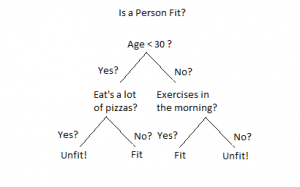
**Decision Trees**



Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision **nodes** and **leaves**. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-1.png)

An example of a decision tree can be explained using above binary tree. Let’s say you want to predict whether a person is fit given their information like age, eating habit, and physical activity, etc. The decision nodes here are questions like ‘What’s the age?’, ‘Does he exercise?’, ‘Does he eat a lot of pizzas’? And the leaves, which are outcomes like either ‘fit’, or ‘unfit’. In this case this was a binary classification problem (a yes no type problem).

There are two main types of Decision Trees:

1. **Classification trees** (Yes/No types)

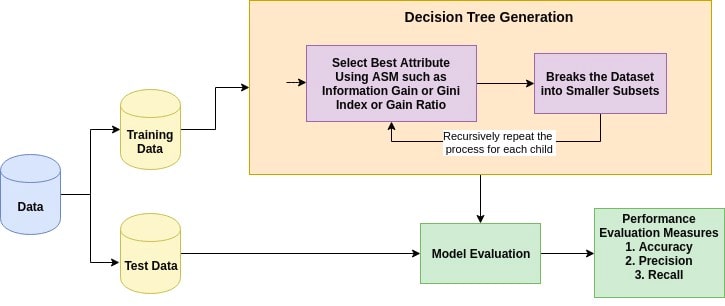
What we’ve seen above is an example of classification tree, where the outcome was a variable like ‘fit’ or ‘unfit’. Here the decision variable is **Categorical**.

1. **Regression trees** (Continuous data types)

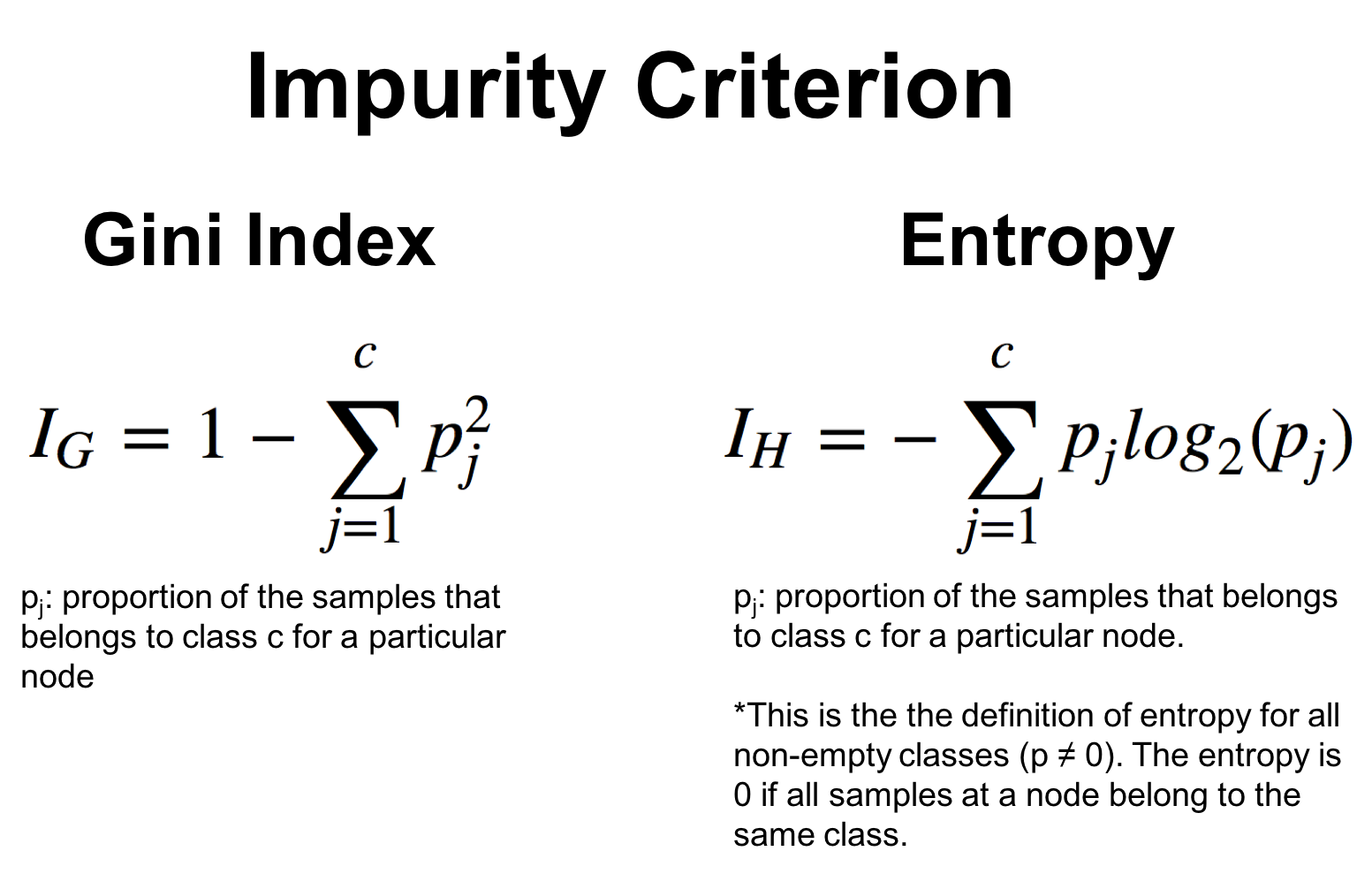
Here the decision or the outcome variable is **Continuous**, e.g. a number like 123.

