**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**

1. What is a Decision Tree?

2. Example of a Decision Tree

3. Entropy

4. Information Gain

5. When to stop Splitting?

6. How to stop overfitting?

* max\_depth
* min\_samples\_split
* min\_samples\_leaf
* max\_features

7. Pruning

* Post-pruning
* Pre-pruning

8. Endnotes

**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.

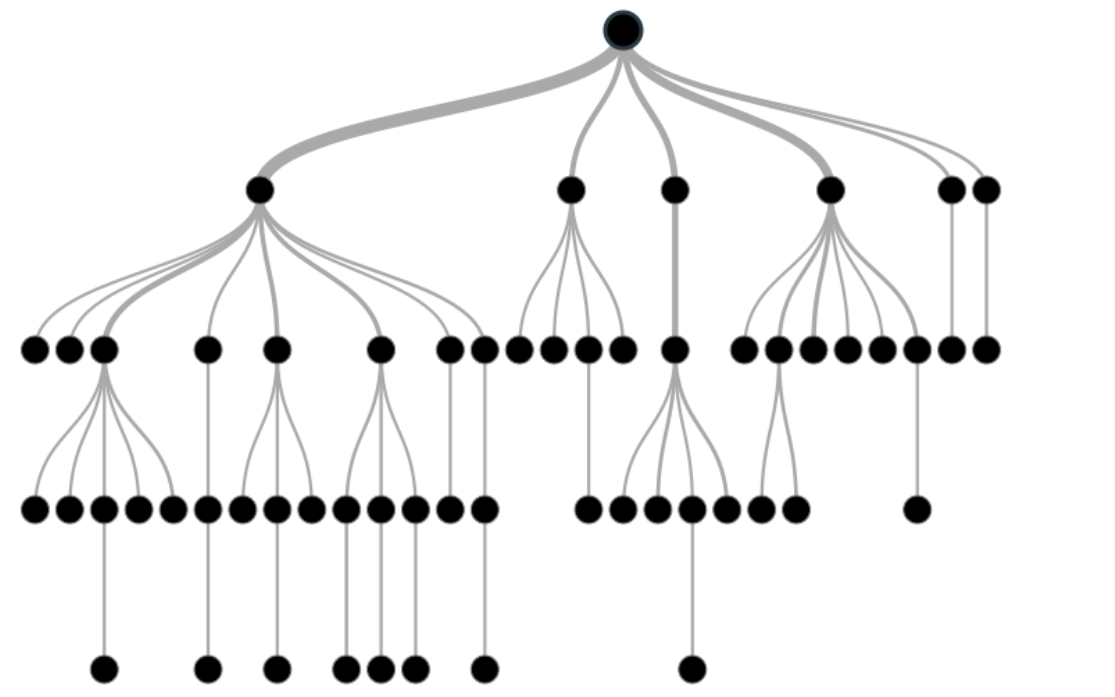


Image 1

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.

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

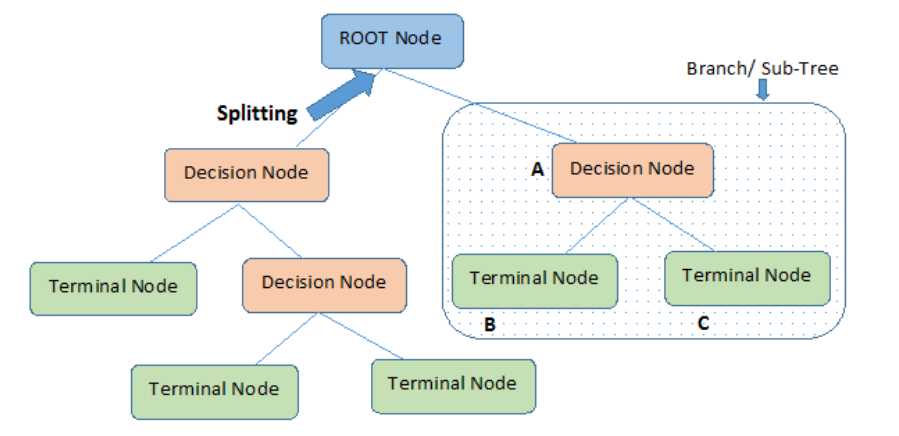


Image 2

**Example of a decision tree**

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

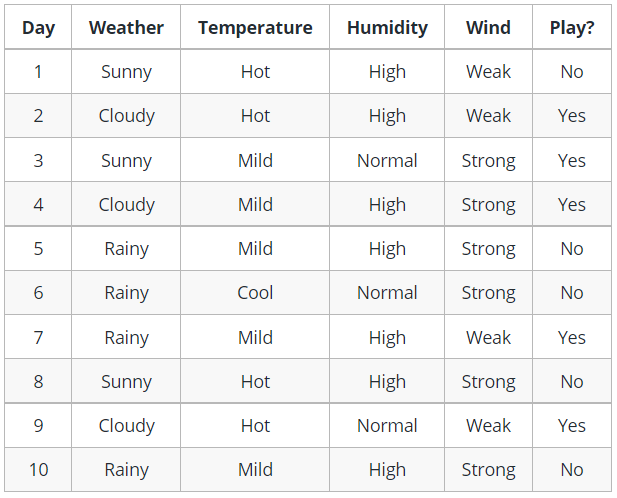


Image 3

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.

