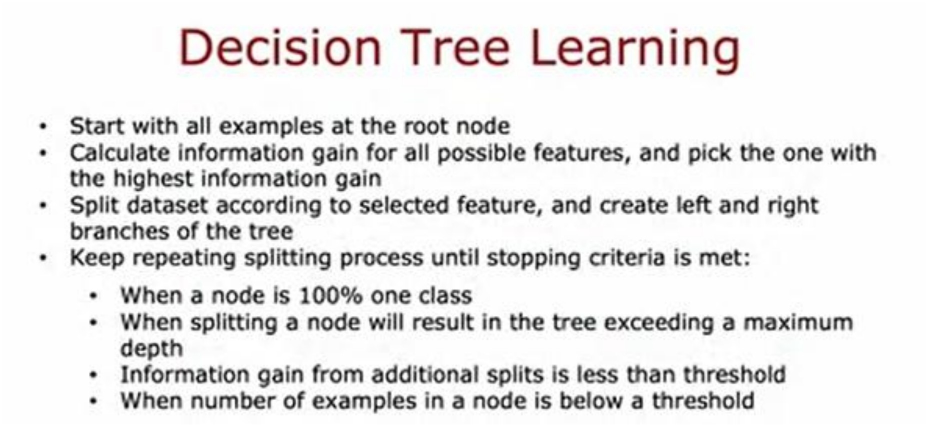
The implementation

The information gain criteria lets you decide how to choose one feature to split a one-node. Let's take that and use that in multiple places through a decision tree in order to figure out how to build a large decision tree with multiple nodes. Here is the overall process of building a decision tree. Starts with all training examples at the root node of the tree and calculate the information gain for all possible features and pick the feature to split on, that gives the highest information gain.

Having chosen this feature, you would then split the dataset into two subsets according to the selected feature, and create left and right branches of the tree and send the training examples to either the left or the right branch, depending on the value of that feature for that example. This allows you to have made a split at the root node. After that, you will then keep on repeating the splitting process on the left branch of the tree, on the right branch of the tree and so on.

Keep on doing that until the stopping criteria is met. Where the stopping criteria can be, when a node is 100 percent a single clause, someone has reached entropy of zero, or when further splitting a node will cause the tree to exceed the maximum depth that you had set, or if the information gain from an additional splits is less than the threshold, or if the number of examples in a nodes is below a threshold.



Decision Tree Learning

- Start with all examples at the root node
- Calculate information gain for all possible features, and pick the one with the highest information gain
- Split dataset according to selected feature, and create left and right branches of the tree
- Keep repeating splitting process until stopping criteria is met:
  - When a node is 100% one class
  - When splitting a node will result in the tree exceeding a maximum depth
  - Information gain from additional splits is less than threshold
  - When number of examples in a node is below a threshold

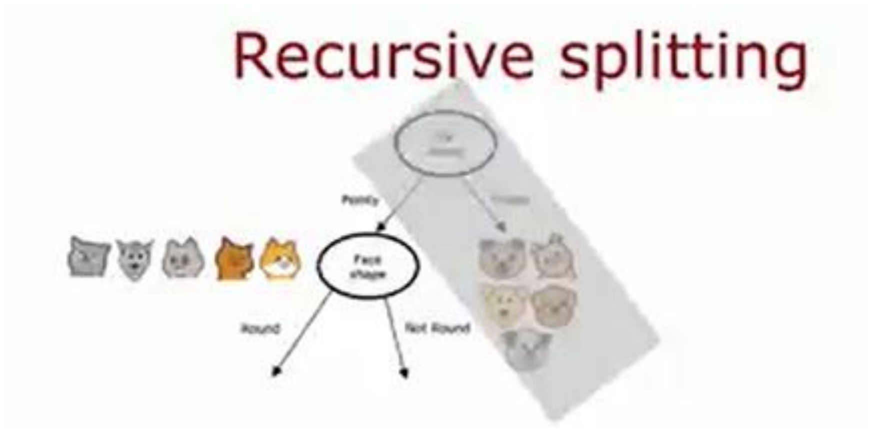
You will keep on repeating the splitting process until the stopping criteria that you've chosen, which could be one or more of these criteria is met. Let's look at an illustration of how this process will work. We started all of the examples at the root nodes and based on computing information gain for all three features, decide that ear-shaped is the best feature to split on.

Based on that, we create a left and right sub-branches and send the subsets of the data with pointy versus floppy ear to left and right sub-branches. Let me cover the root node and the right sub-branch and just focus on the left sub-branch where we have these five examples.

Let's see off splitting criteria is to keep splitting until everything in the node belongs to a single class, so either all cancel all nodes. We will look at this node and see if it meets the splitting criteria, and it does not because there is a mix of cats and dogs here. The next step is to then pick a feature to split on.

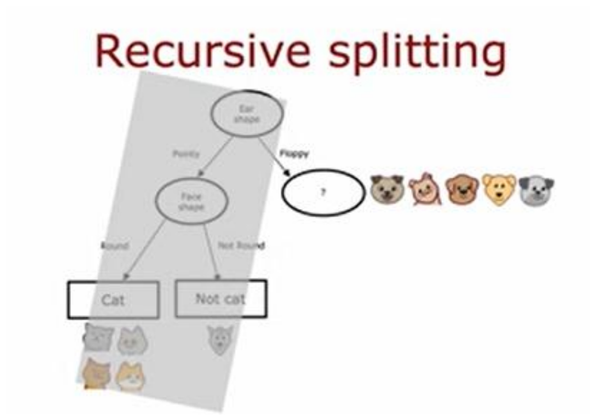
We then go through the features one at a time and compute the information gain of each of those features as if this node, were the new root node of a decision tree that was trained using just five training examples shown here. We would compute the information gain for splitting on the whiskers feature, the information gain on splitting on the face shape feature.

