**Decision Trees Classification**

Okay now let’s get to how we actually construct a tree.

**Constructing the Decision Tree**

So below I’ll describe the decision tree. So we start by putting all the data into the root node. And then we’re basically looking to categorize the data into categories/leaves, that maximize the information gain/minimize gini impurity loss or whatever. I’m going to presume we’re using information gain for the sake of discussion.

Diagram

Description automatically generated

So we want to find leaves, Lℓ, which maximize:



and then the prediction, or fit, of those leaves will be whatever is the most frequent value of Yi in those leaves, i.e., the mode:



If we used the diagram above, then L = {A1, B1, B2}. We will typically use a greedy algorithm to find these leaves. It won’t be guaranteed to produce the correct result, but I don’t think we actually care, because we won’t usually fit a tree precisely to the data anyway (that’d be overfitting). The way we use the greedy algorithm is as follows. Say we’re at some node, A2 (and this could be the root of course). In this node are all rows with value A = A2. These rows have outcomes Yj. We can calculate the entropy of this node S(Y|A2) = -Σj=1n\_A2 P(Yj|A2)logP(Yj|A2) as shown. And the prediction of this node would be fA2 = mode(Yj|A2), i.e., the most frequent value of Y in this subset A2. To make further progress classifying the data, we can split Yj’s in A2 into two groups according to the B values: B1, B2. And we can calculate the entropy S(Y|B1) of the group in B1, and the entropy S(Y|B2) of the group in B2. Averaged together this gives the total entropy S(Y|B) = P(B1)S(Y|B1) + P(B2)S(Y|B2). Note P(B1,2) would be conditional probabilities, using A2 as the reference set. So really, we’d write this as P(B1,2|A2), say. But don’t want the notation to get out of hand. And then we calculate the information gain for this split: IG(B) = S(Y|B) – S(Y|A2). And we choose the column B for which IG(B) is largest. And we continue in like fashion until we’ve broken the tree down as far as we want to go. And certainly if we should find that the information gain were ever negative, then we wouldn’t bother making the split.

Note, BTW, that R, A2 are internal nodes, also called split nodes. And A1, B1, B2 are called terminal, or leaf nodes.

The depth of a decision tree is the length of the longest path from the root to a leaf node. So the depth of the tree above would be 2.

And now let’s do some examples ourselves.

**Example**

Let’s do an example from Stat Quest. We want to predict whether someone will love Cool as Ice movie, given information from the three categories preceding it.

|  |  |  |  |
| --- | --- | --- | --- |
| **Loves Popcorn** | **Loves Soda** | **Age** | **Loves Cool as Ice** |
| Yes | Yes | 7 | No |
| Yes | No | 12 | No |
| No | Yes | 18 | Yes |
| No | Yes | 35 | Yes |
| Yes | Yes | 38 | Yes |
| Yes | No | 50 | No |
| No | No | 83 | No |

We’ll use the Gini Impurity measure, ‘cause he does. And turns out the Gini Impurity measure works better as a decision making guide than entropy sometimes. Don’t know a general rule apropros which measure to use when.

**Zeroth level**

So first we’ll calculate the Gini Impurity of Loves Cool as Ice:



and the most common value of Loves Cool as Ice is No. So our root is:

A picture containing graphical user interface

Description automatically generated

Moving on,

**First Level**

And now we’ll do Loves Popcorn. So separating values entries by their Loves Popcorn values,

|  |  |  |  |
| --- | --- | --- | --- |
| **Loves Popcorn** | **Loves Soda** | **Age** | **Loves Cool as Ice** |
| Yes | Yes | 7 | No |
| Yes | No | 12 | No |
| No | Yes | 18 | Yes |
| No | Yes | 35 | Yes |
| Yes | Yes | 38 | Yes |
| Yes | No | 50 | No |
| No | No | 83 | No |



Now we’ll do it for Loves Soda,

|  |  |  |  |
| --- | --- | --- | --- |
| **Loves Popcorn** | **Loves Soda** | **Age** | **Loves Cool as Ice** |
| Yes | Yes | 7 | No |
| Yes | No | 12 | No |
| No | Yes | 18 | Yes |
| No | Yes | 35 | Yes |
| Yes | Yes | 38 | Yes |
| Yes | No | 50 | No |
| No | No | 83 | No |

And we have,



So Loves Soda has a lower impurity value. This is due to fact that Loves Soda is more correlated with Loves Cool as Ice (as can see that P(C=yes|S=yes) = 1 for instance.