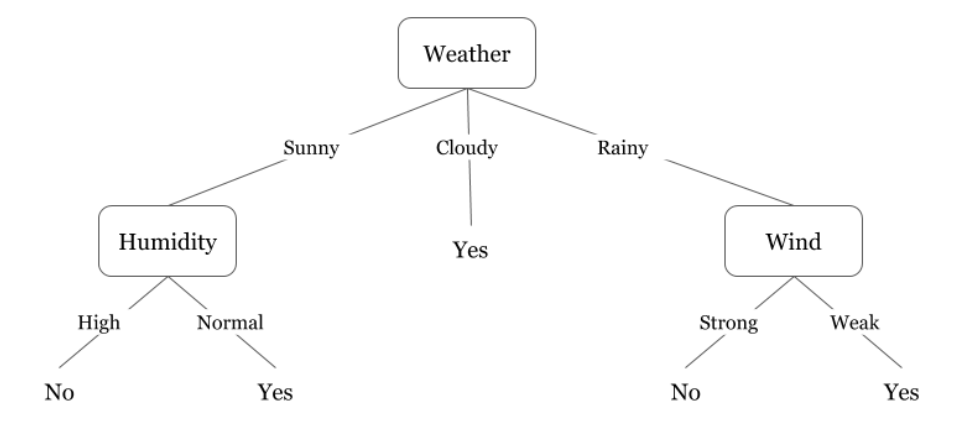


Image 4

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**

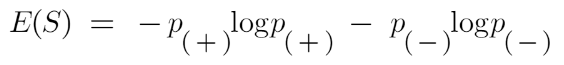
Entropy is nothing but the uncertainty in our dataset or measure of disorder. Let me try to explain this with the help of an example.

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:



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.

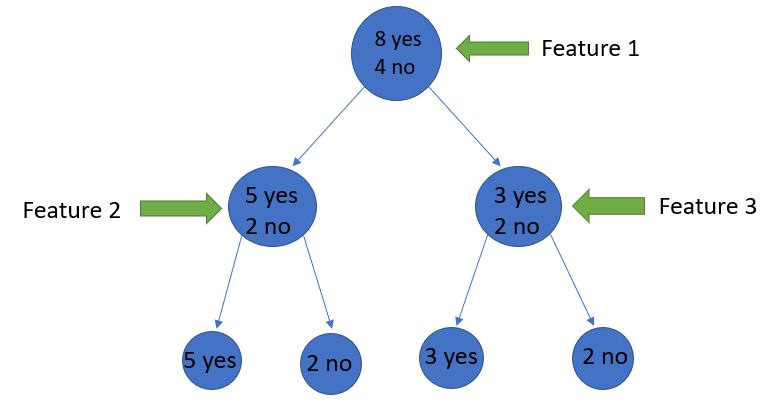
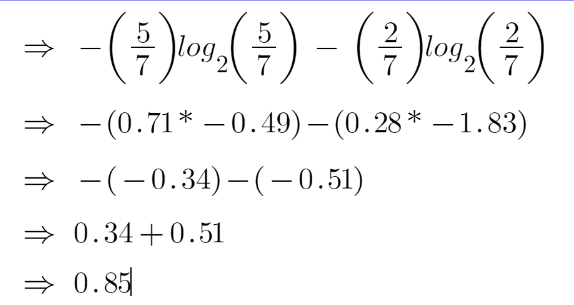
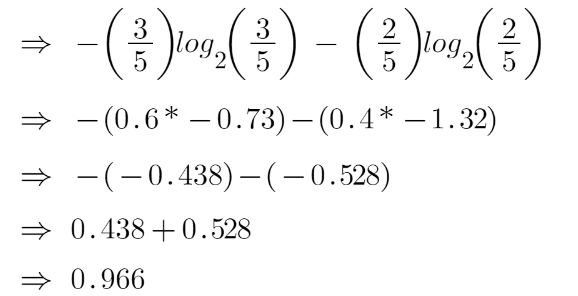


Image Source: Author



For feature 3,



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 Decision tree algorithm

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.

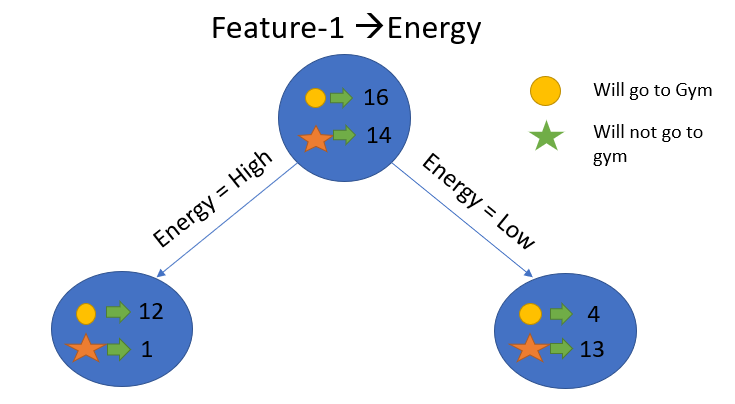
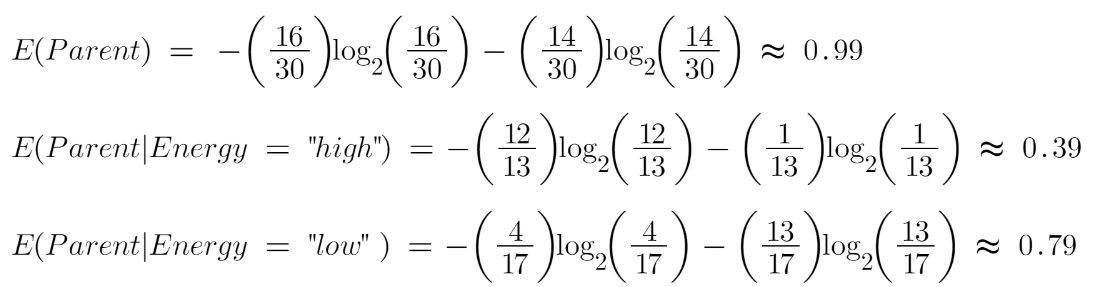
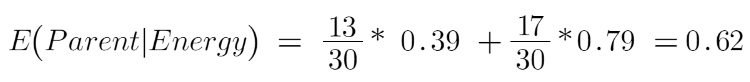


Image Source: Author

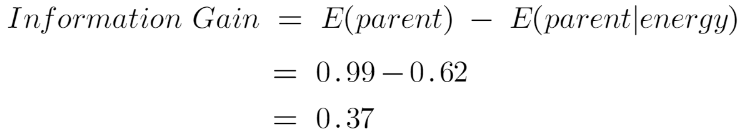
Let’s calculate the entropy:



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



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



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.

