Double-click (or enter) to edit

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.

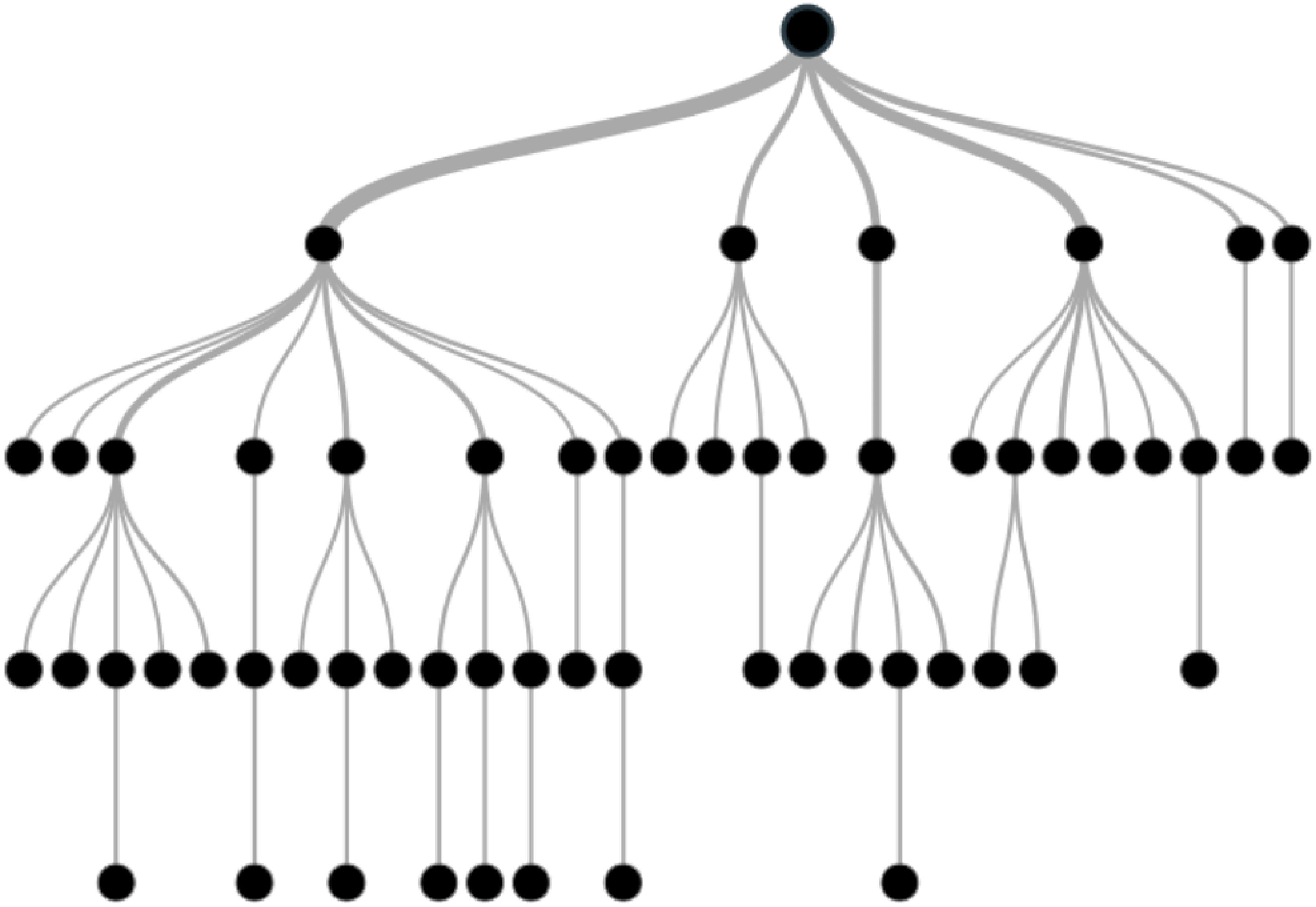
Contents

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.

