Entropy: How Decision Trees Make Decisions

The simple logic and math behind a very effective machine learning algorithm

Sam T Jan 11, 2019 · 9 min read

You're a Data Scientist in training. You've come a long way from writing your first line of Python or R code. You know your way around Scikit-Learn like the back of your hand. You spend more time on Kaggle than Facebook now. You're no stranger to building awesome random forests and other tree based ensemble models that get the job done. However , you're nothing if not thorough. You want to dig deeper and understand some of the intricacies and concepts behind popular machine learning models. Well , so do I.

In this blog post, I will introduce the concept of Entropy as a general topic in statistics which will allow me to further introduce the concept of Information Gain and subsequently explain why both these fundamental concepts form the basis of how decision trees build themselves on the data we supply them.

Right. Let's get on with it then.

What is Entropy? In the most layman terms, Entropy is nothing but the **measure of disorder.** (You can think of it as a measure of purity as well. You'll see. I like disorder because it sounds cooler.)

The Mathematical formula for Entropy is as follows -

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

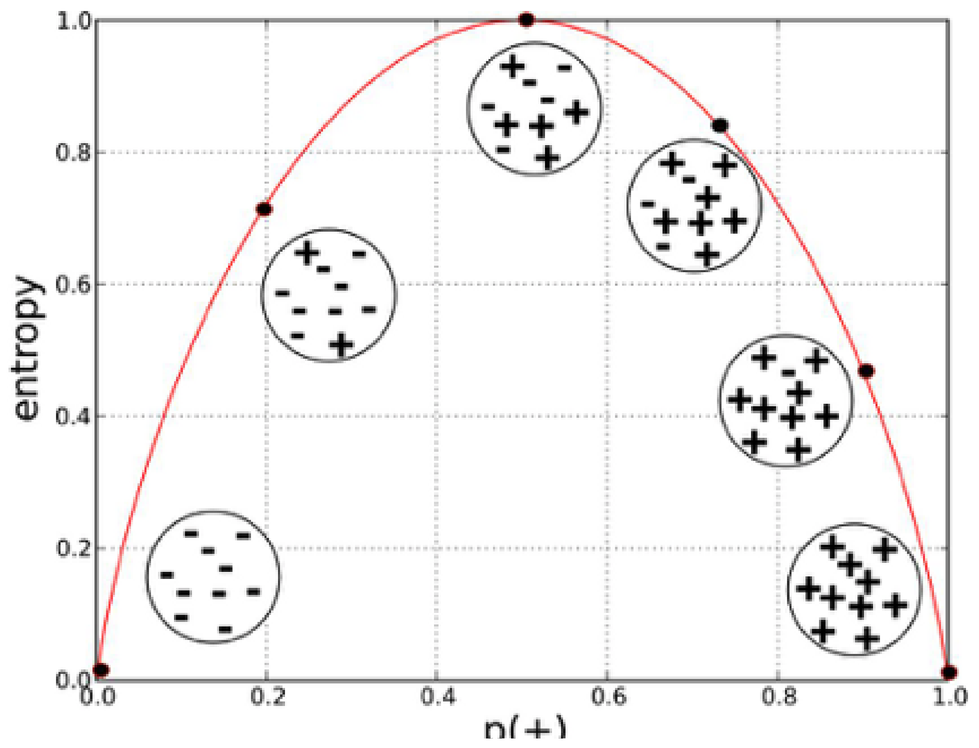
Where ‘Pi’ is simply the frequentist probability of an element/class ‘i’ in our data. For simplicity’s sake let’s say we only have two classes , a positive class and a negative class. Therefore ‘i’ here could be either + or (-). So if we had a total of 100 data points in our dataset with 30 belonging to the positive class and 70 belonging to the negative class then ‘P+’ would be 3/10 and ‘P-’ would be 7/10. Pretty straightforward.

