ADVANCED ANALYSIS | MACHINE LEARNING

Machine Learning: Pruning Decision Trees

by Jake Hoare

You can quickly create your own decision trees in Displayr.

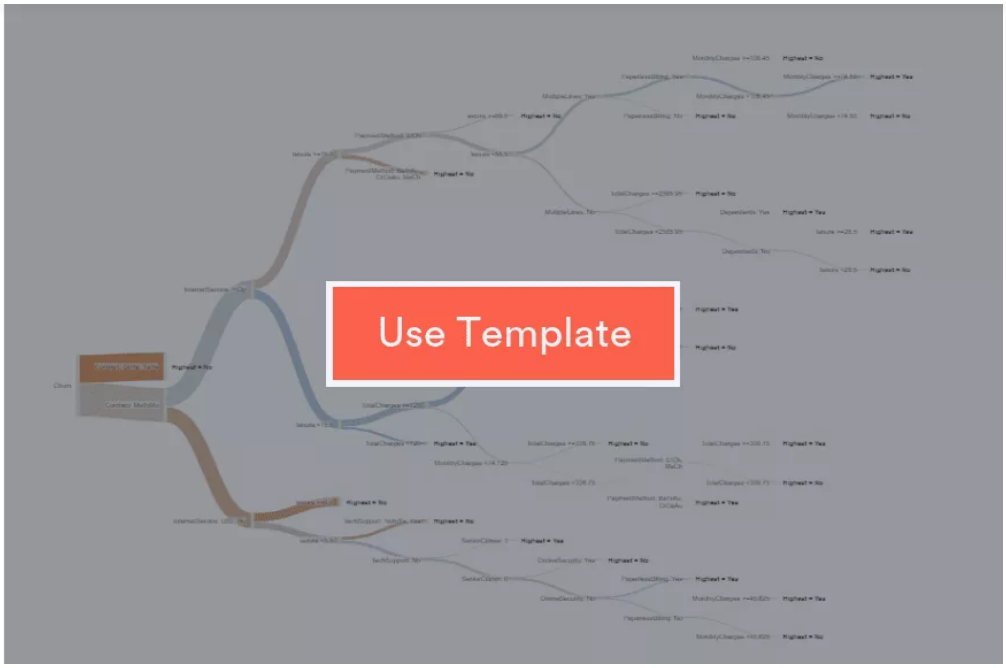
Get started here

In machine learning and data mining, pruning is a technique associated with decision trees. Pruning reduces the size of decision trees by removing parts of the tree that do not provide power to classify instances. Decision trees are the most susceptible out of all the machine learning algorithms to overfitting and effective pruning can reduce this likelihood. This post will go over two techniques to help with overfitting - pre-pruning or early stopping and post-pruning with examples.

Machine learning is a problem of trade-offs. The classic issue is overfitting versus underfitting. Overfitting happens when a model memorizes its training data so well that it is learning noise on top of the signal. Underfitting is the opposite: the model is too simple to find the patterns in the data. Simplicity versus accuracy is a similar consideration. Do you want a model that can fit onto one sheet of paper and be understood by a broad audience? Or do you want the best possible accuracy, even if it is a "black box"? In this post I am going to look at two techniques (called *pruning* and *early stopping or post-pruning and pre-pruning*) for managing these trade-offs in the context of decision trees.

The techniques I am going to describe in this post give you the power to find a model that suits your needs.

You can follow my steps or replicate them for yourself in this document. If you want to create your own decision tree, use the template below.



Classification and regression trees (CART)

CART is one of the most well-established machine learning techniques. In non-technical terms, CART algorithms works by repeatedly finding the best predictor variable to split the data into two subsets. The subsets partition the target outcome better than before the split. Pruning is a technique associated with classification and regression trees.

I am not going to go into details here about what is meant by the best predictor variable, or a better partition. Instead I am going to discuss two enhancements to that basic outline: pruning and early stopping. Sometimes these are referred to simplistically as post-pruning and pre-pruning. As the names suggest, pre-pruning or early stopping involves stopping the tree before it has completed classifying the training set and post-pruning refers to pruning the tree after it has finished. I prefer to differentiate these terms more clearly by using early-stopping and pruning.

