_2\Big(\frac{6}{10}\Big) = 0.97$$

$$E\big(Parent|Motivation = "Highly\ motivated"\big) = -\Big(\frac{5}{12}\Big)log_2\Big(\frac{5}{12}\Big) - (\frac{7}{12})\log_2\Big(\frac{7}{12}\Big) = 0.98$$

To see the weighted average of entropy of each node we will do as follows:

$$E(Parent|Motivation) = \frac{8}{30} * 0.54 + \frac{10}{30} * 0.97 + \frac{12}{30} * 0.98 = 0.86$$

Now we have the value of E(Parent) and E(Parent|Motivation), information gain will be:

$$Information\ Gain = E(Parent) - E(Parent|Motivation)$$
$$= 0.99 - 0.86$$
$$= 0.13$$

We now see that the "Energy" feature gives more reduction which is 0.37 than the "Motivation" feature. Hence we will select the feature which has the highest information gain and then split the node based on that feature.

# Decision Tree Algorithm – A Complete Guide

Image Source: Author

In this example “Energy” will be our root node and we’ll do the same for sub-nodes. Here we can see that when the energy is “high” the entropy is low and hence we can say a person will definitely go to the gym if he has high energy, but what if the energy is low? We will again split the node based on the new feature which is “Motivation”.

## When to stop splitting?

You must be asking this question to yourself that when do we stop growing our tree? Usually, real-world datasets have a large number of features, which will result in a large number of splits, which in turn gives a huge tree. Such trees take time to build and can lead to overfitting. That means the tree will give very good accuracy on the training dataset but will give bad accuracy in test data.

There are many ways to tackle this problem through hyperparameter tuning. We can set the maximum depth of our decision tree using the ***max_depth*** parameter. The more the value of ***max_depth***, the more complex your tree will be. The training error will off-course decrease if we increase the ***max_depth*** value but when our test data comes into the picture, we will get a very bad accuracy. Hence you need a value that will not overfit as well as underfit our data and for this, you can use GridSearchCV.

Another way is to set the minimum number of samples for each spilt. It is denoted by ***min_samples_split***. Here we specify the minimum number of samples required to do a spilt. For example, we can use a minimum of 10 samples to reach a decision. That means if a node has less than 10 samples then using this parameter, we can stop the further splitting of this node and make it a leaf node.

There are more hyperparameters such as :

***min_samples_leaf*** – represents the minimum number of samples required to be in the leaf node. The more you increase the number, the more is the possibility of overfitting.

***max_features*** – it helps us decide what number of features to consider when looking for the best split.

To read more about these hyperparameters you can read it here.

## Pruning

It is another method that can help us avoid overfitting. It helps in improving the performance of the tree by cutting the nodes or sub-nodes which are not significant. It removes the branches which have very low importance.

**while growing** the tree.

(ii) **Post-pruning** – once our **tree is built to its depth**, we can start pruning the nodes based on their significance.

# Endnotes

To summarize, in this article we learned about decision trees. On what basis the tree splits the nodes and how to can stop overfitting. why linear regression doesn't work in the case of classification problems.

In the next article, I will explain Random forests, which is again a new technique to avoid overfitting.
To check out the full implementation of decision trees please refer to my Github repository.

Let me know if you have any queries in the comments below.

# About the Author

I am an undergraduate student currently in my last year majoring in Statistics (Bachelors of Statistics) and have a strong interest in the field of data science, machine learning, and artificial intelligence. I enjoy diving into data to discover trends and other valuable insights about the data. I am constantly learning and motivated to try new things.
I am open to collaboration and work.
For any **doubt and queries,** feel free to contact me on Email

Connect with me on LinkedIn and Twitter

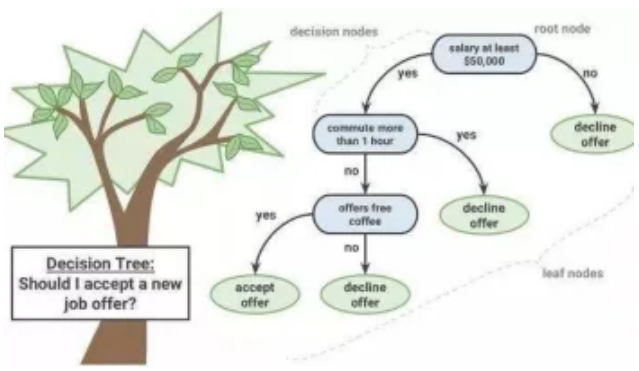
**The media shown in this article are not owned by Analytics Vidhya and are used at the Author's discretion.**