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

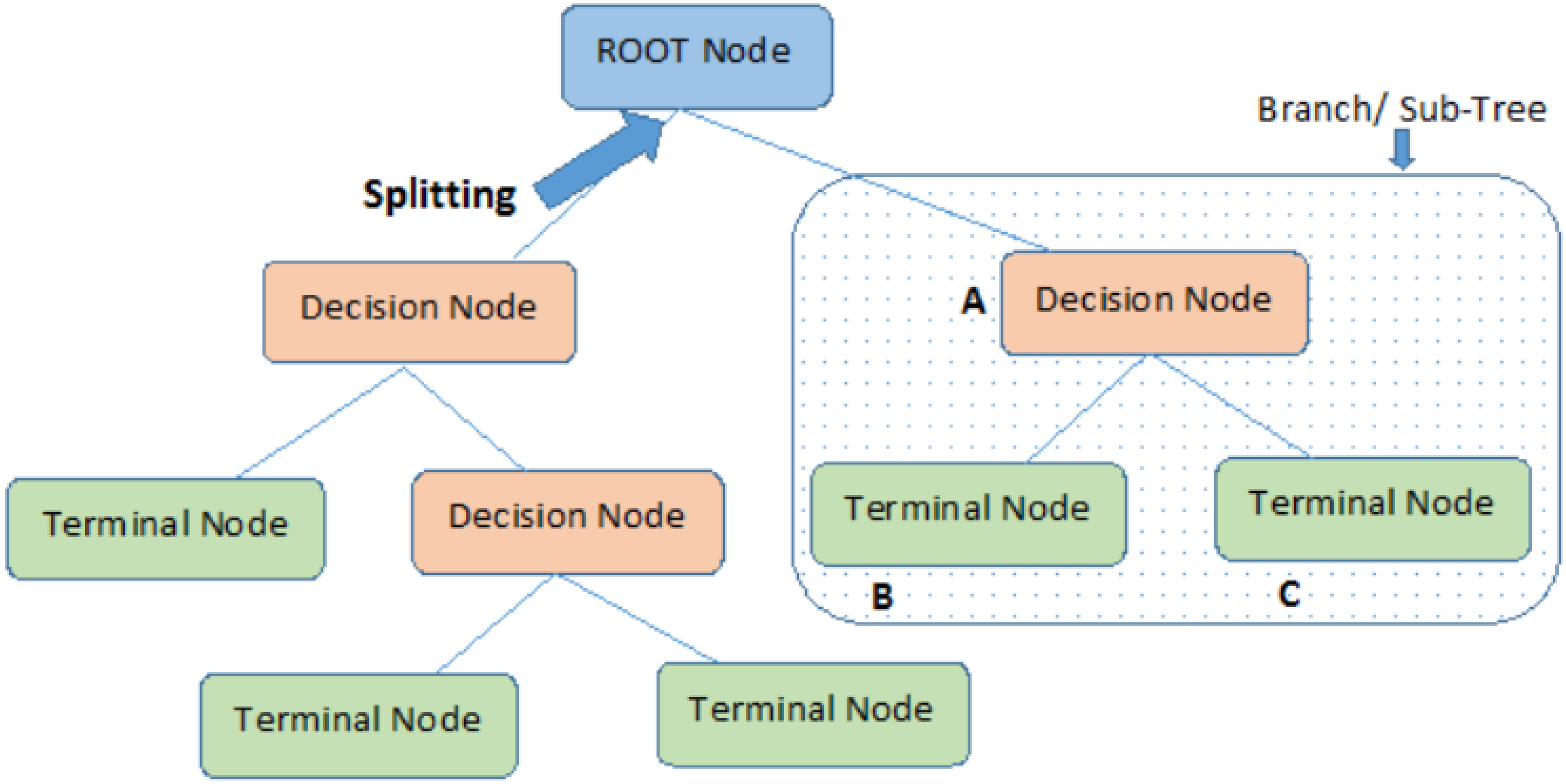
Root Nodes – It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features.