**Working**

Now that we know what a Decision Tree is, we’ll see how it works internally. There are many algorithms out there which construct Decision Trees, but one of the best is called as **ID3 Algorithm**. ID3 Stands for **Iterative Dichotomiser 3**.

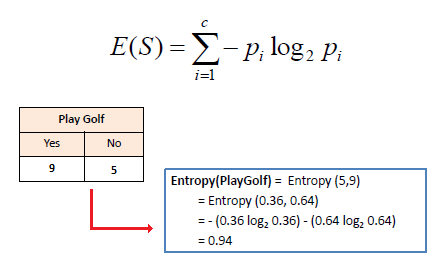


Before discussing the ID3 algorithm, we’ll go through few definitions.



**Entropy**

Entropy, also called as Shannon Entropy is denoted by H(S) for a finite set S, is the measure of the amount of **uncertainty or randomness** in data.



Intuitively, it tells us about the predictability of a certain event. Example, consider a coin toss whose probability of heads is 0.5 and probability of tails is 0.5. Here the entropy is the highest possible, since there’s no way of determining what the outcome might be. Alternatively, consider a coin which has heads on both the sides, the entropy of such an event can be predicted perfectly since we know beforehand that it’ll always be heads. In other words, this event has **no randomness** hence it’s entropy is zero.

In particular, **lower values** imply **less uncertainty** while higher values imply high uncertainty.

**Information Gain**

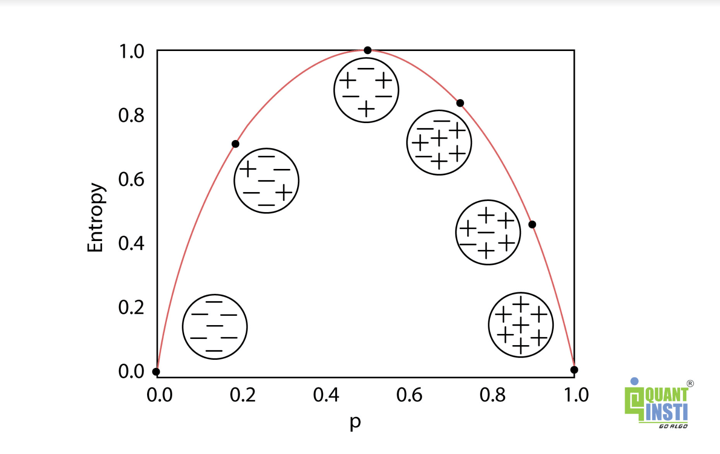
Information gain is also called as Kullback-Leibler divergence denoted by IG(S,A) for a set S is the effective change in entropy after deciding on a particular attribute A. It measures the relative change in entropy with respect to the independent variables.