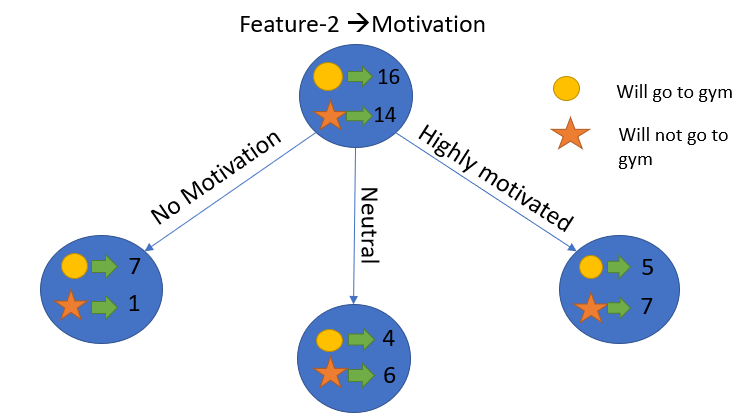
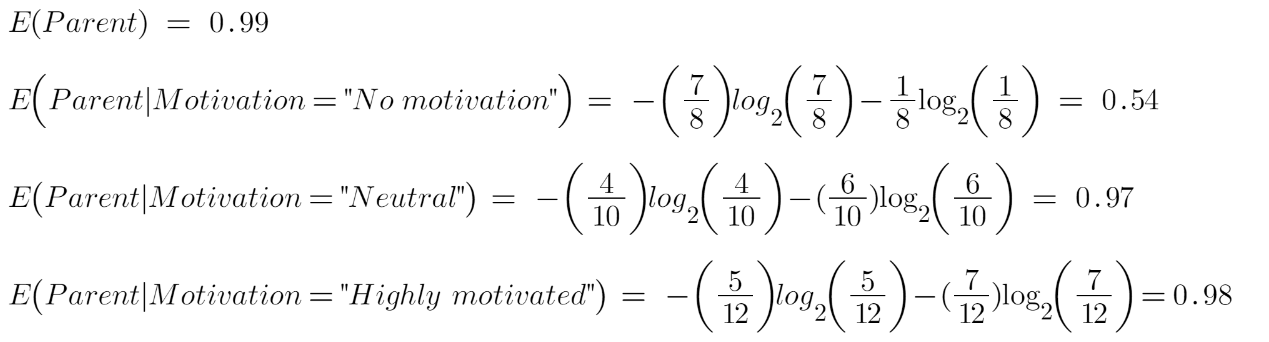
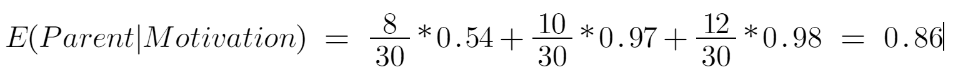


Image Source: Author

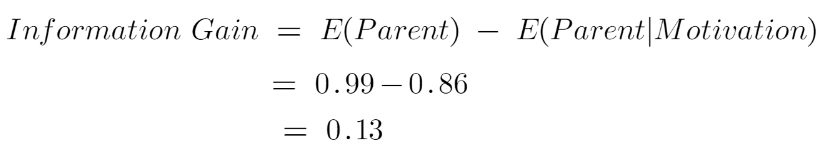
Let’s calculate the entropy here:



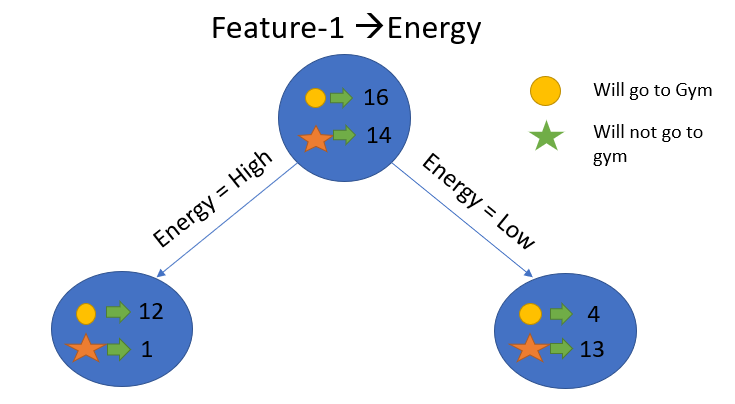
To see the weighted average of entropy of each node we will do as follows:



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



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.

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](https://www.youtube.com/watch?v=XABw4Y3GBR4).

**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.

There are mainly 2 ways for pruning:

(i) **Pre-pruning** – we can stop growing the tree earlier, which means we can prune/remove/cut a node if it has 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](https://github.com/AnshulSaini17/Decision-Trees) repository.

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

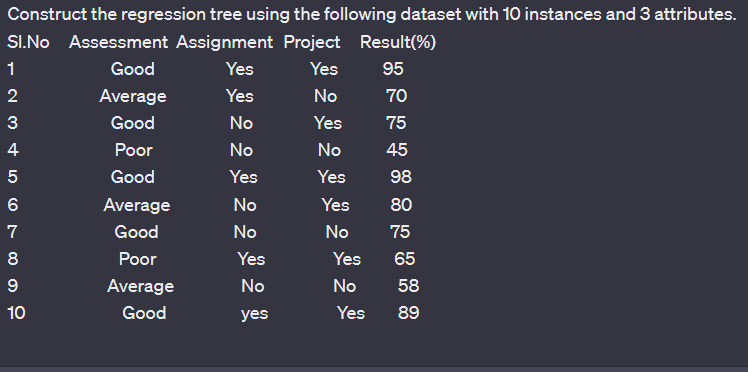
**Example: Decision Tree for Loan Approval**

Suppose a bank is making decisions about approving loans based on two factors: income level and credit history. Here's the dataset:



