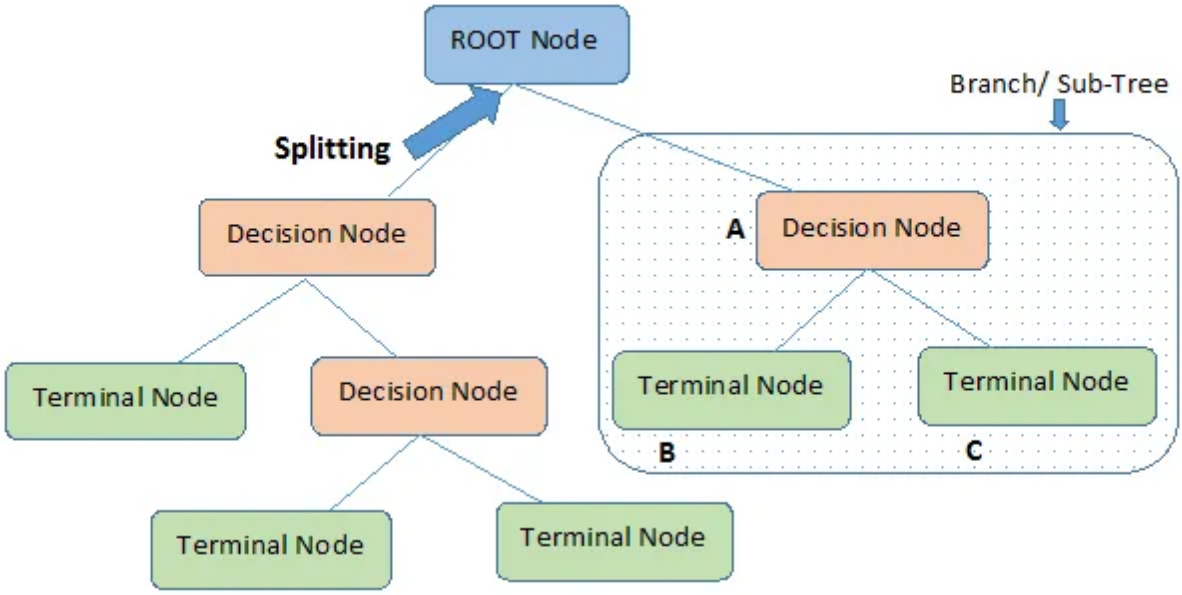
Decision Tree Algorithm With Hands-On Example

Arun Mohan · Follow

Published in DataDrivenInvestor

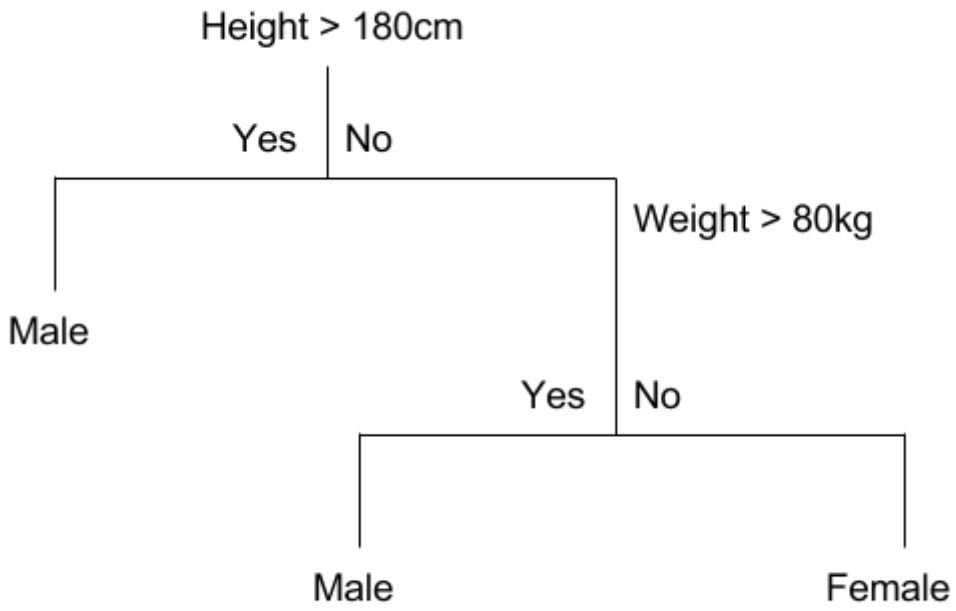
6 min read · Jan 23, 2019



The decision tree is one of the most important machine learning algorithms. It is used for both classification and regression problems. In this article, we will go through the classification part.