[Decision Trees modified](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-3.jpg)

Alternatively,

[Decision Trees modified](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-4.jpg)

where IG(S, A) is the information gain by applying feature A. H(S) is the Entropy of the entire set, while the second term calculates the Entropy after applying the feature A, where P(x) is the probability of event x.



**ID3 Algorithm will perform following tasks recursively**

1. ***Create root node for the tree***
2. ***If all examples are positive, return leaf node ‘positive’***
3. ***Else if all examples are negative, return leaf node ‘negative’***
4. ***Calculate the entropy of current state H(S)***
5. ***For each attribute, calculate the entropy with respect to the attribute ‘x’ denoted by H(S, x)***
6. ***Select the attribute which has maximum value of IG(S, x)***
7. ***Remove the attribute that offers highest IG from the set of attributes***
8. ***Repeat until we run out of all attributes, or the decision tree has all leaf nodes.***

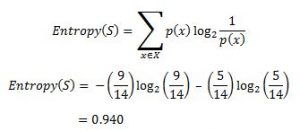
Let’s understand this with the help of an example

Consider a piece of data collected over the course of 14 days where the features are Outlook, Temperature, Humidity, Wind and the outcome variable is whether Golf was played on the day. Now, our job is to build a predictive model which takes in above 4 parameters and predicts whether Golf will be played on the day. We’ll build a decision tree to do that using **ID3 algorithm.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Day | Outlook | Temperature | Humidity | Wind | Play Golf |
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

Now we’ll go ahead and grow the decision tree. The initial step is to calculate H(S), the Entropy of the current state. In the above example, we can see in total there are 5 No’s and 9 Yes’s.

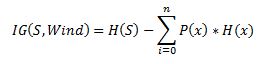
|  |  |  |
| --- | --- | --- |
| Yes | No | Total |
| 9 | 5 | 14 |

[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-5.jpg)

Remember that the Entropy is 0 if all members belong to the same class, and 1 when half of them belong to one class and other half belong to other class that is perfect randomness. Here it’s 0.94 which means the distribution is fairly random.

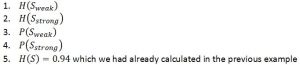
**Now the next step is to choose the attribute that gives us highest possible Information Gain** which we’ll choose as the root node.

Let’s start with ‘Wind’

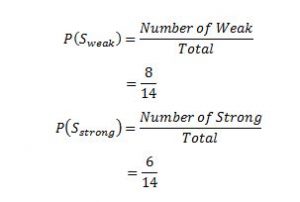
[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-6.jpg)

where ‘x’ are the possible values for an attribute. Here, attribute ‘Wind’ takes two possible values in the sample data, hence x = {Weak, Strong}

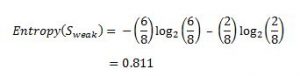
We’ll have to calculate:

[[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-77.jpg)](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-77.jpg)Amongst all the 14 examples we have **8 places where the wind is weak and 6 where the wind is Strong**.

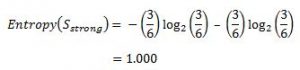
|  |  |  |
| --- | --- | --- |
| Wind = Weak | Wind = Strong | Total |
| 8 | 6 | 14 |

[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-8.jpg)

Now out of the 8 Weak examples, 6 of them were ‘Yes’ for Play Golf and 2 of them were ‘No’ for ‘Play Golf’. So, we have,

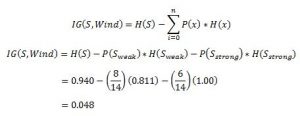
[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-9.jpg)

Similarly, out of 6 Strong examples, we have **3 examples where the outcome was ‘Yes’ for Play Golf and 3 where we had ‘No’ for Play Golf**.