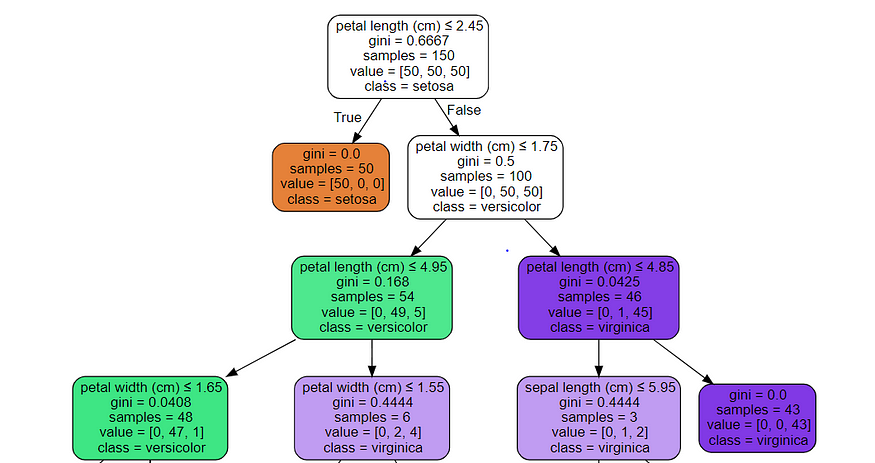
Let's construct a decision tree using Gini impurity and entropy to determine the best splits.

**Example:**

****

**NOTE-2**

Though Decision Trees look simple and intuitive, there is nothing very simple about how the algorithm goes about the process deciding on splits and how tree pruning occurs. In this post I take you through a simple example to understand the inner workings of Decision Trees.



Iris Decision Tree from Scikit Learn ( Image source: [sklearn](https://scikit-learn.org/stable/modules/tree.html))

Decision Trees are a popular and surprisingly effective technique, particularly for classification problems. But, the seemingly intuitive interface hides complexities. The criterion for selecting variables and hierarchy can be tricky to get, not to mention Gini index, Entropy ( wait, isn’t that physics?) and information gain (isn’t that information theory?). As you can see there are lots of tricky problems on which you can get stuck on. The best way to understand Decision Trees is to work through a small example which has sufficient complexity to be able to demonstrate some of the common points one suddenly goes, ‘ not sure what happens here…?’.

This post is therefore more like a tutorial or a demo where I will work through a toy dataset that I have created to understand the following:

1. What is a decision tree: root node, sub nodes, terminal/leaf nodes

2. Splitting criteria: Entropy, Information Gain vs Gini Index

3. How do sub nodes split

4. Why do trees overfit and how to stop this

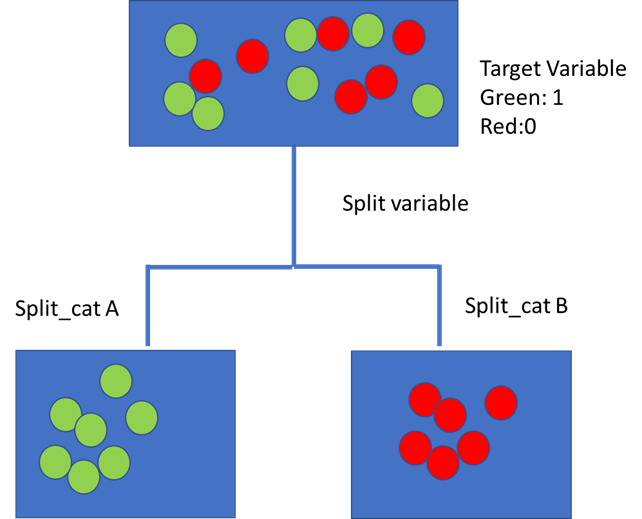
5. How to predict using a decision tree

So, let’s get demonstrating…

**1.** **What does a Decision Tree do?**

Let’s begin at the real beginning with core problem. For example, we are trying to classify whether a patient is diabetic or not based on various predictor variables such as fasting blood sugar, BMI, BP, etc. This is obviously a prediction problem for a new patient. We also have 1000 patient records to help us develop an understanding of which features are most useful in predicting. Unlike other classification algorithms such as Logistic Regression, Decision Trees have a somewhat different way of functioning and identifying which variables are important.

The first thing to understand in Decision Trees is that they split the predictor space, i.e., the target variable into different sub groups which are relatively more homogenous from the perspective of the target variable. For example, if the target variable is binary, with categories 1 and 0 ( shown by green and red dots in the image below, then the decision tree works to split the target variable space into sub groups that are more homogenous in terms of having either 1’s or 0’s.



Target Variable Splitting process (Image source: author)

That is the overall concept. Let us begin with understanding the various elements of a decision tree.

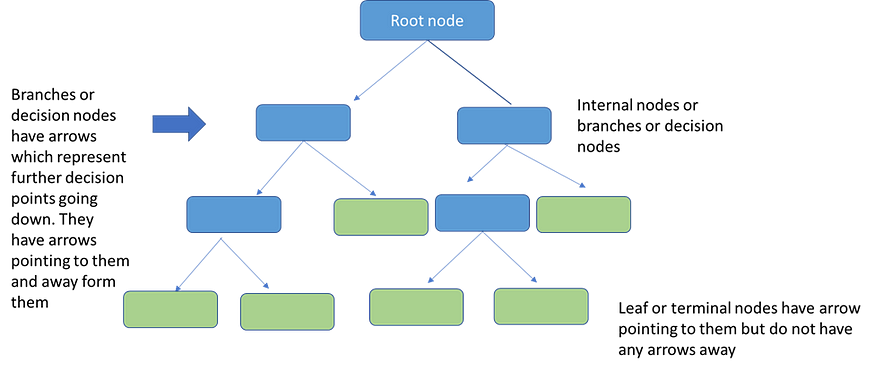
**Understanding components of a Decision Tree**

A decision tree is a branching flow diagram or tree chart. It comprises of the following components:

. A target variable such as diabetic or not and its initial distribution.

