# Study Note: Model Validation and Holdout Data

04/02/2018

The importance of testing models on out-of-sample data should not be underestimated. However, there are different types of testing that accomplish different things. For example, it is common to divide one's data into three parts: (1) a **training** set, used for building models; (2) a **test** set, used frequently during the model build process to test those models; and (3) a **holdout** set, saved until the end of the modeling process, to provide an objective metric of goodness of fit that can be reported to management. (The term "validation set" is sometimes used, but since it is sometimes used for (2) and sometimes for (3), we have avoided this term.) This metric could be a Gini coefficient, the area under the ROC curve, the squared error (or other cost function), a gains chart, or a lift curve, in each case applied to the holdout data the model has not yet seen. In order to ensure that the holdout test is objective, one should separate the holdout data as early in the process as practicable[1], and store it in a separate file not accessible during the modeling process.

There is no hard-and-fast rule for the sizes of these sets across all types of models, but for many types of models, it is usual to assign one-third of the data at random to each set. It may be tempting to make the training set larger and the test and holdout datasets smaller, but (a) building on a smaller dataset is a good way to avoid a natural tendency toward overfit models with too many parameters and (b) if your test and holdout datasets are too small, one will be tempted not to trust the out-of-sample tests, which rather defeats their purpose.

The reason that the test set will not serve the purpose of the objective test is that, since one has repeatedly compared to it, **one has in some sense fit the model to the test data** as well as to the training data. In fact, one can make a virtue of this and swap the roles of the training and test data during the course of the modeling process, as one zeroes in on the best model.

We are often asked why one needs holdout data when one can get an out-of-sample test using cross-validation, where a dataset is divided into $K$ folds and the goodness of fit calculated by using the model fit to all folds but the $i^{th}$ to make predictions for the

[1] Typically one does some level of data profiling before separating the holdout data in order to: (a) match control totals to ensure the dataset is complete and correct; and (b) ensure that all possible values are known for each categorical field.

observations in the $i^{th}$ fold. This can indeed work if the model building process is completely automated. However, when human beings are building the model, **the choice of model** (including type of model, which variables were candidates to final feature selection, which derived features were created) **has typically been impacted by all the data the modeler has worked with**…the whole dataset. Even though one may be fitting parameters and even performing final feature selection separately for each collection of *K-1* folds, those models will have been influenced by all the data. This does not mean that cross-validation is useless for human-built models. It merely means that the proper use of cross-validation is as a substitute for the training/test split, not as a substitute for having true holdout data. (In fact, the suggestion above to swap training and test data during the modeling process is a form of two-fold cross-validation that is very tractable for human modelers.) Also, note that nothing is lost in terms of the final model by using the holdout approach. It is familiar in the cross-validation approach that the union of the cross-validation models provides the goodness of fit metrics whilst the model to be used is fit on all the data at once. **The same is true in the holdout approach. After testing on holdout data to obtain the objective quality of the model, one can (and usually does) refit on the complete data to obtain the final model. One can even change the model if adding the holdout data makes it clear that one should. What one cannot do is claim a better goodness of fit than one has already measured.**

In order to ensure independence of the training, test, and validation data, **it is good practice to ensure the correlated observations (i.e., ones that are still correlated even after conditioning on the model variables) are all put into the same group**. For example, if the unit of observation is one policy for one year, the series of observations that correspond to a single policyholder over that period of time would be put into the same group. Or if the model is a severity model for weather claims, claims from the same storm might be put into the same group. If time is the main driver of correlation then one might use a contiguous block of time for each group, known as out-of-time validation. For many insurance problems, however, time is not the main dimension, and splitting on the time variable may actually be undesirable because one may want to incorporate time explicitly in the model or to test consistency of effects across time within the training data as one of the criteria for including a variable. This latter is actually straightforward to do: Look at residuals across the levels of the variable in question for the earlier half (temporally) of the data and for the latter half of the data, and compare.

Finally, one practical aspect of splitting data for testing and validation is that datasets evolve. Perhaps, for example, it turns out that a small set of observations were erroneously included, or the business identifies certain observations as not relevant to going-forward strategy and asks that they be omitted. If you have used a random

number generator and appended a string of random numbers to a list of policy numbers in order to split the data into thirds, now you not only need to have stored the random number seed and have access to the exact same random number generator to reproduce the random split, and to have sorted your data the precise way you had done previously, you also need to match at precisely the right midway point in your data filtering process. Thus, it can be much simpler to use a **hash** of the relevant unit information (e.g., policy number and policyholder name) rather than a random number generator in order to assign an observation to a group. However, the selection of hash function is critical, since one needs to avoid assigning similar hashes to similar initial values, and most hash functions will fail this test. However, the hash function MD5 is considered suitable for this purpose.[2]

[2] The interested reader will find additional information in section 4.1 of Kohavi, Henne, and Sommerfield, "Practical Guide to Controlled Experiments on the Web", KDD2007, currently available at https://ai.stanford.edu/~ronnyk/2007GuideControlledExperiments.pdf.