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

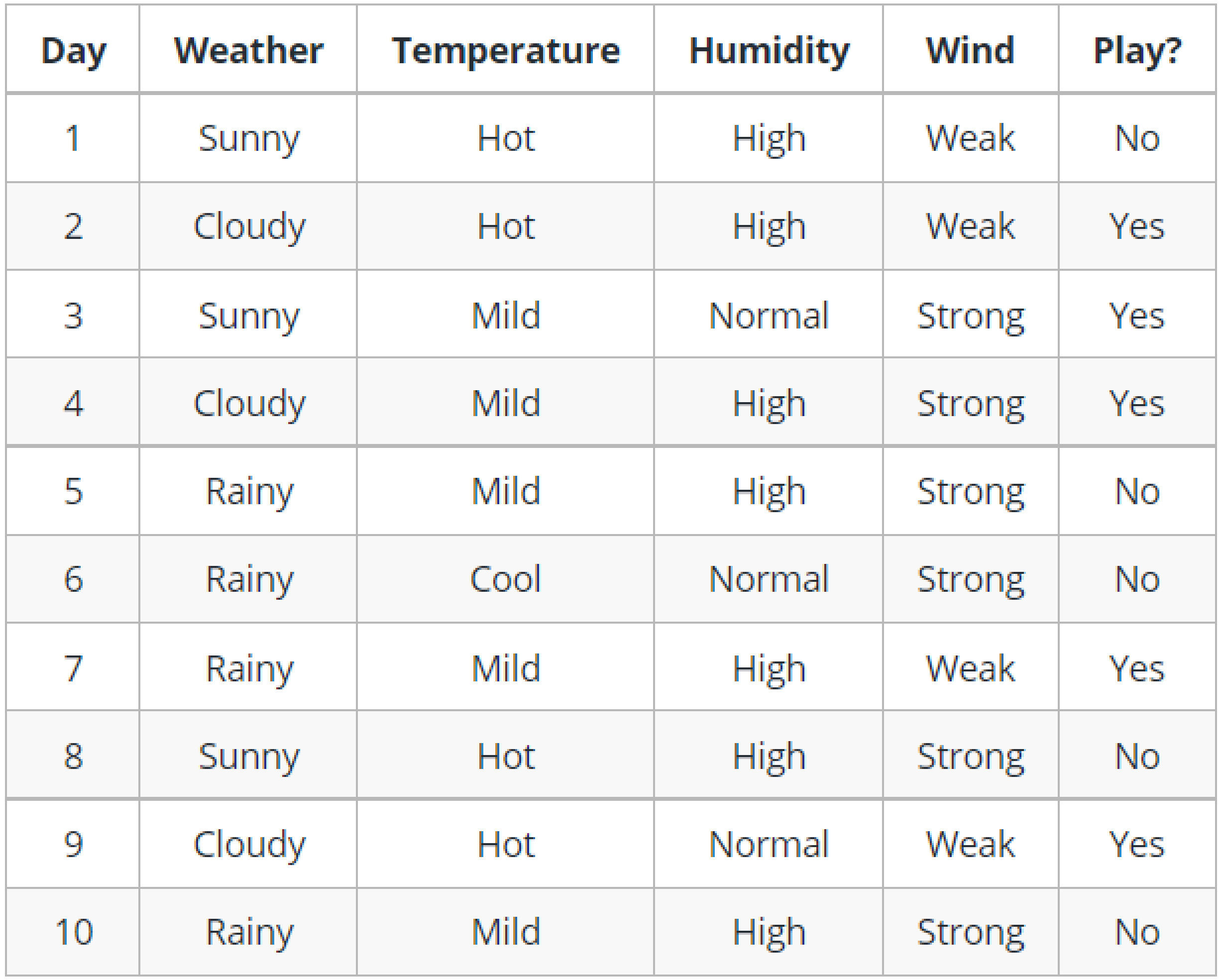
Sub-tree – just like a small portion of a graph is called sub-graph similarly a sub-section of this decision tree is called sub-tree.

Pruning – is nothing but cutting down some nodes to stop overfitting.



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

Decision trees are upside down which means the root is at the top and then this root is split into various several nodes.

