# ADS – H2O

## Datasets

190 datasets: binary and multiclass

## Metafeatures

77 statistical metrics are exported from each dataset

## Algorithm for Classification

Our first approach uses Random Forest h2o algorithm,

Random forest is an ensemble algorithm, meaning more than one model is made, and their results used together, the aim being to cope better with unseen situations (i.e.,to avoid overfitting). If you train a decision tree on a fairly complex data set (and don’t take precautions against overfitting), you will find a very deep tree full of fragile rules. The idea behind random forest is to instead have lots of trees. Then, when you use it to predict on new data, you give the new data to each of those trees and ask each for their prediction. If it’s a classification you choose the most popular answer, and if it’s a regression you

take the mean of each tree’s answer. That is the “forest” half. The other half, the “random,” says that when training you don’t give each tree all the training data; you randomly hold back some rows, or hold

back some columns. This makes each individual tree a bit dumber than if it had seen all the data. But when their results are averaged together the whole is more intelligent than any one part.

Parameters:

The most important parameters are:

* Ntress: how many trees in your forest
* Max\_depth: hoe deep a tree allowed to grow, how complex each tree is allowed to be.
* Sample\_rate: the default is 0.632, which means that each tree is trained of the 63.2% of the training data.
* Etc.

## Hyperparameter Search: Method Grid Search

All the H2O machine learning algorithms have parameters: knobs, which you can tweak, that will often affect the performance of the model you build. But the interactions between the parameters can be complex. The labor-intensive way is to try a model, evaluate it, then fiddle with one of the parameters, and repeat. If your intuition is good this may be the most efficient way.

Grids are the solution to this dilemma, and the H2O implementation currently comes

in two forms:

• Comprehensive ("Cartesian")

• Random ("RandomDiscrete")

Summary

Random forests models are generally quick to build, and give effective results on most problems. There are not too many tuning knobs and, looking back over this chapter, the most effective technique was to increase ntrees, in combination with early stopping. Increasing max\_depth was also effective. Random forest is not the only way that the basic decision tree idea has been improved, though, and the next chapter will look at an alternative approach.