H2O reproducibility — Reproducibility in H2O$^+$

$^+$These features are part of StataNow.

Description Also see

Description

Reproducibility is an important consideration in all scientific research, data analyses, and machine learning experiments. The goal is ensure that repeating the same analysis under the same conditions will yield identical results. In H2O, reproducibility can be affected by randomness in data splitting, model training, and the design of the machine learning method.

Below, we provide a list of guidelines to help you ensure that your analysis and results are reproducible. For more details, see H2O's reproducibility page.

1. **Control data splitting:** If you split the dataset into multiple datasets, such as training and testing sets, by using the _h2oframe split command, control the randomness of the splitting by setting the random-number seed with the rseed() option. For example, you might type

   ```
   . _h2oframe split mydata, into(train test) split(0.8 0.2) rseed(19)
   ```

2. **Set a seed when fitting a model:** Gradient boosting machine (GBM) and random forest methods use random-number generation for various operations throughout estimation and grid search. For example, the observation sampling rate and column sampling rate set by the samprate() and colsamprate() options in the commands for GBM use a seed for sampling. To ensure reproducibility, set a seed via the h2orseed() option for both the model and the grid search. For example, you might type

   ```
   . h2oml gbregress y x1 x2, h2orseed(19) ntrees(10(4)20)
   > tune(grid(random, h2orseed(20)))
   ```

3. **Make sure hyperparameters are the same in every execution:** For reproducibility, the hyperparameters of the model, such as those set by the maxdepth(), samprate(), minobsleaf(), and other hyperparameter options, should be identical in each execution of the estimation command.

4. **Be careful with early stopping:** Early stopping, specified by the stop() option in GBM and random forest commands, stops the training process early when the model performance does not improve. Even though early stopping may prevent overfitting and significantly improve execution time, it is a potential source of nonreproducibility. By default, during training H2O determines an interval T, and the model performance is scored only after T trees are added to the model. In each execution of the estimation command, this default interval T can vary, which affects the scoring of the model performance, and the training may stop at different times. To ensure that the scoring of the model is consistent throughout multiple executions, specify the scoreevery() option with early stopping. For example, you might type

   ```
   . h2oml gbregress y x1 x2, h2orseed(19) ntrees(100) stop(3) scoreevery(1)
   ```

5. **Control parallelism:** The number of machine cores, the specified number of threads during cluster initialization, and the parallelism level determine how a dataset is partitioned in memory (referred to as "chunks" by H2O) and affect the estimation of various methods, such as GBM. While H2O leverages parallelism to improve training time, this can introduce some randomness when running on multiple threads and cores.

You can limit parallelism during cluster initialization by specifying the desired number of threads using the nthread() option in the h2o.init command. For example, you can type

```
. h2o init, nthread(1)
```

However, even though nthreads() is closely related to the number of cores, in H2O this does not determine how it partitions the dataset into chunks, as this depends on the number of cores available on the machine. If the number of chunks varies, the order of operations executed by H2O will also differ. As a result, certain numeric operations may produce slightly different outcomes depending on the order of operations. This can lead to small variations in metrics sensitive to ordering, such as AUC, AUCPR, etc, when the same model with the same parameters is run in a machine with different number of cores.

The reproducibility issues described above also apply when you choose to enable parallel model building during grid search to reduce computational time. For example,

```
. h2oml gbregress y x1 x2, h2orseed(19) ntrees(100(50)200) tune(parallel(0))
```

6. **Use the same version of H2O:** A different version of H2O may contain slight differences in implementation of the method, which can affect the reproducibility. To avoid discrepancies, ensure that the same version of H2O is used each time the command is executed. The version of H2O in Stata can be checked by using the h2o query command. In the output below, the H2O version is 3.46.0.6. For details on how to download and set up H2O, see [H2OML] **H2O setup**.

```
. h2o query
Cluster is running at http://127.0.0.1:54321.
──────────────────────────────────────────────────────────────────────────────

H2O cluster uptime:         1 hour 0 mins
H2O cluster timezone:       America/Chicago
H2O data parsing timezone: UTC
H2O cluster version:        3.46.0.6
H2O cluster version age:    3 months
H2O cluster total nodes:    1
H2O cluster free memory:    6.892 Gb
H2O cluster total cores:    28
H2O cluster allowed cores: 28
H2O cluster status:         locked, healthy
H2O connection url:         http://127.0.0.1:54321
──────────────────────────────────────────────────────────────────────────────
```

Also see

[H2OML] **h2oml** — Introduction to commands for Stata integration with H2O machine learning+

[H2OML] ***h2oml gbm*** — Gradient boosting machine for regression and classification+

[H2OML] ***h2oml rf*** — Random forest for regression and classification+

For suggested citations, see the FAQ on citing Stata documentation.