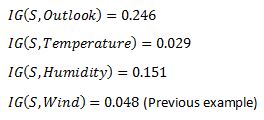
[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-10.jpg)

Remember, here half items belong to one class while other half belong to other. Hence we have perfect randomness.

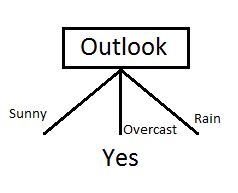
Now we have all the pieces required to calculate the Information Gain,

[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-11.jpg)

Which tells us the Information Gain by considering ‘Wind’ as the feature and give us information gain of **0.048**. Now we must similarly calculate the Information Gain for all the features.

[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-12.jpg)

We can clearly see that IG(S, Outlook) has the highest information gain of 0.246,**hence we chose Outlook attribute as the root node**. At this point, the decision tree looks like.

[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-13.jpg)

Here we observe that whenever the outlook is Overcast, Play Golf is always ‘Yes’, it’s no coincidence by any chance, the simple tree resulted because of **the highest information gain is given by the attribute Outlook**.

Now how do we proceed from this point? We can simply apply **recursion**, you might want to look at the algorithm steps described earlier.

Now that we’ve used Outlook, we’ve got three of them remaining Humidity, Temperature, and Wind. And, we had three possible values of Outlook: Sunny, Overcast, Rain. Where the Overcast node already ended up having leaf node ‘Yes’, so we’re left with two subtrees to compute: Sunny and Rain.

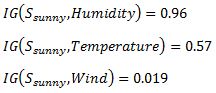
[Decision Trees modified](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-33.jpg)

Table where the value of Outlook is Sunny looks like:

|  |  |  |  |
| --- | --- | --- | --- |
| Temperature | Humidity | Wind | Play Golf |
| Hot | High | Weak | No |
| Hot | High | Strong | No |
| Mild | High | Weak | No |
| Cool | Normal | Weak | Yes |
| Mild | Normal | Strong | Yes |

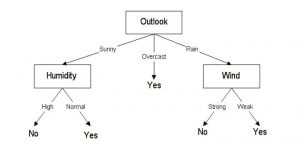
[Decision Trees modified](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-14.jpg)