## Image Sources

1. Image 1 – https://wiki.pathmind.com/decision-tree
2. Image 2 – https://wiki.pathmind.com/decision-tree
3. Image 3 – www.hackerearth.com
4. Image 4 – www.hackerearth.com

Related


Tree Based Algorithms: A Complete Tutorial from Scratch (in R & Python)




25 Questions to Test Your Skills on Decision Trees

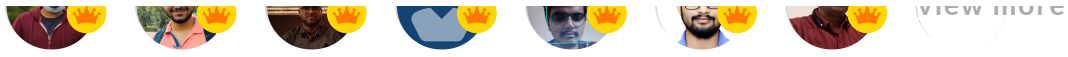
Let's Solve Overfitting! Quick Guide to Cost Complexity Pruning of Decision Trees

blogathon data science decision tree machine learning

## About the Author

Anshul Saini

# Decision Tree Algorithm – A Complete Guide






**Previous Post**

ML-trained Predictive model with a Django API

**Next Post**

Your Guide to Object Detection with Detectron2 in PyTorch


## Leave a Reply

**Your email address will not be published. Required fields are marked ***

Comment

Name*

Email*

Website

☑ Notify me of follow-up comments by email.

☑ Notify me of new posts by email.

Submit

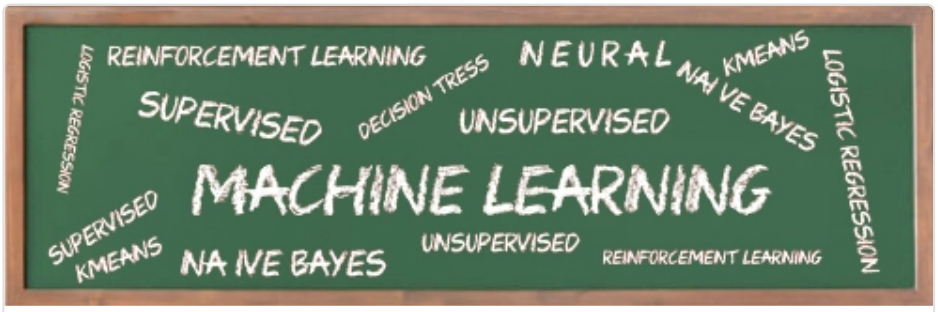
## Top Resources

# Decision Tree Algorithm – A Complete Guide

Harika Bonthu - AUG 21, 2021

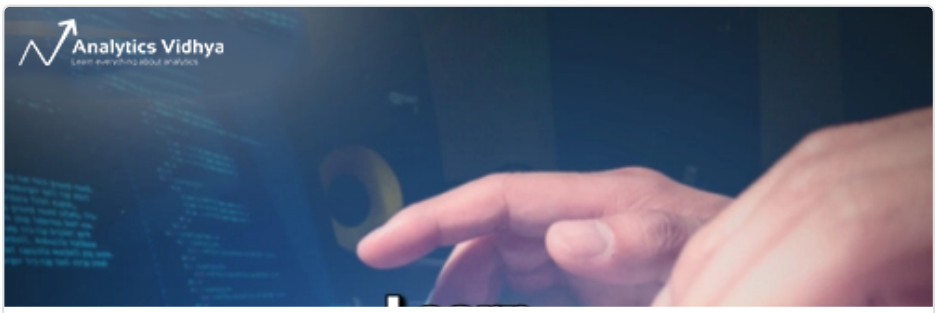
Techniques..

Saurav Kaushik - FEB 05, 2017




Commonly used Machine Learning Algorithms (with Python and R Codes)

Sunil Ray - SEP 09, 2017




Understanding Support Vector Machine(SVM) algorithm from examples (along with code)

Sunil Ray - SEP 13, 2017