* A root node: this is the node that begins the splitting process by finding the variable that best splits the target variable
* Node purity: Decision nodes are typically impure, or a mixture of both classes of the target variable (0,1 or green and red dots in the image). Pure nodes are those that have one class — hence the term pure. They either have green or red dots only in the image.
* Decision nodes: these are subsequent or intermediate nodes, where the target variable is again split further by other variables
* Leaf nodes or terminal nodes are pure nodes, hence are used for making a prediction of a numerical or class is made.

Let’s see this visually..



Structure of a Decision Tree (image source: my collection)

In general a decision tree takes a statement or hypothesis or condition and then makes a decision on whether the condition holds or does not. The conditions are shown along the branches and the outcome of the condition, as applied to the target variable, is shown on the node.

Arrows leading away from a node indicate a condition which is being applied to the node. Arrows pointing to a node indicate a condition that is being satisfied.

This is the first level of the Decision Tree — understanding the flow of splitting the decision space into smaller spaces which ultimately become more and more homogenous in the target variable. This ultimately leads to a prediction.

Decision Trees offer tremendous flexibility in that we can use both numeric and categorical variables for splitting the target data. Categoric data is split along the different classes in the variable. Numeric is a little more tricky as we have to split into thresholds for the condition being tested such as <18 and ≥18, for example. A numeric variable can appear multiple times in the data with different cut offs or thresholds. Also final classifications can be repeated.

The important things from data science perspective are:

1. Flow of information through the Decision Tree

2. How does Decision Trees select which variable to split on at decision nodes?

3. How does it decide that the tree has enough branches and that it should stop splitting?

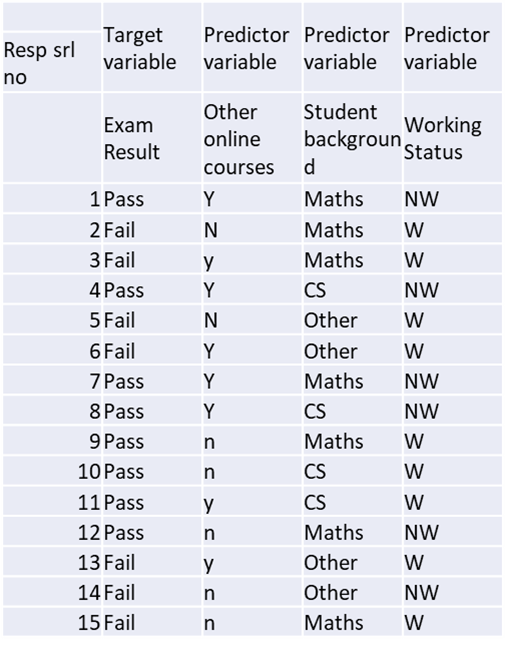
Now let us look at a simplified toy example to understand the above process more concretely.

**First the problem:**

We have data for 15 data points of student data on pass or fail an online ML exam. To understand the basic process we begin with a dataset which comprises a target variable that is binary ( Pass/Fail) and various binary or categorical predictor variables such as:

* whether enrolled in other online courses
* whether student is from a maths, computer science or other background
* Whether working or not working

The dataset is given below:



Toy Dataset for Online ML Exam( source: author)

Notice that only one variable, ‘Student Background’ has more than 2 levels or categories — Maths, CS, Others. It is one for the advantages of Decision Trees compared to other classification models such as Logistic Regression or SVM, that we do not need to carry out one hot encoding to make these into dummy variables.

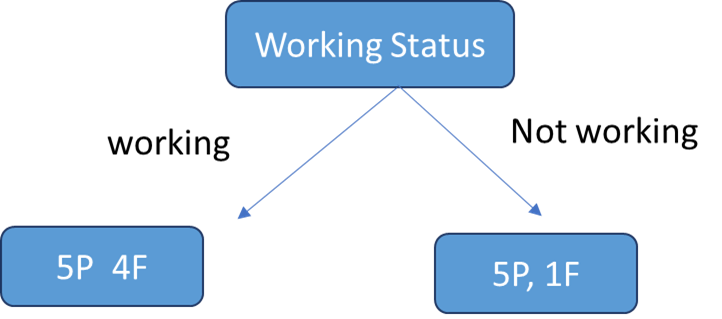
Let us first look at the flow of how a decision tree works and then we will dive into the complexities of how the decisions are actually made…

**Flow of a Decision Tree**

A decision tree begins with the target variable. This is usually called the parent node. The Decision Tree then makes a sequence of splits based in hierarchical order of impact on this target variable. From the analysis perspective the first node is the **root node**, which is the first variable that splits the target variable.

To identify the root node we would evaluate the impact of all the variables that we have currently on the target variable to identify the variable that splits the exam Pass/Fail classes into the most homogenous groups. Our candidates for splitting this are: Background, Working Status and Other Online Courses.

What do we hope to achieve with this split? Suppose we begin with Working Status as the root node. This splits into 2 sub nodes, one each for Working and Not working. Thus the Pass/Fail status is updated in each sub node respectively.



Sample Decision Tree Flow( Image source: author created

So, this is the basic flow of the Decision Tree. As long as there is a a mixture of Pass and Fail in a sub node, there is scope to split further to try and get it to be only one category. This is termed the purity of the node. For example, Not Working has 5 Pass and 1 Fail, hence it is purer than the Working node which has 5P and 4F. A leaf node would be one which contains either Pass or Fail class only.

A node which is impure can be branched further for improving purity. However, most of the time we do not necessarily go down to the point where each leaf is ‘pure’. It is also important to understand that each node is standalone and hence the attribute that best splits the ‘Working’ node may not be the one that best splits the ‘Not Working’ node.

Now, let us move on to learning the core part of Decision Trees - key questions:

**How does the Tree decide which variable to branch out at each level?**

**Greedy top down approach**

Decision trees follow a a top-down, greedy approach that is known as recursive binary splitting. The recursive binary splitting approach is top-down because it begins at the top of the tree and then it successively splits the predictor space. At each split the predictor space gets divided into 2 and is shown via two new branches pointing downwards. The algorithm is termed is greedy because at each step of the process, the best split is made for that step. It does not project forwards and try and pick a split that might be more optimal for the overall tree.