Create your own decision tree

Pruning or post-pruning

As the name implies, pruning involves cutting back the tree. After a tree has been built (and in the absence of early stopping discussed below) it may be overfitted. The CART algorithm will repeatedly partition data into smaller and smaller subsets until those final subsets are homogeneous in terms of the outcome variable. In practice this often means that the final subsets (known as the *leaves* of the tree) each consist of only one or a few data points. The tree has learned the data exactly, but a new data point that differs very slightly might not be predicted well.

I will consider 3 pruning strategies,

- *Minimum error*. The tree is pruned back to the point where the cross-validated error is a minimum. *Cross-validation* is the process of building a tree with most of the data and then using the remaining part of the data to test the accuracy of the decision tree.

- *Smallest tree*. The tree is pruned back slightly further than the minimum error. Technically the pruning creates a decision tree with cross-validation error within 1 standard error of the minimum error. The smaller tree is more intelligible at the cost of a small increase in error.

- None.

Create your own decision tree

Early stopping or pre-pruning

An alternative method to prevent overfitting is to try and stop the tree-building process early, before it produces leaves with very small samples. This heuristic is known as *early stopping* but is also sometimes known as pre-pruning decision trees.

At each stage of splitting the tree, we check the cross-validation error. If the error does not decrease significantly enough then we stop. Early stopping may underfit by stopping too early. The current split may be of little benefit, but having made it, subsequent splits more significantly reduce the error.

Early stopping and pruning can be used together, separately, or not at all. Post pruning decision trees is more mathematically rigorous, finding a tree at least as good as early stopping. Early stopping is a quick fix heuristic.

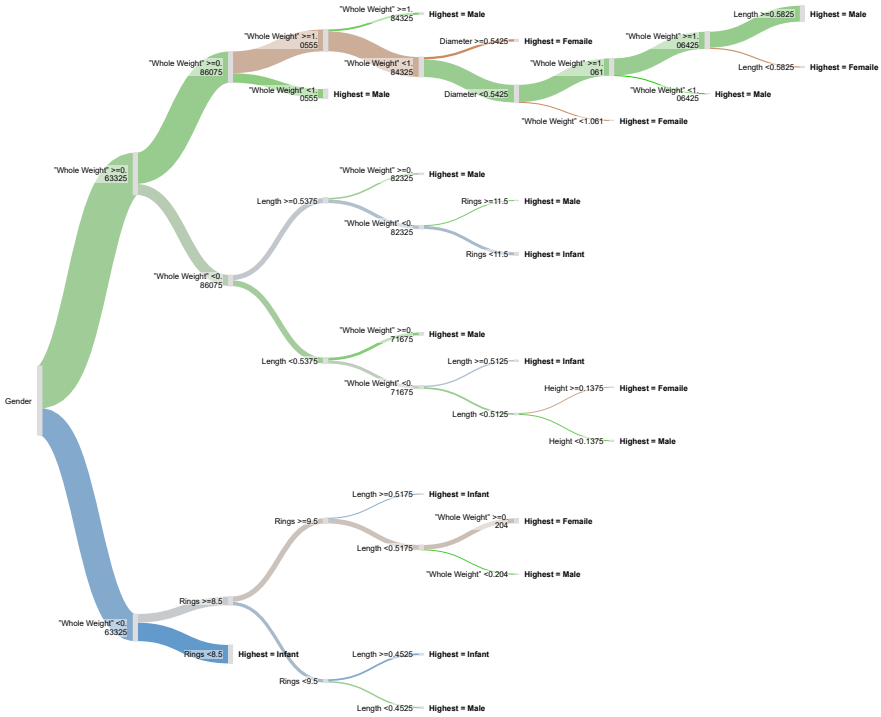
If used together with pruning, early stopping may save time. After all, why build a tree only to prune it back again?

Create your own Box Plot

Another portion of abalone?