If I was to calculate the entropy of my classes in this example using the formula above. Here’s what I would get.

$$-\frac{3}{10} \times \log_2\left(\frac{3}{10}\right) - \frac{7}{10} \times \log_2\left(\frac{7}{10}\right) \approx 0.88$$

The entropy here is approximately 0.88. This is considered a high entropy , a high level of disorder (meaning low level of purity). Entropy is measured between 0 and 1. (Depending on the number of classes in your dataset, entropy can be greater than 1 but it means the same thing , a very high level of disorder. For the sake of simplicity, the examples in this blog will have entropy between 0 and 1).

Take a look at this graph below.



Data-Analytic Thinking

The x-axis measures the proportion of data points belonging to the positive class in each bubble and the y-axis axis measures their respective entropies. Right away, you can see the inverted 'U' shape of the graph. Entropy is lowest at the extremes, when the bubble either contains no positive instances or only positive instances. That is, when the bubble is pure the disorder is 0. Entropy is highest in the middle when the bubble is evenly split between positive and negative instances. Extreme disorder , because there is no majority.

Does it matter why entropy is measured using log base 2 or why entropy is measured between 0 and 1 and not some other range? No. It's just a metric. It's not important to know how it came to be. It's important to know how to read it and what it tells us, which we just did above. Entropy is a measure of disorder or uncertainty and the goal of machine learning models and Data Scientists in general is to reduce uncertainty.

Now we know how to measure disorder. Next we need a metric to measure the reduction of this disorder in our target variable/class given additional information(features/independent variables) about it. This is where Information Gain comes in. Mathematically it can be written as:

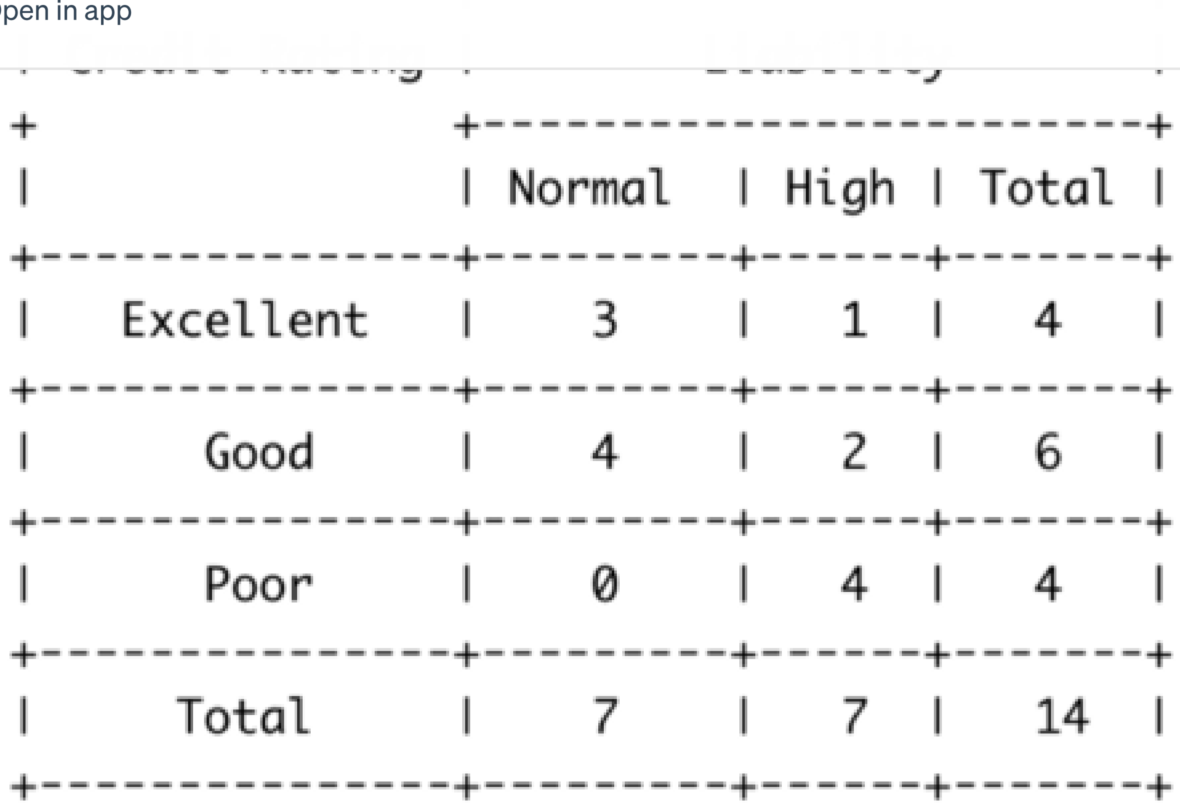
$$IG(Y, X) = E(Y) - E(Y|X)$$

Information Gain from X on Y

We simply subtract the entropy of Y given X from the entropy of just Y to calculate the reduction of uncertainty about Y given an additional piece of information X about Y. This is called Information Gain. The greater the reduction in this uncertainty, the more information is gained about Y from X.

Let me illustrate with an example using a simple contingency table and then I'll bring all of it together with how decisions trees use entropy and information gain to decide what feature to split their nodes on as they are being trained on a data set.

Example: Contingency Table



Credit Rating	Liability		
	Normal	High	Total
Excellent	3	1	4
Good	4	2	6
Poor	0	4	4
Total	7	7	14

Contingency Table

Here our target variable is Liability which can take on two values "Normal" and "High" and we only have one feature called Credit Rating which can take on values "Excellent", "Good" and "Poor". There are a total of 14 observations. 7 of them belong to the Normal Liability class and 7 belong to High Liability Class. So it's an even split by itself. Summing across the top row we can see there are 4 observations that have value Excellent for the feature credit rating. Furthermore , I can also see how my target variable is split for "Excellent" Credit Rating. For observations that take have value "Excellent" for their credit rating there are 3 that belong to the Normal Liability class and only 1 that belongs to the High Liability class. I can similarly figure out such values for other values of Credit Rating from the contingency table.

For this illustration , I will use this contingency table to calculate the entropy of our target variable by itself and then calculate the entropy of our target variable given additional information about the feature, credit rating. This will allow me to calculate how much additional information does "Credit Rating" provide for my target variable "Liability".

Let's get to it then.

$$= -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right)$$

$$= 1$$

The entropy of our target variable is 1, at maximum disorder due to the even split between class label "Normal" and "High". Our next step is to calculate the entropy of our target variable Liability given additional information about credit score. For this we will calculate the entropy for Liability for each value of Credit Score and add them using a weighted average of the proportion of observations that end up in each value. Why we use a weighted average will become clearer when we discuss this in the context of decision trees.

$$E(Liability \mid CR = Excellent) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) \approx 0.811$$

$$E(Liability \mid CR = Good) = -\frac{4}{6}\log_2\left(\frac{4}{6}\right) - \frac{2}{6}\log_2\left(\frac{2}{6}\right) \approx 0.918$$

$$E(Liability \mid CR = Poor) = -0\log_2(0) - \frac{4}{4}\log_2\left(\frac{4}{4}\right) = 0$$

$$Weighted\ Average:$$

$$E(Liability \mid CR) = \frac{4}{14}\times 0.811 + \frac{6}{14}\times 0.918 + \frac{4}{14}\times 0$$

We got the entropy for our target variable given the feature Credit Rating. Now we can compute the Information Gain on Liability from Credit Rating to see how informative this feature is.