The algorithm therefore evaluates all variables on some statistical criteria and then chooses the variable that performs best on the criteria.

**Variable selection criterion**

Here is where the true complexity and sophistication of decision lies. Variables are selected on a complex statistical criterion which is applied at each decision node. Now, variable selection criterion in Decision Trees can be done via two approaches:

1. Entropy and Information Gain

2. Gini Index

Both criteria are broadly similar and seek to determine which variable would split the data to lead to the underlying child nodes being most homogenous or pure. Both are used in different Decision Tree algorithms. To add to the confusion, it is not clear which one is the preferred approach. So, one has to have an understanding of both.

Let us begin with Entropy and Information Gain criterion

**What is Entropy?**

Entropy is a term that comes from physics and means a measure of disorder. More specifically, we can define it as:

**Entropy** is a scientific concept as well as a measurable physical property that is most commonly associated with a state of disorder, randomness, or uncertainty. The term and the concept are used in diverse fields, from [classical thermodynamics](https://en.wikipedia.org/wiki/Classical_thermodynamics), where it was first recognized, to the microscopic description of nature in [statistical physics](https://en.wikipedia.org/wiki/Statistical_physics), and to the principles of [information theory](https://en.wikipedia.org/wiki/Information_theory).

<https://en.wikipedia.org/wiki/Entropy>

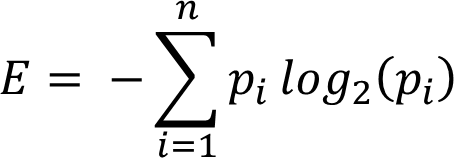
In [information theory](https://en.wikipedia.org/wiki/Information_theory), the **entropy** of a [random variable](https://en.wikipedia.org/wiki/Random_variable) is the average level of “information”, “surprise”, or “uncertainty” inherent to the variable’s possible outcomes.

<https://en.wikipedia.org/wiki/Entropy_(information_theory)>

In the context of Decision Trees, entropy is a measure of disorder or impurity in a node. Thus, a node with more variable composition, such as 2Pass and 2 Fail would be considered to have higher Entropy than a node which has only pass or only fail. The maximum level of entropy or disorder is given by 1 and minimum entropy is given by a value 0.

Leaf nodes which have all instances belonging to 1 class would have an entropy of 0. Whereas, the entropy for a node where the classes are divided equally would be 1.

Entropy is measured by the formula:



Where the pi is the probability of randomly selecting an example in class i. Let us understand this a bit better in the context of our example. So, the initial entropy of at the parent node is given by the probability of getting a pass vs fail. In our dataset, the target variable has 9 passes and 6 fails. Hence the probabilities for the entropy formula are:





Now essentially what a Decision Tree does to determine the root node is to calculate the entropy for each variable and its potential splits. For this we have to calculate a potential split from each variable, calculate the average entropy across both or all the nodes and then the change in entropy vis a vis the parent node. This change in entropy is termed Information Gain and represents how much information a feature provides for the target variable.



Entropy\_parent is the entropy of the parent node and Entropy\_children represents the average entropy of the child nodes that follow this variable. In the current case since we have 3 variables for which this calculation must be done from the perspective of the split.

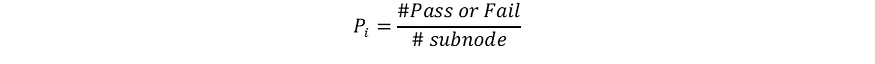
1. Work Status

2. Online Course Status

3. Student Background

To calculate entropy, first let us put our formulas for Entropy and Information Gain in terms of variables in our dataset:

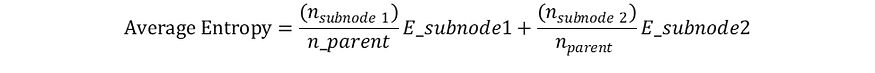
1. Probability of pass and fail at each node, i.e, the Pi:



2. Entropy:



3. Average Entropy at child nodes:



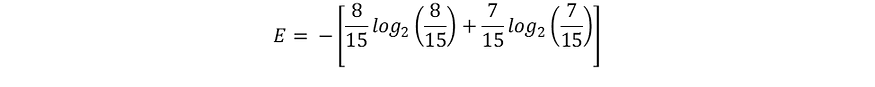
Note, average Entropy is the weighted average of all the sub nodes that a parent node splits into. Thus, in our example this would be 2 sub nodes for Working Status and 3 sub nodes for Student Background.

4. Information Gain:



**Parent Node Calculations**

First we will calculate parent node entropy using the formula above. Use any log2 calculator online to calculate the log values. In our case they work out to:





(Mathematical note: log to base 2 of anything less than 1 is a negative number, hence we multiply by a minus sign to get a positive number)

So far, this is just the entropy of the parent node. Now we have to decide which attribute or variable to use to split this to get the root node.

**Calculating the Root Node**

For this we have to calculate a potential split from each variable, calculate the average entropy across both the nodes and then the change in entropy via a vis the parent node.

Let us begin with Work Status variable and calculate the entropy of the split.





We then calculate the average entropy for the Working status split as a weighted average with weights of the share of observations from the total number of observations that fall in each sub node.