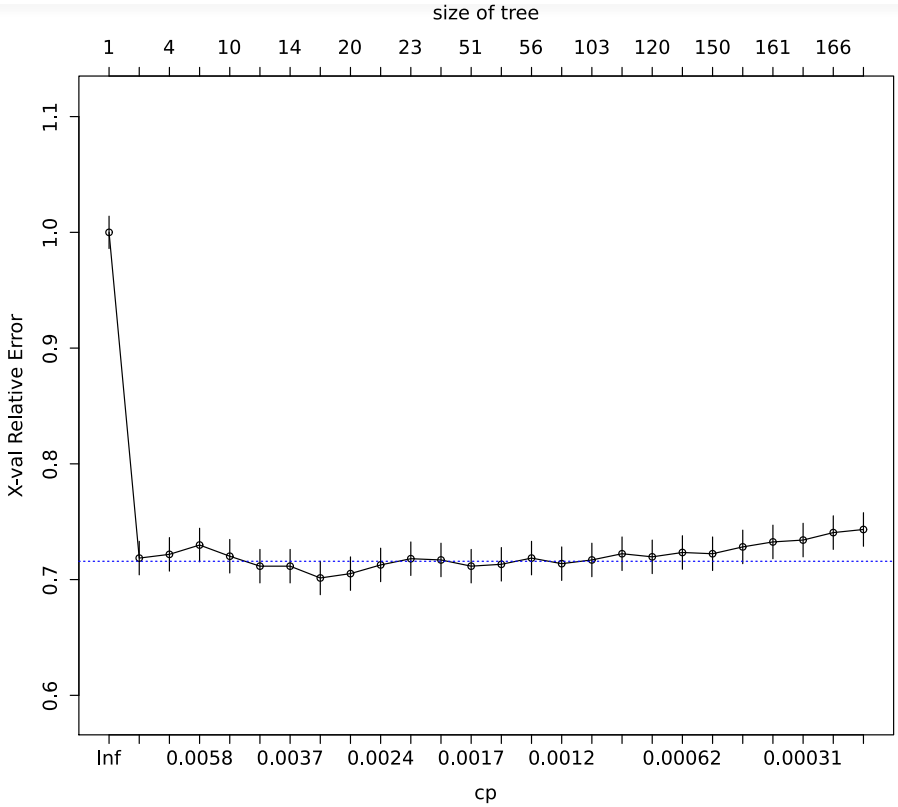
I have used a data set containing information about abalone in another blog post, which looks at how abalone gender could be predicted by physical characteristics. Let's go back for a second helping, this time with CART.

I will set *Gender* as the outcome variable, and *Length, Diameter, Height, Whole Weight* and *Rings* (a proxy for age) as predictors. The output with minimum error pruning and no early stopping is as follows,



You can use your mouse wheel to zoom in and see the detail, which is necessary because this is not a small tree. The labels for each leaf show which of the abalone sex categories has the highest representation within that part of the sample. Information about the sample in each leaf will pop up when you hover your mouse. The label for each split indicates which variable determined the split.

We can use cross validation to see how the error in the tree changes with the size of the tree. This is shown in the chart below. We see that the minimum error tree has around 18 leaves. Beyond this size the cross validation error gradually increases, which can be a sign of overfitting.



Now let's try setting pruning to the smallest tree and turning early stopping on. This gives something much more simple - a tree with a single pair of leaves. However, we know from the cross validation chart above that this does not have the best error. In fact, the cross validation error temporarily increases after 2 leaves, which causes the early stopping in to kick in as the model assumes it can no longer improve. This highlights a risk of early stopping, as there may be more accurate decision trees to be found by continuing.

Number of leaves

The following table summarizes the tree size for all 6 combinations of pruning and stopping. In this case early stopping produces such a simple simple tree that pruning has no effect. Without early stopping, smallest tree pruning cuts back the minimum error tree.

	No early stopping	Early stopping
Minimum error pruning	18	2
Smallest tree pruning	11	2
No pruning	169	2

Accuracy

The trees previously referred to are trained on only 70% of the data. I have held back a random 30% sample for testing the accuracy, allowing an unbiased accuracy to be calculated. The results for the 6 combinations are:

	No early stopping	Early stopping
Minimum error pruning	53.07%	51.72%
Smallest tree pruning	52.11%	51.72%
No pruning	51.48%	51.72%

For this data the lowest accuracy results from early stopping (underfitting), or neither pruning nor stopping (overfitting). The "sweet spot" in the middle is without early stopping, and pruning to the minimum cross validation error.

Create your own decision tree

Summary

Pruning and early stopping help with the trade-offs involved when using CART. The learning above boils down to a few guidelines:

- For best accuracy, minimum error pruning without early stopping is usually a good choice.
- For a compromise between accuracy and an interpretable tree, try smallest tree pruning without early stopping.
- To produce an even smaller tree or reduce the running time while allowing accuracy to decrease, you can turn on early stopping.

Use the template above to create and prune your own decision tree.

The analysis above is produced using R in Displayr. It uses the `flipTrees` package, which uses the `rpart` package.

To start creating your own decision trees using Displayr, get started below.

Sign up for free