What is a decision tree?

A decision tree is a classification and prediction tool having a tree-like structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



Above we have a small decision tree. An important advantage of the decision tree is that it is highly interpretable. Here If Height > 180cm or if height < 180cm and weight > 80kg person is male.Otherwise female. Did you ever think about how we came up with this decision tree? I will try to explain it using the weather dataset.

Before going to it further I will explain some important terms related to decision trees.

Entropy

In machine learning, entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information.

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$

Information Gain

Information gain can be defined as the amount of information gained about a random variable or signal from observing another random variable.It can be considered as the difference between the entropy of parent node and weighted average entropy of child nodes.

$$IG(S,A) = H(S) - H(S,A)$$

Alternatively,

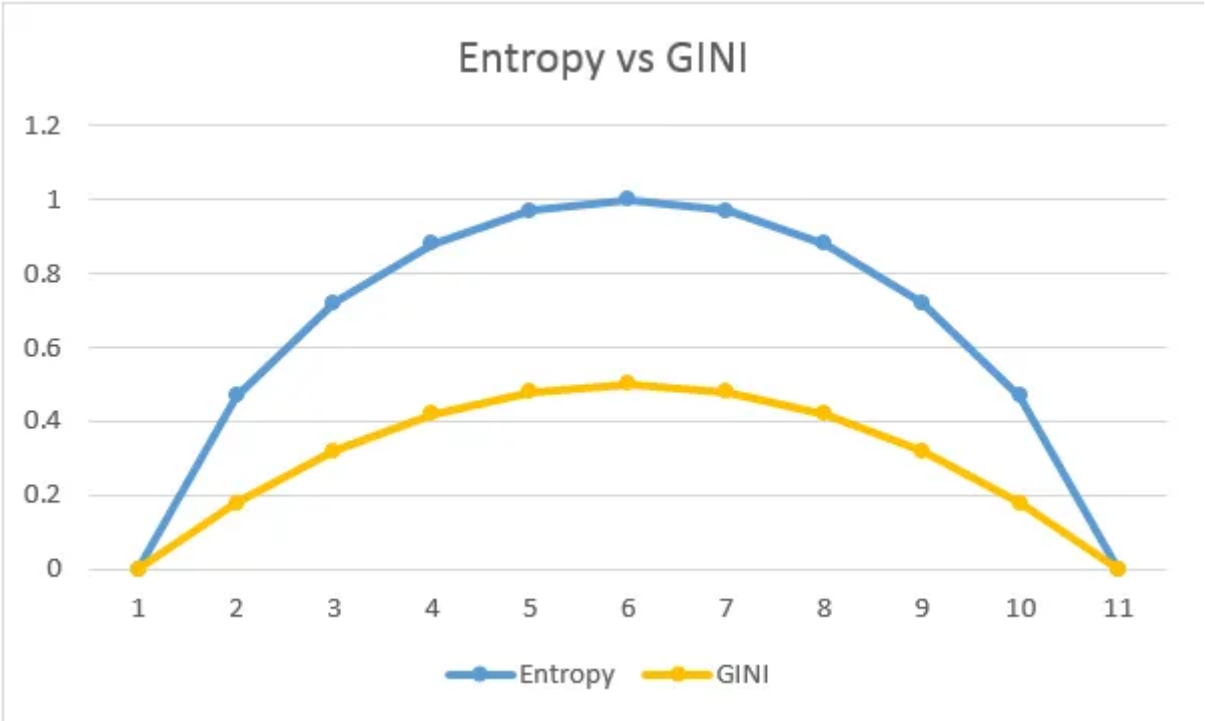
$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Gini Impurity

Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

Gini impurity is **lower bounded by 0**, with 0 occurring if the data set contains only one class.