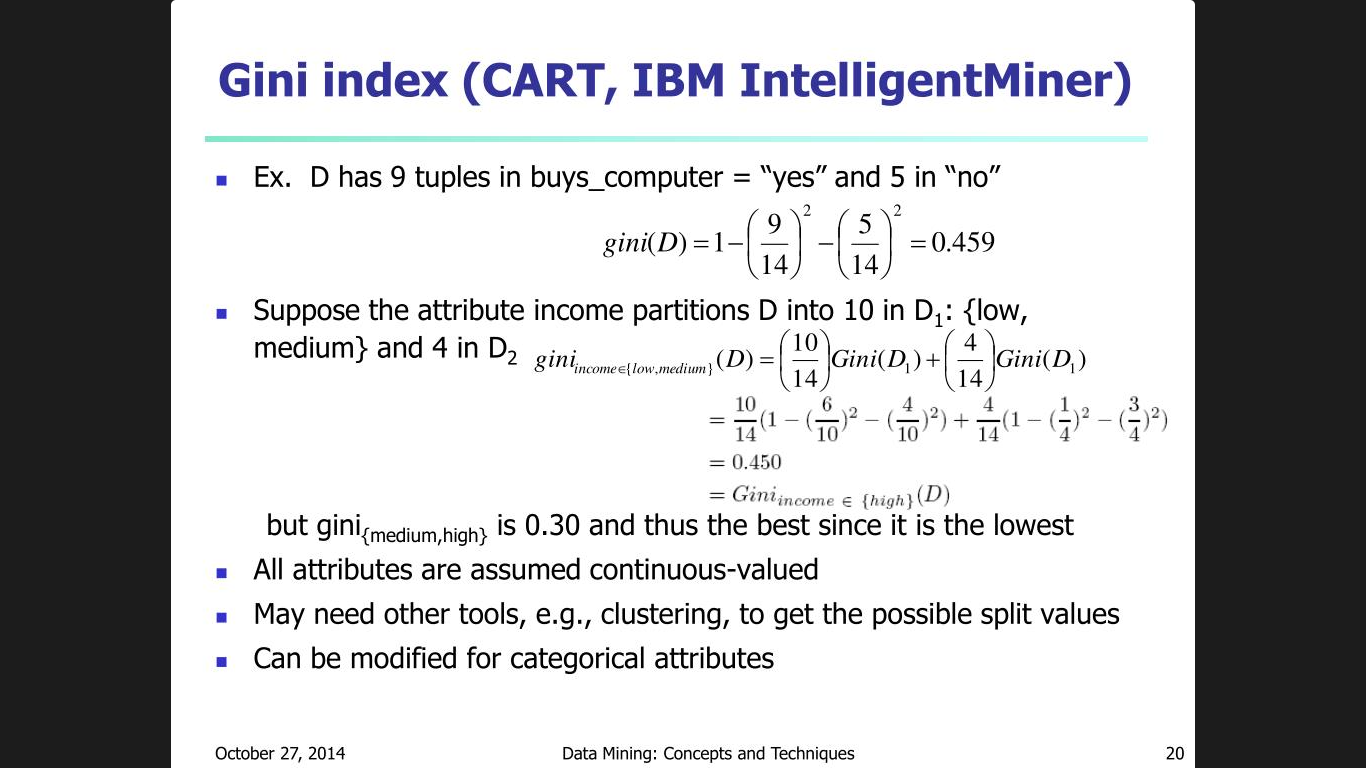
In the similar fashion, we compute the following values

[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-15.jpg)

As we can see the **highest Information Gain is given by Humidity**. Proceeding in the same way with [Decision Trees modified](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-55.jpg) will give us Wind as the one with highest information gain. The final Decision Tree looks something like this.

The final Decision Tree looks something like this.

[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-16.jpg)



***#Importing required libraries***

from sklearn.datasets import load\_iris

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

***#Loading the iris data***

data = load\_iris()

print('Classes to predict: ', data.target\_names)

print('Classes to predict: ', data.feature\_names)

***#Extracting data attributes***

X = data.data

***#Extracting target/ class labels***

y = data.target

print('Number of examples in the data:', X.shape)

***#First four rows in the variable 'X'***

print(X[:4])

***#Using the train\_test\_split to create train and test sets.***

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state = 42, train\_size = 0.7)

***#Importing the Decision tree classifier from the sklearn library.***

***#from sklearn.tree import DecisionTreeClassifier***

clf = DecisionTreeClassifier()

***#Training the decision tree classifier.***

clf.fit(X\_train, y\_train)

***#Predicting labels on the test set.***

y\_pred = clf.predict(X\_test)

***#Importing the accuracy metric from sklearn.metrics library***

from sklearn.metrics import accuracy\_score

print('Accuracy Score on train data: ', accuracy\_score(y\_true=y\_train, y\_pred=clf.predict(X\_train))\*100)

print('Accuracy Score on test data: ', accuracy\_score(y\_true=y\_test, y\_pred=y\_pred)\*100)

from sklearn.metrics import classification\_report, confusion\_matrix

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

***#Next, we will tune the parameters of the decision tree to increase its accuracy.***

***#One of those parameters is 'min\_samples\_split', which is the minimum number of samples required to split an internal node.***

***#Its default value is equal to 2 because we cannot split on a node containing only one example/ sample.***

clf = DecisionTreeClassifier(criterion='entropy', min\_samples\_split=50)

clf.fit(X\_train, y\_train)

print('Accuracy Score on train data: ', accuracy\_score(y\_true=y\_train, y\_pred=clf.predict(X\_train))\*100)

print('Accuracy Score on the test data: ', accuracy\_score(y\_true=y\_test, y\_pred=clf.predict(X\_test))\*100)