Now we come to the age category. This is a little more complicated to handle for two reasons. First there are a lot of different ages. We *could* treat each age in our data as a distinct category and get a perfect decision tree for *our* data. But then it wouldn’t have any predictive power for someone with a different age. Second fitting our decision tree to so many different ages would likely result in overfitting the data – like if had a 2D logistic regression curve and we just drew a squiggle surface around all the data of one class to separate it from the other class. So to address both issues, we’ll want to break the age into categories. He uses two age categories in his example: age < critical age, and age > critical age. I suppose we could use three age categories instead, if we had a meaningful reason to do so. But without one, we’ll just default to two. Next question is, ‘what should be our critical age?’. We try different ones, and use the one which gives us the best GI value. Our critical age options are basically the ages directly in between any two adjacent ages in the column (arranged in ascending or descending age order), so: 9.5, 15, 26.5, 36.5, 44, 66.5. Turns out the second one, 15, is the best one, i.e., has the lowest GI score. So let’s calculate it. First splitting the data by age,

|  |  |  |  |
| --- | --- | --- | --- |
| **Loves Popcorn** | **Loves Soda** | **Age** | **Loves Cool as Ice** |
| Yes | Yes | 7 | No |
| Yes | No | 12 | No |
| No | Yes | 18 | Yes |
| No | Yes | 35 | Yes |
| Yes | Yes | 38 | Yes |
| Yes | No | 50 | No |
| No | No | 83 | No |

And now calculating,



Okay, well the column with the lowest GI, and therefore highest GIL is the Loves Soda column. So we’ll start with that. So based on our Loves Soda split,

|  |  |  |  |
| --- | --- | --- | --- |
| **Loves Popcorn** | **Loves Soda** | **Age** | **Loves Cool as Ice** |
| Yes | Yes | 7 | No |
| Yes | No | 12 | No |
| No | Yes | 18 | Yes |
| No | Yes | 35 | Yes |
| Yes | Yes | 38 | Yes |
| Yes | No | 50 | No |
| No | No | 83 | No |

Our Decision Tree would look like this so far (C = Loves Cool as Ice, S = Loves Soda):

Diagram

Description automatically generated

And on to the,

**Second Level**

And now we’ll work on the second level. If Love Soda = True, then we have the following data left, in blue,

|  |  |  |  |
| --- | --- | --- | --- |
| **Loves Popcorn** | **Loves Soda** | **Age** | **Loves Cool as Ice** |
| Yes | Yes | 7 | No |
| Yes | No | 12 | No |
| No | Yes | 18 | Yes |
| No | Yes | 35 | Yes |
| Yes | Yes | 38 | Yes |
| Yes | No | 50 | No |
| No | No | 83 | No |

What is the GI(C) now? Note, this would really be GI(C|S=yes), but in interest of space, I’m going to leave S = True implicit. So we have:



And let’s see if assuming popcorn or age status lowers the GI(C). So we’ll look at Loves Popcorn first. Splitting the remaining blue data up by Loves Popcorn values,

|  |  |  |  |
| --- | --- | --- | --- |
| **Loves Popcorn** | **Loves Soda** | **Age** | **Loves Cool as Ice** |
| Yes | Yes | 7 | No |
| Yes | No | 12 | No |
| No | Yes | 18 | Yes |
| No | Yes | 35 | Yes |
| Yes | Yes | 38 | Yes |
| Yes | No | 50 | No |
| No | No | 83 | No |

And now calculating…all those conditionals in the probabilities below should have as well a Loves Soda = yes label in the conditional. But that’s too much to write. So it’s implicit.



And now we’ll look at the second category, age. Like before, we’ll want to split the age groups into two categories: those below a critical age, and those above it. The possible critical ages are the averages of adjacent age values in the age column: 12.5, 26.5, 36.5. We’ll look at the first one. So splitting data by age in relation to 12.5,

|  |  |  |  |
| --- | --- | --- | --- |
| **Loves Popcorn** | **Loves Soda** | **Age** | **Loves Cool as Ice** |
| Yes | Yes | 7 | No |
| Yes | No | 12 | No |
| No | Yes | 18 | Yes |
| No | Yes | 35 | Yes |
| Yes | Yes | 38 | Yes |
| Yes | No | 50 | No |
| No | No | 83 | No |