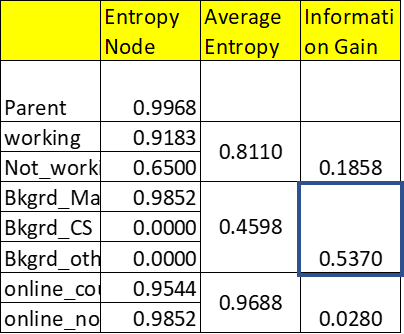
Information Gain = Entropy\_Parent — Entropy\_child =

0.9183–0.8119 = .1858

(Calculations are shown in the spreadsheet below)

In a similar fashion we can evaluate the entropy and information gain for Student Background and Online Courses variables. The results are provided in the table below:

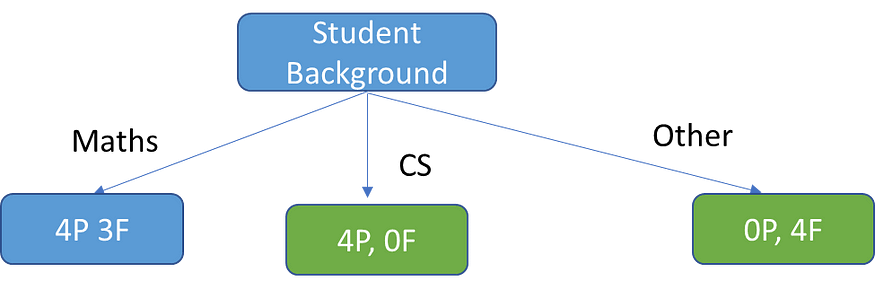
We will drop this variable for the time being and move on to evaluate the other variables. The spreadsheet below shows the Entropy calculations for all the variables:



Root Node Entropy Calculations (source: author)

To find the root node attribute we look at the Information gain from Student Background vis a vis initial parent entropy. This shows the maximum reduction of .5370. Hence, this is the attribute that would be selected as the root node. The other’s variables — ‘Working Status’ and ‘Online Courses’ show a much lower decrease in entropy vis a vis the parental node.

So, on the basis of the above calculations, we have determined what the root node would be. The tree would now look as follows:



Root Node of Decision Tree

Student Background splits the target variable into 3 groups. Everyone from CS background clearly passes and hence this is a terminal or leaf node. Everyone from Other backgrounds fails and this is also a terminal node. Maths background is split into 3 Pass and 4 Fail and hence it is impure and there is some scope for further splitting to attain greater purity.

Now to split the Maths background sub node, we need to calculate Entropy and Information Gain for the remaining variables, i.e., Working Status and Online Courses. We would then select the variable that shows the highest Information Gain.

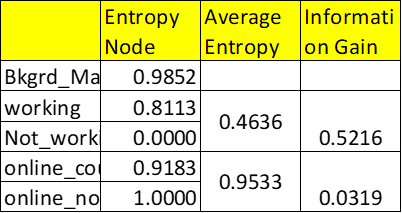
The Entropy and Information Gain Calculations for the Maths Background node can be seen in the table below. Notice, we now have the Maths Background as the node that is being split, hence average Entropy for the splits is calculated using it as a base.

Note: putting in log(0) throws an error. However mathematically we can use the limit. Normally we just don’t include the pj=0 case in the calculation. However, I have included just to show the complete calculation.

By convention **0 log 0 = 0**, since y log y → 0 as y → 0.

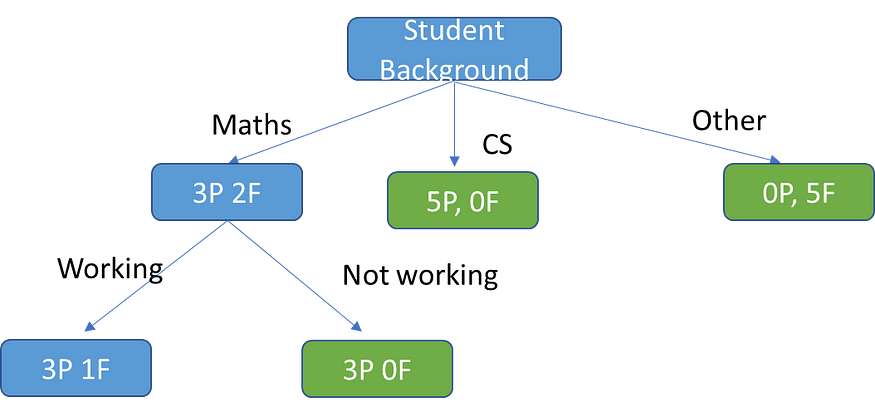
<https://www.icg.isy.liu.se/courses/infotheory/lect1.pdf>

The Entropy for each potential split is:



Splitting the Maths Subnode (Image source: author)

As we can see Information Gain is higher for the Working Status variable. Hence this is the variable used to continue branching.



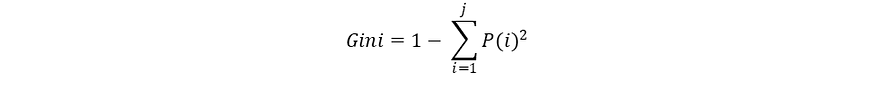
Maths Node Branching

We now see that the Maths node has split into 1 terminal node on the right and one node which is still impure. Notice, now almost all our nodes are terminal nodes. There is only one node which is not terminal. We can try splitting it further using Other Online Courses. Anyway, you get the picture. In any case most Decision Trees do not necessarily split to the point where every node is a terminal node. Most algorithms have built in stops which we will discuss a little further down. Further, if the Decision Tree continues to split we have another problem which is that of overfitting. Again we shall discuss that below after we have briefly reviewed an alternative approach to developing a Decision Tree using the Gini Index.

**Gini Index**

The other way of splitting a decision tree is via the Gini Index. The Entropy and Information Gain method focuses on purity and impurity in a node. The Gini Index or Impurity measures the probability for a random instance being misclassified when chosen randomly. The lower the Gini Index, the better the lower the likelihood of misclassification.

The formula for Gini Index



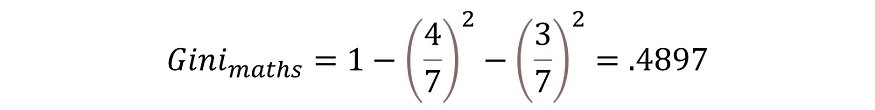
Where j represents the no. of classes in the target variable — Pass and Fail in our example

P(i) represents the ratio of Pass/Total no. of observations in node.

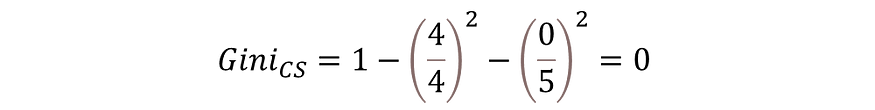
So, Let’s take an example from the decision tree above. Let’s begin with the root node and calculate the Gini Index for each of the splits. The Gini Index has a minimum (highest level of purity) of 0. It has a maximum value of .5. If Gini Index is .5, it indicates a random assignment of classes.