There are many algorithms there to build a decision tree. They are

1. **CART** (Classification and Regression Trees) — This makes use of Gini impurity as the metric.
2. **ID3** (Iterative Dichotomiser 3) — This uses entropy and information gain as metric.

In this article, I will go through ID3. Once you got it it is easy to implement the same using CART.

Classification using the ID3 algorithm

Consider whether a dataset based on which we will determine whether to play football or not.

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Here There are for independent variables to determine the dependent variable. The independent variables are Outlook, Temperature, Humidity, and Wind. The dependent variable is whether to play football or not.

As the first step, we have to find the parent node for our decision tree. For that follow the steps:

Find the entropy of the class variable.

$E(S) = -[(9/14)\log(9/14) + (5/14)\log(5/14)] = 0.94$

note: Here typically we will take log to base 2.Here total there are 14 yes/no. Out of which 9 yes and 5 no.Based on it we calculated probability above.

From the above data for outlook we can arrive at the following table easily

		play		
		yes	no	total
	sunny	3	2	5
Outlook	overcast	4	0	4
	rainy	2	3	5
				14

Now we have to calculate average weighted entropy. ie, we have found the total of weights of each feature multiplied by probabilities.

E(S, outlook) = (5/14)*E(3,2) + (4/14)*E(4,0) + (5/14)*E(2,3) = (5/14)(-(3/5)log(3/5)-(2/5)log(2/5))+ (4/14)(0) + (5/14)((2/5)log(2/5)-(3/5)log(3/5)) = 0.693

The next step is to find the information gain. It is the difference between parent entropy and average weighted entropy we found above.

IG(S, outlook) = 0.94 - 0.693 = 0.247

Similarly find Information gain for Temperature, Humidity, and Windy.

IG(S, Temperature) = 0.940 - 0.911 = 0.029

IG(S, Humidity) = 0.940 - 0.788 = 0.152

IG(S, Windy) = 0.940 - 0.8932 = 0.048

Now select the feature having the largest entropy gain. Here it is Outlook. So it forms the first node(root node) of our decision tree.

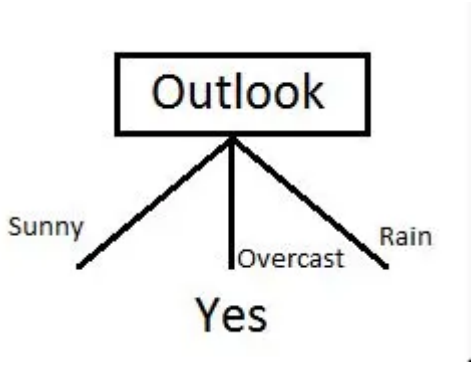
Now our data look as follows

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Overcast	Hot	High	Weak	Yes
Overcast	Cool	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Rain	Mild	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Since overcast contains only examples of class 'Yes' we can set it as yes. That means If outlook is overcast football will be played. Now our decision tree looks as follows.



The next step is to find the next node in our decision tree. Now we will find one under sunny. We have to determine which of the following Temperature, Humidity or Wind has higher information gain.

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

Calculate parent entropy E(sunny)

E(sunny) = (-(3/5)log(3/5)-(2/5)log(2/5)) = 0.971.

Now Calculate the information gain of Temperature. IG(sunny, Temperature)

		play		
		yes	no	total
	hot	0	2	2
Temperature	cool	1	1	2
	mild	1	0	1
				5

E(sunny, Temperature) = (2/5)*E(0,2) + (2/5)*E(1,1) + (1/5)*E(1,0)=2/5=0.4

Now calculate information gain.

