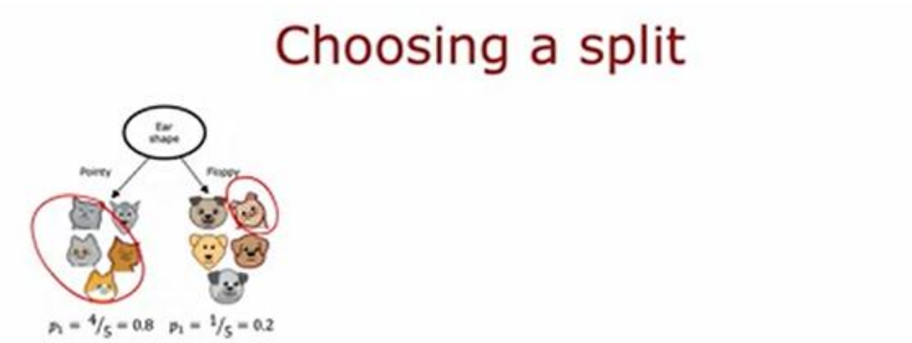
Choosing a split system: Gaining information

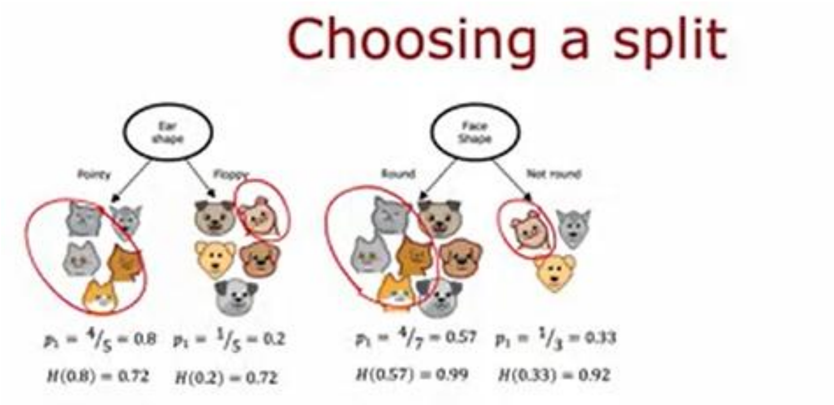
When building a decision tree, the way we'll decide what feature to split on at a node will be based on what choice of feature reduces entropy the most. Reduces entropy or reduces impurity, or maximizes purity. In decision tree learning, the reduction of entropy is called information gain. Let's take a look, in this video, at how to compute information gain and therefore choose what features to use to split on at each node in a decision tree.

Let's use the example of deciding what feature to use at the root node of the decision tree we were building just now for recognizing cats versus not cats. If we had split using their ear shape feature at the root node, this is what we would have gotten, five examples on the left and five on the right. On the left, we would have four out of five cats, so P1 would be equal to 4/5 or 0.8.



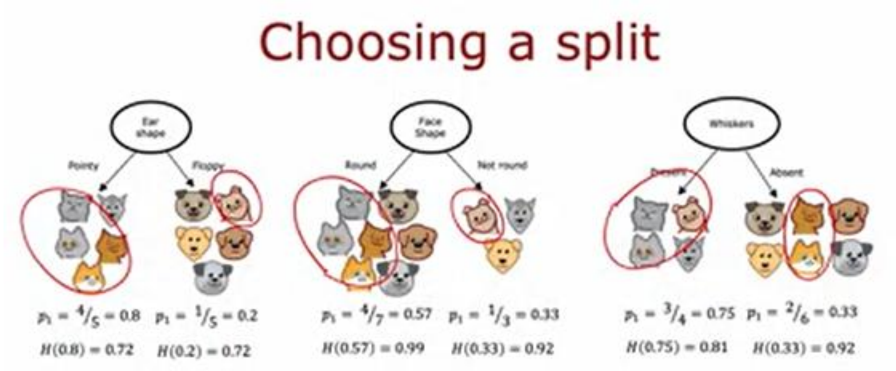
On the right, one out of five are cats, so P1 is equal to 1/5 or 0.2. If you apply the entropy formula from the last video to this left subset of data and this right subset of data, we find that the degree of impurity on the left is entropy of 0.8, which is about 0.72, and on the right, the entropy of 0.2 turns out also to be 0.72. This would be the entropy at the left and right subbranches if we were to split on the ear shape feature.

One other option would be to split on the face shape feature. If we'd done so then on the left, four of the seven examples would be cats, so P1 is 4/7 and on the right, 1/3 are cats, so P1 on the right is 1/3. The entropy of 4/7 and the entropy of 1/3 are 0.99 and 0.92. So the degree of impurity in the left and right nodes seems much higher, 0.99 and 0.92 compared to 0.72 and 0.72.