$$Information\ Gain:$$

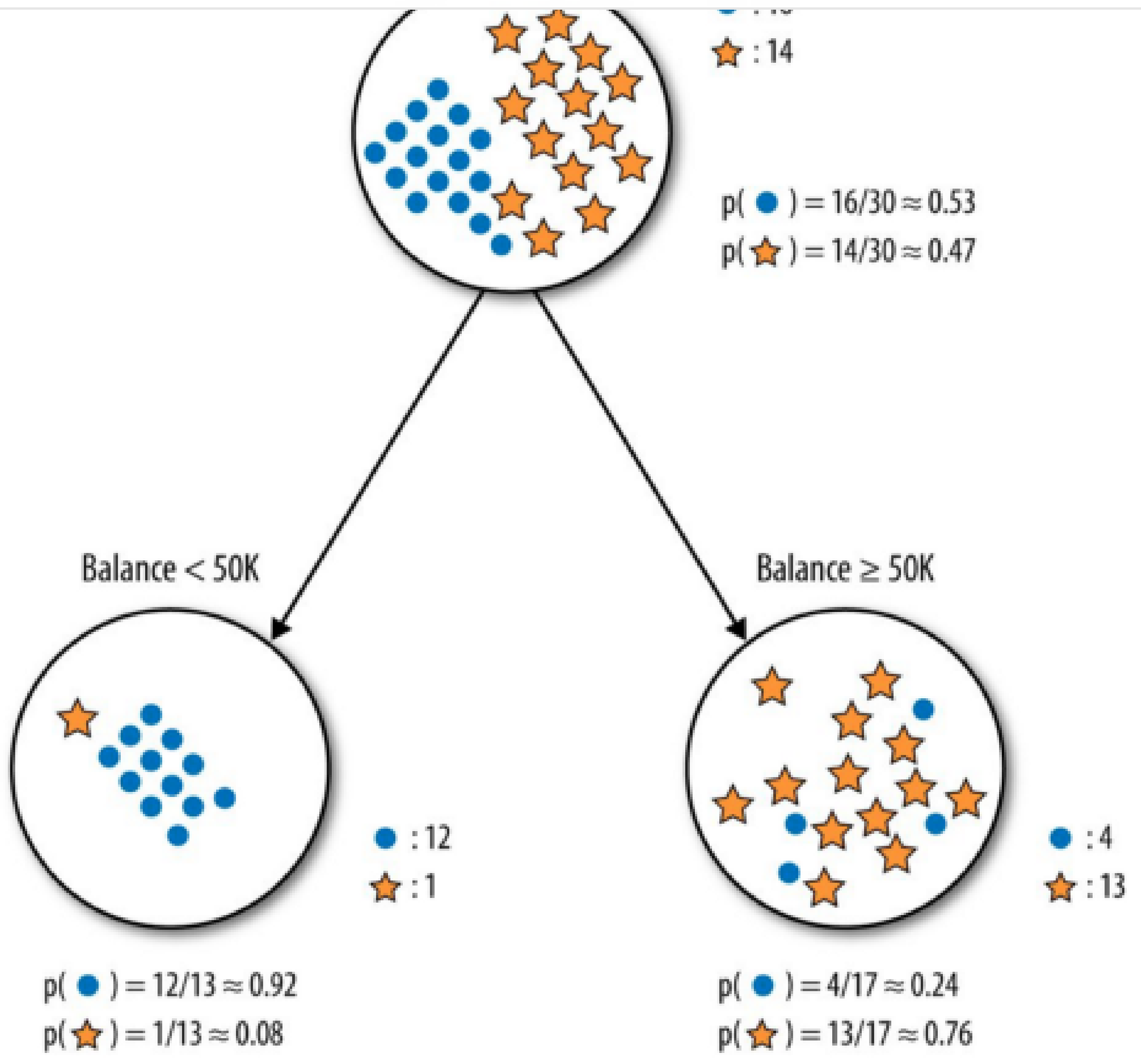
$$\begin{aligned} IG(Liability,\ CR) &= E(Liability) - E(Liability \mid CR) \\ &= 1 - 0.625 \\ &= 0.375 \end{aligned}$$

Knowing the Credit Rating helped us reduce the uncertainty around our target variable, Liability! Isn't that what a good feature is supposed to do? Provide us information about our target variable? Well that's exactly how and why decision trees use entropy and information gain to determine which feature to split their nodes on to get closer to predicting the target variable with each split and also to determine when to stop splitting the tree! (in addition to hyper-parameters like max depth of course). Let's see this in action with another example using decision trees.

