1 Resampling Methods

2 Resampling Methods

The resampling methods used by **caret** are:

- bootstrapping,
- k-fold crossvalidation,
- leave-one-out cross-validation,
- leave-group-out cross-validation (i.e., repeated splits without replacement).

2.1 Avoiding Over-Fitting

A major issue that arises when applying regression or classification trees to "real" data with much random error noise concerns the decision when to stop splitting. For example, if you had a data set with 10 cases, and performed 9 splits (determined 9 if-then conditions), you could perfectly predict every single case. In general, if you only split a sufficient number of times, eventually you will be able to "predict" ("reproduce" would be the more appropriate term here) your original data (from which you determined the splits). Of course, it is far from clear whether such complex results (with many splits) will replicate in a sample of new observations; most likely they will not.

This general issue is also discussed in the literature on tree classification and regression methods, as well as neural networks, under the topic of **overlearning** or **overfitting**. If not stopped, the tree algorithm will ultimately "extract" all information from the data, including information that is not and cannot be predicted in the population with the current set of predictors, i.e., random or noise variation.

The general approach to addressing this issue is first to stop generating new split nodes when subsequent splits only result in very little overall improvement of the prediction. For example, if you can predict 90% of all cases correctly from 10 splits, and 90.1% of all cases from 11 splits, then it obviously makes little sense to add that 11th split to the tree. There are many such criteria for automatically stopping the splitting (tree-building) process.

2.1.1 Pruning

Once the tree building algorithm has stopped, it is always useful to further evaluate the quality of the prediction of the current tree in samples of observations that did not participate in the original computations. These methods are used to "prune back" the tree, i.e., to eventually (and ideally) select a simpler tree than the one obtained when the tree building algorithm stopped, but one that is equally as accurate for predicting or classifying "new" observations.

2.1.2 Crossvalidation

One approach is to apply the tree computed from one set of observations (learning sample) to another completely independent set of observations (testing sample). If most

or all of the splits determined by the analysis of the learning sample are essentially based on "random noise," then the prediction for the testing sample will be very poor. Hence one can infer that the selected tree is not very good (useful), and not of the "right size."

2.1.3 V-fold crossvalidation

Continuing further along this line of reasoning (described in the context of crossvalidation above), why not repeat the analysis many times over with different randomly drawn samples from the data, for every tree size starting at the root of the tree, and applying it to the prediction of observations from randomly selected testing samples. Then use (interpret, or accept as your final result) the tree that shows the best average accuracy for cross-validated predicted classifications or predicted values.

In most cases, this tree will not be the one with the most terminal nodes, i.e., the most complex tree. This method for pruning a tree, and for selecting a smaller tree from a sequence of trees, can be very powerful, and is particularly useful for smaller data sets. It is an essential step for generating useful (for prediction) tree models, and because it can be computationally difficult to do, this method is often not found in tree classification or regression software.