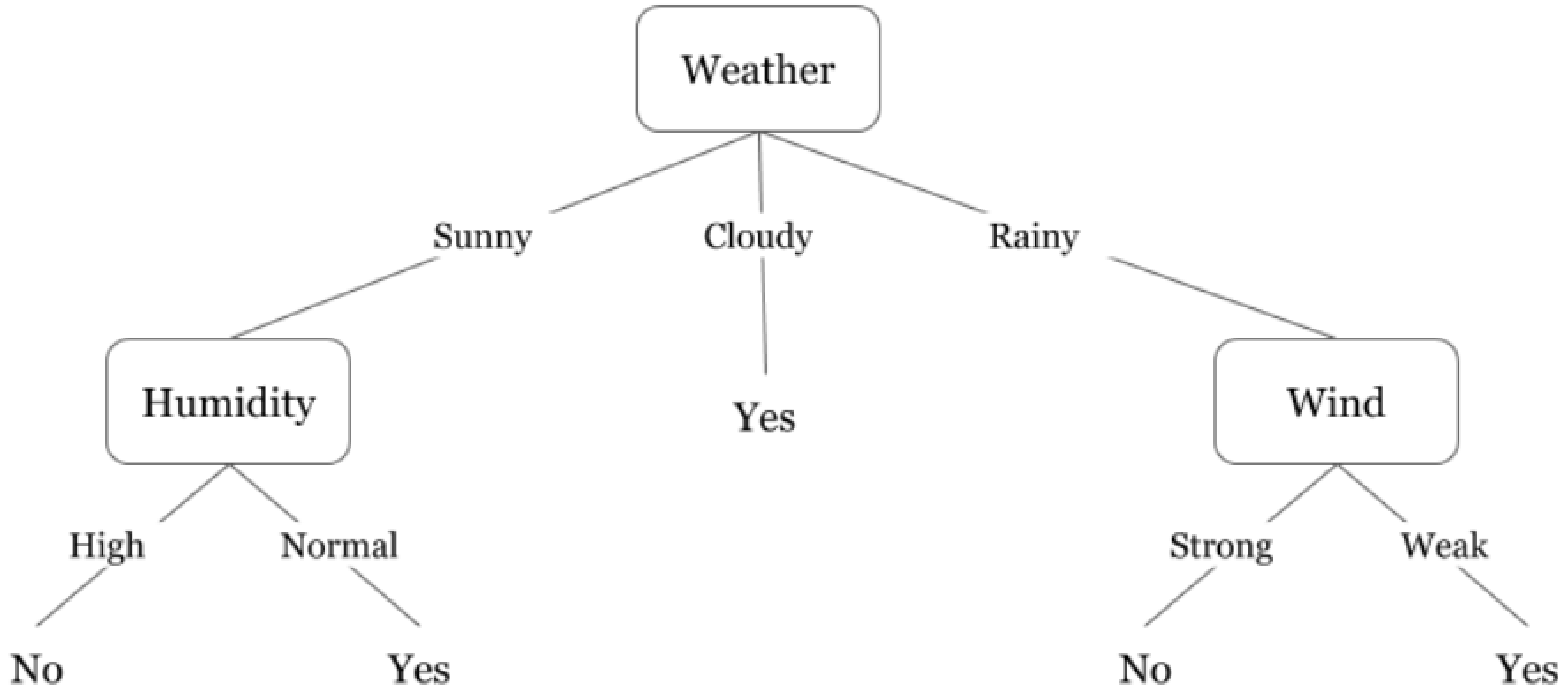
Decision trees are nothing but a bunch of if-else statements in layman terms. It checks if the condition is true and if it is then it goes to the next node attached to that decision.

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.



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.

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

The formula for Entropy is shown below:

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

Here p+ is the probability of positive class

p– is the probability of negative class

S is the subset of the training example

How do Decision Trees use Entropy?