Example: Decision Tree

Consider an example where we are building a decision tree to predict whether a loan given to a person would result in a write-off or not. Our entire population consists of 30 instances. 16 belong to the write-off class and the other 14 belong to the non-write-off class. We have two features, namely "Balance" that can take on two values -> "< 50K" or ">50K" and "Residence" that can take on three values -> "OWN", "RENT" or "OTHER". I'm going to show you how a decision tree algorithm would decide what attribute to split on first and what feature provides more information, or reduces more uncertainty about our target variable out of the two using the concepts of Entropy and Information Gain.

Feature 1: Balance



Provost, Foster; Fawcett, Tom. Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking

The dots are the data points with class right-off and the stars are the non-write-offs. Splitting the parent node on attribute balance gives us 2 child nodes. The left node gets 13 of the total observations with 12/13 (0.92 probability) observations from the write-off class and only 1/13(0.08 probability) observations from the non-write of class. The right node gets 17 of the total observation with 13/17(0.76 probability) observations from the non-write-off class and 4/17 (0.24 probability) from the write-off class.

Let’s calculate the entropy for the parent node and see how much uncertainty the tree can reduce by splitting on Balance.

$$E(\,Balance\,<\,50K)\;=\;-\frac{12}{13}\log_2\left(\frac{12}{13}\right)-\frac{1}{13}\log_2\left(\frac{1}{13}\right)\approx 0.39$$

$$E(\,Balance\,>\,50K)\;=\;-\frac{4}{17}\log_2\left(\frac{4}{17}\right)-\frac{13}{17}\log_2\left(\frac{13}{17}\right)\approx 0.79$$

$$Weighted\ Average\ of\ \ entropy\ for\ each\ node:$$

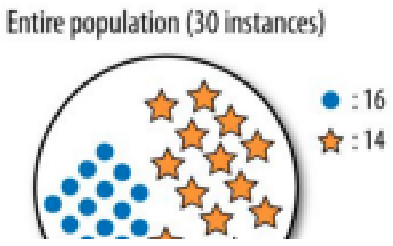
$$\begin{aligned}E(\,Balance) &= \frac{13}{30}\times 0.39 + \frac{17}{30}\times 0.79\\ &= 0.62\end{aligned}$$

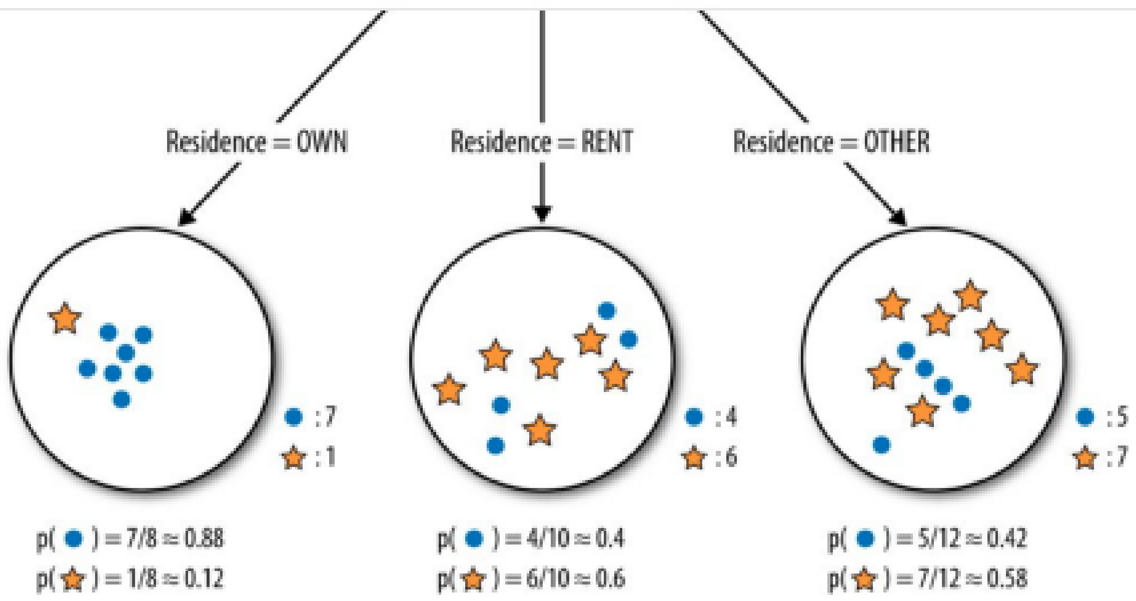
$$Information\ Gain:$$

$$\begin{aligned}IG(\,Parent,\ Balance) &= E(\,Parent) - E(\,Balance)\\ &= 0.99 - 0.62\\ &= 0.37\end{aligned}$$