It turns out that the information gain for splitting on ear shape will be zero because all of these have the same point ear shape. Between whiskers and face shape, face shape turns out to have a highest information gain.



We're going to split on face shape and that allows us to build left and right sub branches as follows. For the left sub-branch, we check for the criteria for whether or not we should stop splitting and we have all cats here. The stopping criteria is met and we create a leaf node that makes a prediction of cat.

For the right sub-branch, we find that it is all dogs. We will also stop splitting since we've met the splitting criteria and put a leaf node there, that predicts not cat. Having built out this left subtree, we can now turn our attention to building the right subtree. Let me now again cover up the root node and the entire left subtree.

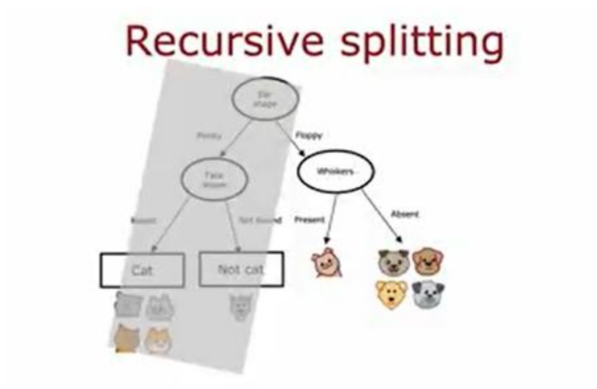


To build out the right subtree, we have these five examples here. Again, the first thing we do is check if the criteria to stop splitting has been met, their criteria being met or not. All the examples are a single class, we've not met that criteria. We'll decide to keep splitting in this right sub-branch as well.

In fact, the procedure for building the right sub-branch will be a lot as if you were training a decision tree learning algorithm from scratch, where the dataset you have understood just these five training examples. Again, computing information gain for all of the possible features to split on, you find that the whiskers feature use the highest information gain. Split this set of five examples according to whether whiskers are present or absent.

Check if the criteria to stop splitting are put in the left and right sub-branches here and decide that they are. You end up with leaf nodes that predict cat and dog cat. This is the overall process for building the decision tree. Notice that there's interesting aspects of what we've done, which is after we decided what to split on at the root node, the way we built the left subtree was by building a decision tree on a subset of five examples.

The way we built the right subtree was by, again, building a decision tree on a subset of five examples. In computer science, this is an example of a recursive algorithm. All that means is the way you build a decision tree at the root is by building other smaller decision trees in the left and the right sub-branches.

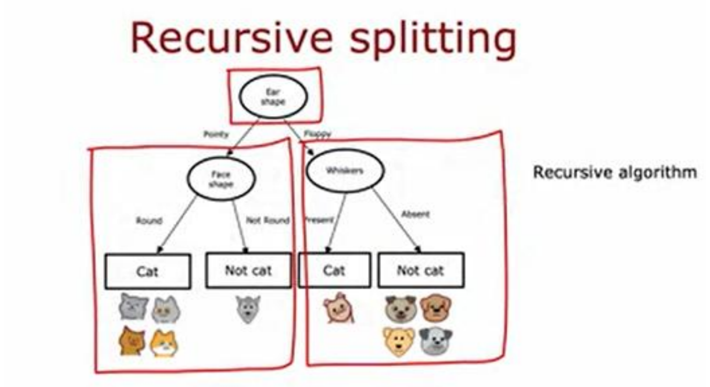


Recursion in computer science refers to writing code that calls itself. The way this comes up in building a decision tree is you build the overall decision tree by building smaller sub-decision trees and then putting them all together. That's why if you look at software implementations of decision trees, you'll see sometimes references to a recursive algorithm.

But if you don't feel like you've fully understood this concept of recursive algorithms, don't worry about it. You still be able to fully complete this week's assignments, as well as use libraries to get decision trees to work for yourself. But if you're implementing a decision tree algorithm from scratch, then a recursive algorithm turns out to be one of the steps you'd have to implement. By the way, you may be wondering how to choose the maximum depth parameter.

There are many different possible choices, but some of the open-source libraries will have good default choices that you can use. One intuition is, the larger the maximum depth, the bigger the decision tree you're willing to build. This is a bit like fitting a higher degree polynomial or training a larger neural network.

It lets the decision tree learn a more complex model, but it also increases the risk of overfitting if this fitting a very complex function to your data. In theory, you could use cross-validation to pick parameters like the maximum depth, where you try out different values of the maximum depth and pick what works best on the cross-validation set.



Although in practice, the open-source libraries have even somewhat better ways to choose this parameter for you. Or another criteria that you can use to decide when to stop splitting is if the information gained from an additional split is less than a certain threshold.

If any feature is splint on, achieves only a small reduction in entropy or a very small information gain, then you might also decide to not bother. Finally, you can also decide to stop splitting when the number of examples in the node is below a certain threshold. That's the process of building a decision tree.

Now that you've learned the decision tree, if you want to make a prediction, you can then follow the procedure that you saw in the very first video of this week, where you take a new example, say a test example, and started a route and keep on following the decisions down until you get to the leaf node, which then makes the prediction.

Now that you know the basic decision tree learning algorithm, in the next few videos, I'd like to go into some further refinements of this algorithm. So far we've only used features to take on two possible values. But sometimes you have a feature that takes on categorical or discrete values, but maybe more than two values.