Finally, the third possible choice of feature to use at the root node would be the whiskers feature in which case you split based on whether whiskers are present or absent. In this case, P1 on the left is 3/4, P1 on the right is 2/6, and the entropy values are as follows. The key question we need to answer is, given these three options of a feature to use at the root node, which one do we think works best? It turns out that rather than looking at these entropy numbers and comparing them, it would be useful to take a weighted average of them, and here's what I mean.

If there's a node with a lot of examples in it with high entropy that seems worse than if there was a node with just a few examples in it with high entropy. Because entropy, as a measure of impurity, is worse if you have a very large and impure dataset compared to just a few examples and a branch of the tree that is very impure. The key decision is, of these three possible choices of features to use at the root node, which one do we want to use?



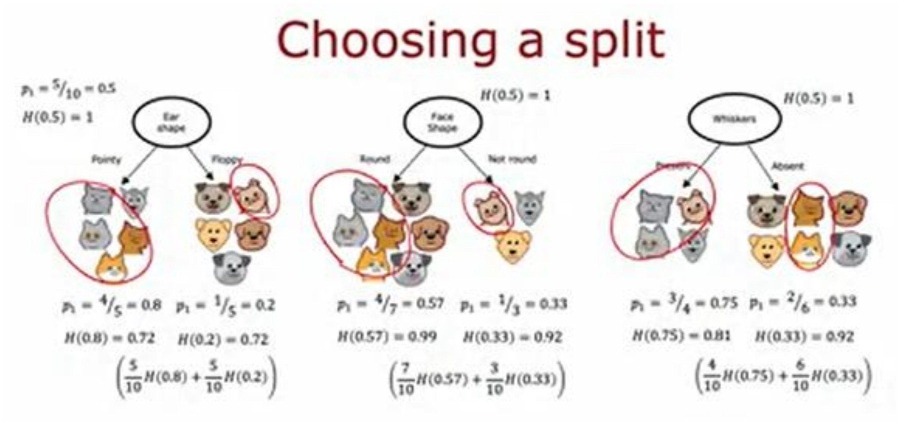
Associated with each of these splits is two numbers, the entropy on the left sub-branch and the entropy on the right sub-branch. In order to pick from these, we like to actually combine these two numbers into a single number. So you can just choose from these three choices, which one does best? The way we're going to combine these two numbers is by taking a weighted average.

Because how important it is to have low entropy in, say, the left or right sub-branch also depends on how many examples went into the left or right sub-branch. Because if there are lots of examples in, say, the left sub-branch then it seems more important to make sure that that left sub-branch's entropy value is low. In this example we have, five of the 10 examples went to the left sub-branch, so we can compute the weighted average as 5/10 times the entropy of 0.8, and then add to that 5/10 examples also went to the right sub-branch, plus 5/10 times the entropy of 0.2.

Now, for this example in the middle, the left sub-branch had received seven out of 10 examples. and so we're going to compute 7/10 times the entropy of 0.57 plus, the right sub-branch had three out of 10 examples, so plus 3/10 times entropy of 0.3 of 1/3. Finally, on the right, we'll compute 4/10 times entropy of 0.75 plus 6/10 times entropy of 0.33.

The way we will choose a split is by computing these three numbers and picking whichever one is lowest because that gives us the left and right sub-branches with the lowest average weighted entropy. In the way that decision trees are built, we're actually going to make one more change to these formulas to stick to the convention in decision tree building, but it won't actually change the outcome.

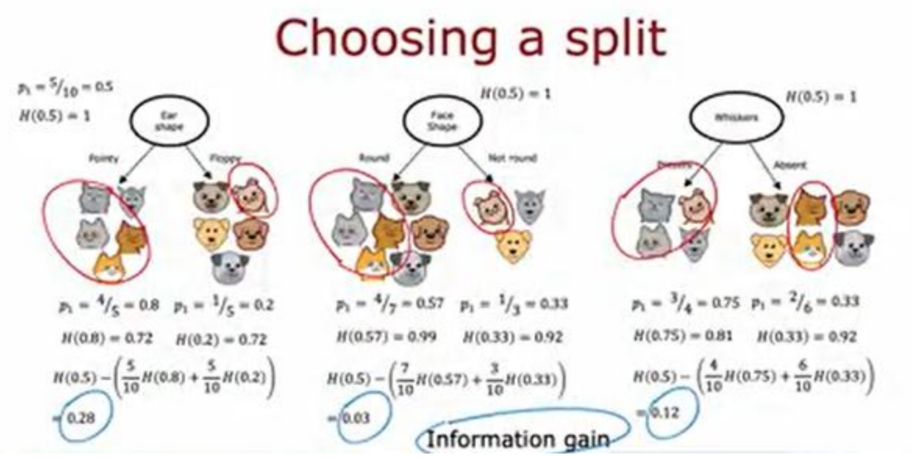
Which is rather than computing this weighted average entropy, we're going to compute the reduction in entropy compared to if we hadn't split at all. If we go to the root node, remember that the root node we have started off with all 10 examples in the root node with five cats and dogs, and so at the root node, we had p_1 equals 5/10 or 0.5. The entropy of the root nodes, entropy of 0.5 was actually equal to 1.