Now let us calculate the Gini index for the root node for Student Background attribute. In this case we have 3 nodes. Gini formula requires us to calculate the Gini Index for each sub node. Then do a weighted average to calculate the overall Gini Index for the node.

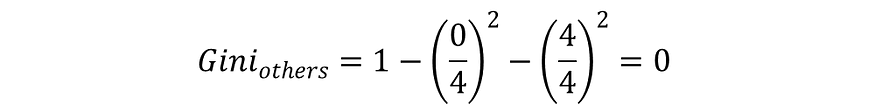
Maths sub node: 4Pass, 3Fail



CS sub node: 4Pass, 0 Fail

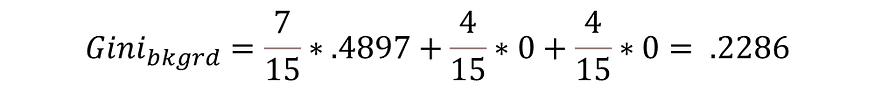


Others sub node: 0Pass, 4 Fail



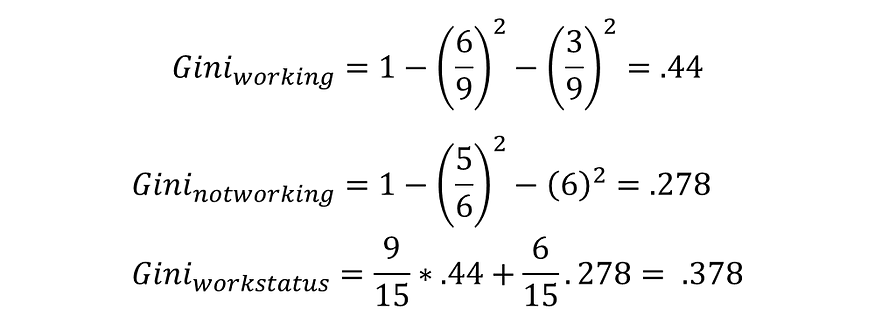
As we can see the probability for misclassification in CS node is zero, since everyone passes. Similarly no scope for misclassification on Others node as everyone fails. Only the maths node has possibility of misclassification, and this is quite high, given that the maximum Gini Index is .5.

The overall Gini Index for this split is calculated similarly to the entropy as weighted average of the distribution across the 3 nodes.

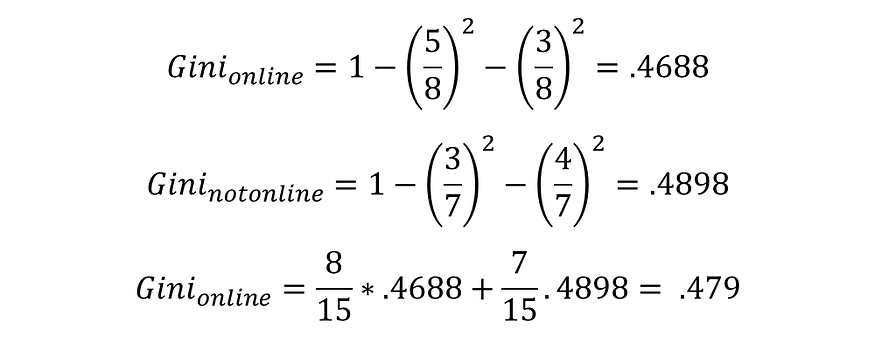


Similarly, we can also compute the Gini Index for Working Status and Online Courses. These are given below:

Working/Not working



Online Courses



The Gini Index is lowest for the Student Background variable. Hence, similar to the Entropy and Information Gain criteria, we pick this variable for the root node. In a similar fashion we would again proceed to move down the tree, carrying out splits where node purity is less

**Gini Index vs Information Gain**

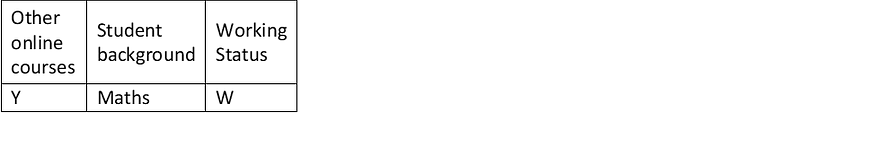
Depending on which impurity measurement is used, tree classification results can vary. This can make small (or sometimes large) impact on your model. There seems to be no one preferred approach by different Decision Tree algorithms. For example, CART uses Gini; ID3 and C4.5 use Entropy.

The Gini index has a maximum impurity is 0.5 and maximum purity is 0, whereas Entropy has a maximum impurity of 1 and maximum purity is 0.

**How does a prediction get made in Decision Trees**

Now that we have understood, hopefully in detail, how Decision Trees carry out splitting and variable selection, we can move on to how they do prediction. Actually, once a tree is trained and tested, prediction is easy. The tree basically provides a flow chart based on various predictor variables. Suppose we have a new instance entering the flow along with its values of different predictor variables. Unlike training and test data , it will not have the class for the target attribute. We are trying to predict this class by moving down the tree, testing its values of different predictor variables at different branches. Ultimately, the new instance will move into a leaf node and will be classified according to the class prevailing in the leaf node.

Suppose it looks like the below configuration:



Based on our tree, we would first check the Math branch, then Working Yes branch. That as we have seen is a leaf node and the new observation would be classified on the basis of the majority vote in this node, i.e., since it is Pass, this new observation would also be predicted to be Pass.

In practice, when the algorithm is evaluating a new example and reaches a leaf node, the prediction is based on the modal value of categories in the leaf node. As seen in the above case, the Working node is not fully pure. However, we go with the prediction of the modal value, which is Pass. In general, most leaf nodes are not pure and, hence for categorical prediction, we use the modal value for prediction. If it is a numerical prediction (regression tree), we predict the mean value of the target values at each leaf node.