Splitting on feature ,“Balance” leads to an information gain of 0.37 on our target variable. Let’s do the same thing for feature, “Residence” to see how it compares.

Feature 2: Residence





Provost, Foster; Fawcett, Tom. Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking

Splitting the tree on Residence gives us 3 child nodes. The left child node gets 8 of the total observations with 7/8 (0.88 probability) observations from the write-off class and only 1/8 (0.12 probability) observations from the non-write-off class. The middle child nodes gets 10 of the total observations with 4/10 (0.4 probability) observations of the write-off class and 6/10(0.6 probability) observations from the non-write-off class. The right child node gets 12 of the total observations with 5/12 (0.42 probability) observations from the write-off class and 7/12 (0.58) observations from the non-write-off class. We already know the entropy for the parent node. We simply need to calculate the entropy after the split to compute the information gain from “Residence”

$$E(\,Residence\; = \;OWN) \; = \; -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) \;\approx 0.54$$

$$E(\,Residence\; = \;RENT\,) \; = \; -\frac{4}{10}\log_2\left(\frac{4}{10}\right) - \frac{6}{10}\log_2\left(\frac{6}{10}\right) \;\approx 0.97$$

$$E(\,Residence\; = \;OTHER) \; = \; -\frac{5}{12}\log_2\left(\frac{5}{12}\right) - \frac{7}{12}\log_2\left(\frac{7}{12}\right) \;\approx 0.98$$

$$E(\,Residence\,) = \frac{8}{30} \times 0.54 + \frac{10}{30} \times 0.97 + \frac{12}{30} \times 0.98 = 0.86$$

$Information\ Gain:$

$$\begin{aligned} IG(\,Parent,\ Residence\,) &= E(\,Parent\,) - E(\,Residence\,) \\ &= 0.99 - 0.86 \\ &= 0.13 \end{aligned}$$

The information gain from feature, Balance is almost 3 times more than the information gain from Residence! If you go back and take a look at the graphs you can see that the child nodes from splitting on Balance do seem purer than those of Residence. However the left most node for residence is also very pure but this is where the weighted averages come in play. Even though that node is very pure, it has the least amount of the total observations and a result contributes a small portion of it’s purity when we calculate the total entropy from splitting on Residence. This is important because we’re looking for overall informative power of a feature and we don’t want our results to be skewed by a rare value in a feature.

By itself the feature, Balance provides more information about our target variable than Residence. It reduces more disorder in our target variable. A decision tree algorithm would use this result to make the first split on our data using Balance. From here on, the decision tree algorithm would use this process at every split to decide what feature it is going to split on next. In a real world scenario , with more than two features the first split is made on the most informative feature and then at every split the information gain for each additional feature needs to be recomputed because it would not be the same as the information gain from each feature by itself. The entropy and information gain would have to be calculated after one or more splits have already been made which would change the results. A decision tree would repeat this process as it grows deeper and deeper till either it reaches a pre-defined depth or no additional split can result in a higher information gain beyond a certain threshold which can also usually be specified as a hyper-parameter!

ensemble decides on the best order of features to split on and decides when to stop when it trains itself on given data. If you every have to explain the intricacies of how decision trees work to someone, hopefully you won't do too bad.

I hope you were able to get some value from this post. If there is anything that I missed or something was inaccurate or if you have absolutely any feedback , please let me know in the comments. I would greatly appreciate it. Thank you.

Machine Learning Data Science Statistics Towards Data Science