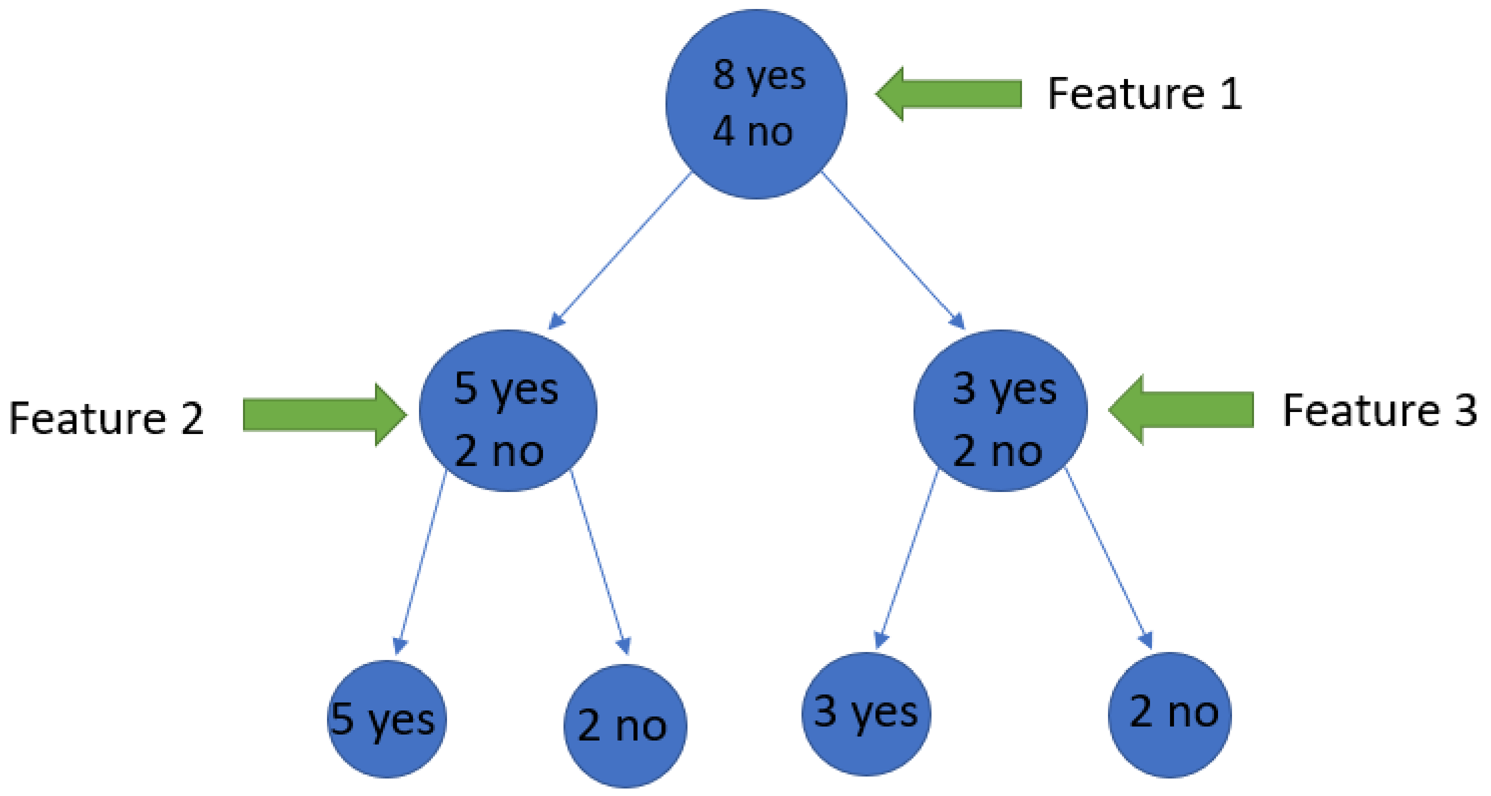
Now we know what entropy is and what is its formula, Next, we need to know that how exactly does it work in this algorithm.

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.