And calculating the impurity,



So Age (with a critical age of 12.5) has the lowest impurity value possible. And this is because, given the Loves Soda value of True, Age is a perfect predictor of Loves Cool as Ice. So we’ll use this split.

|  |  |  |  |
| --- | --- | --- | --- |
| **Loves Popcorn** | **Loves Soda** | **Age** | **Loves Cool as Ice** |
| Yes | Yes | 7 | No |
| Yes | No | 12 | No |
| No | Yes | 18 | Yes |
| No | Yes | 35 | Yes |
| Yes | Yes | 38 | Yes |
| Yes | No | 50 | No |
| No | No | 83 | No |

And our Decision Tree comes to:

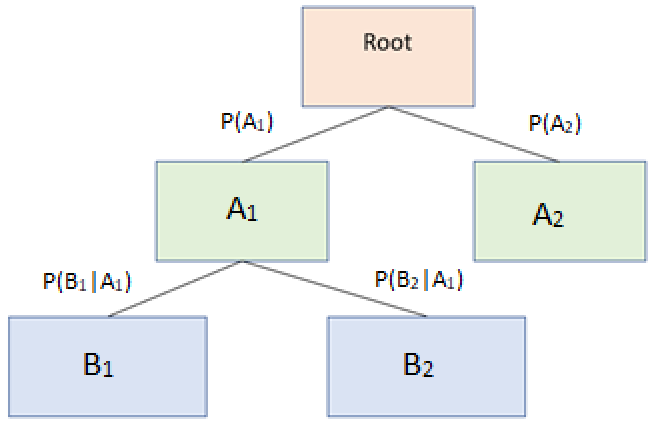
Diagram

Description automatically generated

And we’re done.

**Appendix**

I’ve got a few remaining questions. So what is the entropy of an entire decision tree? Say we have an outcome, C, dependent on two variables, A, B. And let’s say that A and B are both binary events. And say we write down the following decision tree. And I’ve added a probability label for branching line, so we can see the overall probability of each event – it would just be the product of probabilities along the path to that event.



Okay, well I think the overall entropy of the tree would be:



where we sum over the conditional entropies of all the terminal leaves, times the probability of those leaves. Or in our particular case, we could say:



and I think the entropy of just the first level, i.e.,

A diagram of a root system

Description automatically generated

would be:



The difference between the two should be the information gain obtained by making that split on node A1 along the B variable. This would be:



This is a little bit different than what we had before. Before, when calculating information gain, we left off the overall P(A1) that’s multiplying everything. Since it’s an overall factor, it won’t have any role in determining how, if at all, we split the A1 node. So I guess that’s why people don’t care about it. But if we wanted to compare two trees’ overall entropy, to decide if one is better than the other, say, then it does play a role. I imagine we could do a similar sort of calculation to compute a tree’s overall gini impurity. So,



and in our example, for instance,



etc. Well here’s an example. This is a tree I ended up with at some point. I’ve labeled the probability of all the branches (can get from the samples values), and it (sklearn) calculated the gini impurity of each node.

A diagram of a data flow

Description automatically generated with medium confidence

The total gini impurity of the tree is:



BTW, this is how you read the sklearn output,

**samples** is number, n, of samples in that node of the tree, before present decision is made. Can see we start with 222. Then it splits into the daughter nodes/leaves, the sum of which must add up to the parent samples value.

**value** = [y0, y1] = [# samples which correspond to y = 0 (no hd in this case), # samples which correspond to y = 1 (hd in this case)]. Sum of values must add up to n = samples. Assuming this is a binary classification situation.

**gini** is short for gini impurity = 1 – (y0/n)2 – (y1/n)2, where n = # samples. It gives an indication of the spread of yes’s, no’s in that node.

**class** = class\_0 or class\_1, depending on whether y0 or y1 is larger. And it is colored more orange or more blue in proportion to how much it leans one way or the other.

**thai\_7.0** <= 0.5, **old\_peak** <=2.7, **ca** <= 0.5 are all decisions. If true then go left; if false then go right.

**Feature Importance**

The importance of a feature is more or less given by how correlated it is with the outcome. The way it’s assessed is by looking at the tree and calculating the information gain when splitting on that feature. For instance, consider this node,

A diagram of a number

Description automatically generated with medium confidence

splitting on thai\_7.0 < 0.5 results in an information gain of,



We can do likewise for all the features, and then assign each feature a weight,



If a feature’s importance is low, then we should consider eliminating that variable from the model. It might be irrelevant to the underlying ‘physics’, and just be fitting noise.