IG(sunny, Temperature) = 0.971–0.4 =0.571

Similarly we get

IG(sunny, Humidity) = 0.971

IG(sunny, Windy) = 0.020

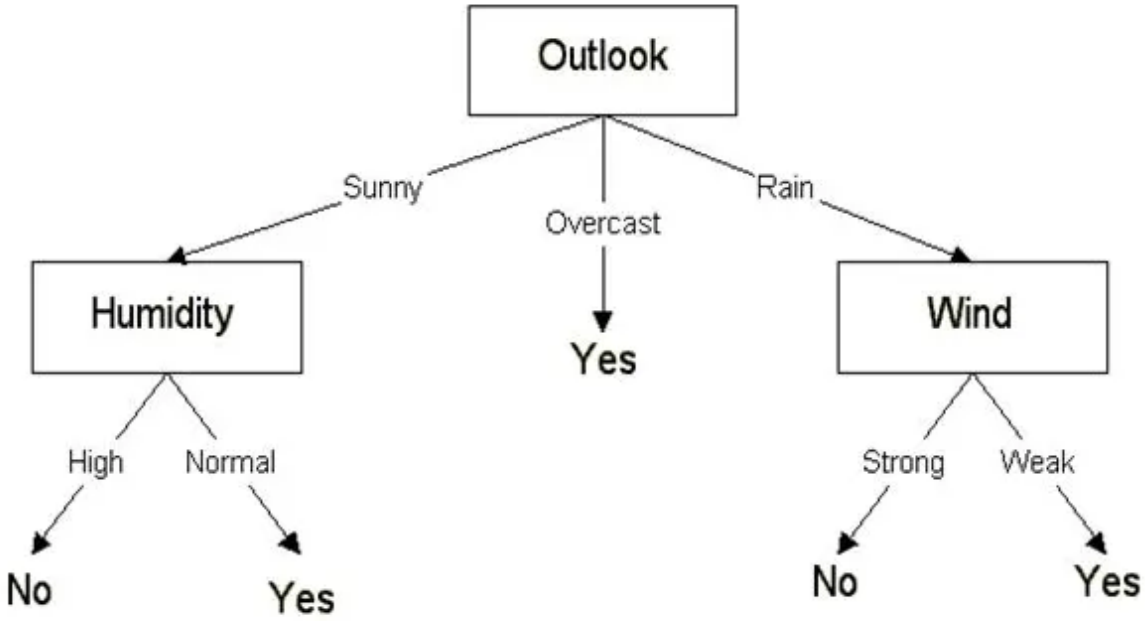
Here IG(sunny, Humidity) is the largest value. So Humidity is the node that comes under sunny.

	play	
Humidity	yes	no
high	0	3
normal	2	0

For humidity from the above table, we can say that play will occur if humidity is normal and will not occur if it is high. Similarly, find the nodes under rainy.

Note: A branch with entropy more than 0 needs further splitting.

Finally, our decision tree will look as below:



Classification using CART algorithm

Classification using CART is similar to it. But instead of entropy, we use Gini impurity.

So as the first step we will find the root node of our decision tree. For that Calculate the Gini index of the class variable

$Gini(S) = 1 - [(9/14)^2 + (5/14)^2] = 0.4591$

As the next step, we will calculate the Gini gain. For that first, we will find the average weighted Gini impurity of Outlook, Temperature, Humidity, and Windy.

First, consider case of Outlook

		play		
		yes	no	total
	sunny	3	2	5
Outlook	overcast	4	0	4
	rainy	2	3	5
				14

$Gini(S, outlook) = (5/14)gini(3,2) + (4/14)*gini(4,0)+ (5/14)*gini(2,3) = (5/14)(1 - (3/5)^2 - (2/5)^2) + (4/14)*0 + (5/14)(1 - (2/5)^2 - (3/5)^2)= 0.171+0+0.171 = 0.342$

$Gini\ gain\ (S, outlook) = 0.459 - 0.342 = 0.117$

Gini gain(S, Temperature) = 0.459 - 0.4405 = 0.0185

Gini gain(S, Humidity) = 0.459 - 0.3674 = 0.0916

Gini gain(S, windy) = 0.459 - 0.4286 = 0.0304

Choose one that has a higher Gini gain. Gini gain is higher for outlook. So we can choose it as our root node.

Now you have got an idea of how to proceed further. Repeat the same steps we used in the ID3 algorithm.

Advantages and disadvantages of decision trees

Advantages:

1. Decision trees are super interpretable
2. Require little data preprocessing
3. Suitable for low latency applications

Disadvantages:

1. More likely to overfit noisy data. The probability of overfitting on noise increases as a tree gets deeper. A solution for it is **pruning**. You can read more about pruning from my ***Kaggle notebook***. Another way to avoid overfitting is to use bagging techniques like Random Forest. You can read more about Random Forest from an article from ***neptune.ai***.

References:

- https://www.saedsayad.com/decision_tree.htm
- Applied-ai course