8 yes
4 no
Feature 1
Feature 2
5 yes
2 no
3 yes
2 no
Feature 3
5 yes
2 no
3 yes
2 no

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

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

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

$$\Rightarrow\ 0.34+0.51$$

$$\Rightarrow\ 0.85$$

$$\Rightarrow\ -\left(\frac{3}{5}\right)log_2\left(\frac{3}{5}\right)\ -\ \left(\frac{2}{5}\right)log_2\left(\frac{2}{5}\right)$$

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

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

$$\Rightarrow\ 0.438+0.528$$

$$\Rightarrow\ 0.966$$

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

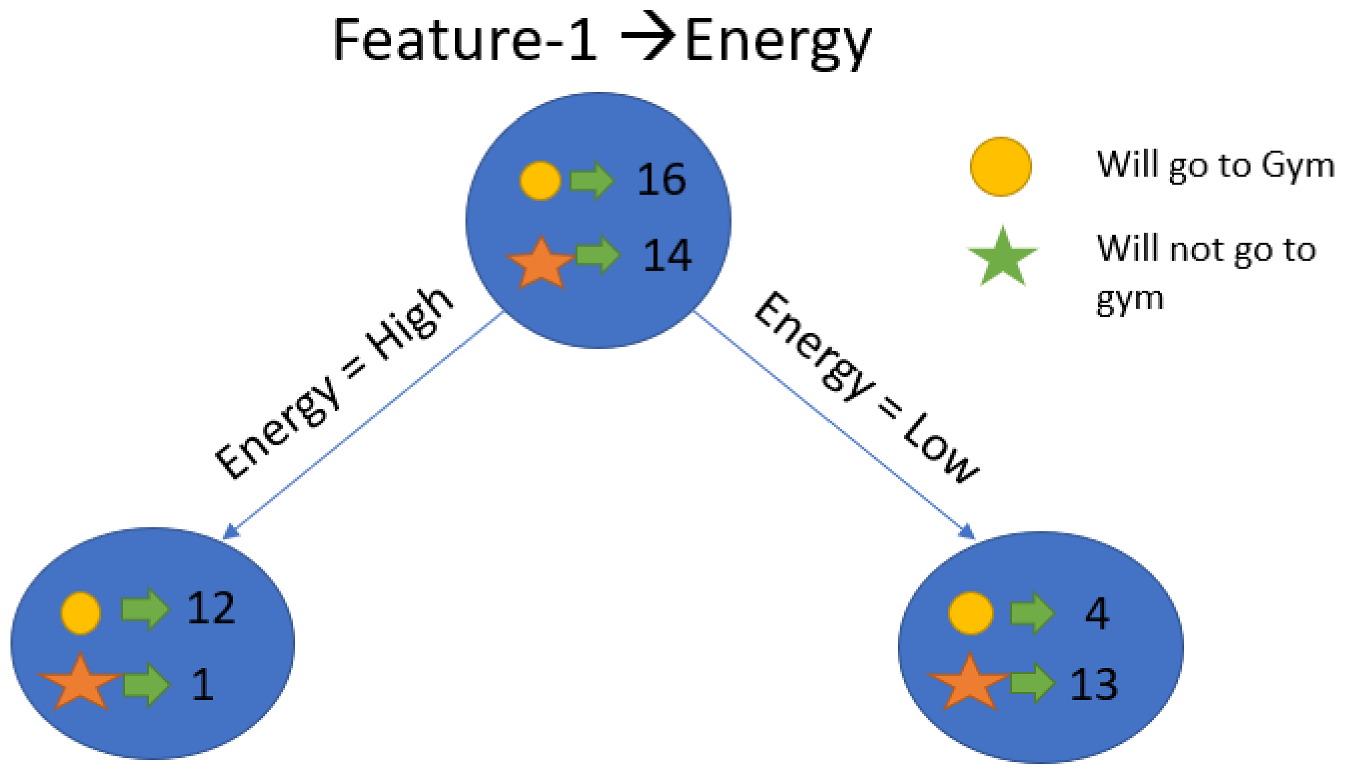
Now we have two features to predict whether he/she will go to the gym or not.

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.



HOw to Calculate 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:

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

$$\begin{aligned} Information\ Gain &= E(parent) - E(parent|energy) \\ &= 0.99 - 0.62 \\ &= 0.37 \end{aligned}$$

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.