This was maximum impurity because it was five cats and five dogs. The formula that we're actually going to use for choosing a split is not this weighted entropy at the left and right sub-branches, instead is going to be the entropy at the root node, which is entropy of 0.5, then minus this formula. In this example, if you work out the math, it turns out to be 0.28.

For the face shape example, we can compute entropy of the root node, entropy of 0.5 minus this, which turns out to be 0.03, and for whiskers, compute that, which turns out to be 0.12. These numbers that we just calculated, 0.28, 0.03, and 0.12, these are called the information gain, and what it measures is the reduction in entropy that you get in your tree resulting from making a split.

Because the entropy was originally one at the root node and by making the split, you end up with a lower value of entropy and the difference between those two values is a reduction in entropy, and that's 0.28 in the case of splitting on the ear shape. Why do we bother to compute reduction in entropy rather than just entropy at the left and right sub-branches?

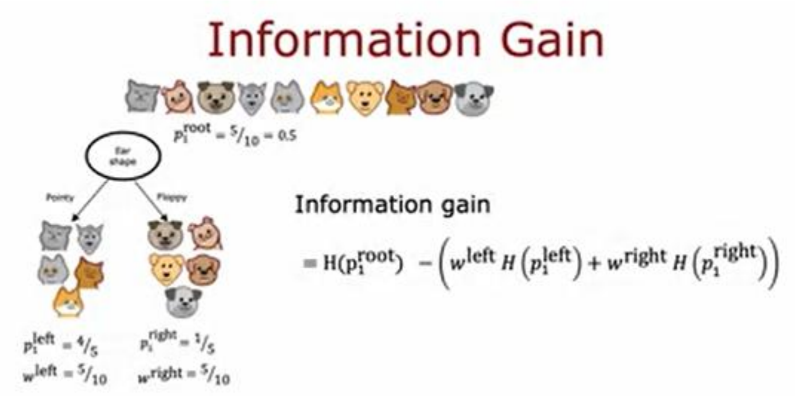


It turns out that one of the stopping criteria for deciding when to not bother to split any further is if the reduction in entropy is too small. In which case you could decide, you're just increasing the size of the tree unnecessarily and risking overfitting by splitting and just decide to not bother if the reduction in entropy is too small or below a threshold. In this other example, spitting on ear shape results in the biggest reduction in entropy, 0.28 is bigger than 0.03 or 0.12 and so we would choose to split onto ear shape feature at the root node. On the next slide, let's give a more formal definition of information gain.

By the way, one additional piece of notation that we'll also introduce in the next slide is these numbers, 5/10 and 5/10. I'm going to call this w^left because that's the fraction of examples that went to the left branch, and I'm going to call this w^right because that's the fraction of examples that went to the right branch. whereas for this another example, w^left would be 7/10, and w^right will be 3/10.

Let's now write down the general formula for how to compute information gain. Using the example of splitting on the ear shape feature, let me define p_1^left to be equal to the fraction of examples in the left subtree that have a positive label, that are cats. In this example, p_1^left will be equal to 4/5. Also, let me define w^left to be the fraction of examples of all of the examples of the root node that went to the left sub-branch, and so in this example, w^left would be 5/10.

Similarly, let's define p_1^right to be of all the examples in the right branch. The fraction that are positive examples and so one of the five of these examples being cats, there'll be 1/5, and similarly, w^right is 5/10 the fraction of examples that went to the right sub-branch. Let's also define p_1^root to be the fraction of examples that are positive in the root node.



In this case, this would be 5/10 or 0.5. Information gain is then defined as the entropy of p_1^root, so what's the entropy at the root node, minus that weighted entropy calculation that we had on the previous slide, minus w^left those were 5/10 in the example, times the entropy applied to p_1^left, that's entropy on the left sub-branch, plus w^right the fraction of examples that went to the right branch, times entropy of p_1^right.

With this definition of entropy, and you can calculate the information gain associated with choosing any particular feature to split on in the node. Then out of all the possible futures, you could choose to split on, you can then pick the one that gives you the highest information gain. That will result in, hopefully, increasing the purity of your subsets of data that you get on the left and right sub-branches of your decision tree and that will result in choosing a feature to split on that increases the purity of your subsets of data in both the left and right sub-branches of your decision tree.

Now that you know how to calculate information gain or reduction in entropy, you know how to pick a feature to split on another node. Let's put all the things we've talked about together into the overall algorithm for building a decision tree given a training set.