h2omlgraph permimp — Produce permutation importance plot

Description

`h2omlgraph permimp` plots the permutation variable importance after `h2oml` *gbm* and `h2oml` *rf*. Permutation variable importance for ensemble decision tree methods, such as random forest and gradient boosting machine, measures the increase or decrease in the performance metric of the model after we permute the values of a predictor, breaking the relationship between the predictor and the response.

Quick start

Plot the permutation variable importance

```
h2omlgraph permimp
```

As above, but plot the top 5 important predictors and set the H2O random seed

```
h2omlgraph permimp, h2orseed(19) top(5)
```

Plot the permutation importance for the predictors x1 and x2

```
h2omlgraph permimp x1 x2
```

As above, but set the number of permutations equal to 5

```
h2omlgraph permimp x1 x2, reps(5)
```

Menu

Statistics > H2O machine learning

Syntax

Multiple permutation repetitions

```
h2omlgraph permimp [predictors] [ , options options1 ]
```

One permutation repetition

```
h2omlgraph permimp [predictors], reps(1) [options options2 ]
```

options	Description
Main	
`reps(#)`	specify number of permutations; default is `reps(10)`
`top(#)`	plot the top # important predictors; default is `top(10)`
`metric(`*metric*`)`	specify the performance metric to estimate the permutation importance
`samples(#)`	specify the number of observations to be randomly sampled to compute the permutation importance
`h2orseed(#)`	set H2O random-number seed
`table`	display results as a table
`savedata(`*filename*`[ , replace ])`	save plot data to *filename*
`train`	specify that the permutation importance be reported using training results
`valid`	specify that the permutation importance be reported using validation results
`test`	specify that the permutation importance be computed using testing frame
`test(`*framename*`)`	specify that the permutation importance be computed using data in testing frame *framename*
`frame(`*framename*`)`	specify that the permutation importance be computed using data in H2O frame *framename*
`framelabel(`*string*`)`	label frame as *string* in the output

`train`, `valid`, `test`, `test()`, `frame()`, and `framelabel()` do not appear in the dialog box.

options$_1$	Description
Plot options	
`overopts(`*over_subopts*`)`	specify `over()` suboptions other than `sort()` and `descending` documented in [G-2] **graph box**
hbox_options	any options other than `by()` and `over()` documented in [G-2] **graph box**

options$_2$	Description
Main	
proportion	plot the proportional contribution of the importance of each predictor; the default
relative	plot relative influence of each predictor
scaled	plot scaled importance of each predictor
Plot options	
bar	plot variable importance as a bar plot; the default
baropts(*bar_opts*)	affect rendition of the bar plot
dot	plot variable importance as a dot plot
dotopts(*dot_opts*)	affect rendition of the dot plot
valuelabel	display variable importance values
valuelabelopts(*label_opts*)	affect the labeling of importance values
twoway_options	any options other than by() documented in [G-3] ***twoway_options***

Options

Main

reps(#) specifies the number of permutation repetitions. The default is reps(10).

top(#) plots the top # permutation important predictors. The default is top(10). top() is not allowed if *predictors* are specified.

metric(*metric*) specifies the performance metric to compute permutation importance. *metric* may be one of the following:

After regression: mse, rmse, or mae.

After binary classification: logloss, meanclasserror, auc, aucpr, mse, or rmse.

After multiclass classification: logloss, meanclasserror, mse, or rmse.

mse is the default metric for regression. logloss is the default metric for binary and multiclass classification. See [H2OML] ***metric_option*** for the full description.

samples(#) specifies the number of observations to be randomly sampled to estimate the permutation importance; if the number of observations of the corresponding frame is less than the specified samples(), the minimum between the number of observations and 10,000 is used. This option is useful for datasets with many observations to reduce computational time.

h2orseed(#) sets the H2O random-number seed for H2O reproducibility. This option is not equivalent to the rseed() option available with other commands or the set seed command. For reproducibility in H2O, see [H2OML] **H2O reproducibility** and H2O's reproducibility page.

table displays results as a table. The table is suppressed by default.

savedata(*filename*[, replace]) saves the plot data to a Stata data file (.dta file). replace specifies that *filename* be overwritten if it exists.

The following options are specific to the case of multiple permutation repetitions:

Plot options

`overopts(`*over_subopts*`)` specifies `over()` suboptions other than `sort()` and `descending` documented in [G-2] **graph box**.

hbox_options specify any options other than `by()` and `over()` documented in [G-2] **graph box**.

The following options are specific to the case of one permutation repetition:

Main

`proportion`, `relative`, and `scaled` specify the type of the variable importance contribution to be plotted.

`proportion` plots the proportional contribution of the importance of each predictor. It is calculated by dividing the importance of each predictor by the total sum of the importance of all predictors. `proportion` is the default.

`relative` plots the importance, which is the relative influence of each predictor.

`scaled` plots the scaled importance. It is calculated by dividing the importance of each predictor by the largest importance score of the predictors.

Only one of `proportion`, `relative`, or `scaled` is allowed.

Plot options

`bar` plots the variable importance as a bar plot. This is the default. `bar` is not allowed with `dot`.

`baropts(`*bar_opts*`)` affects rendition of the bar plot. *bar_opts* are any of the options documented in [G-2] **graph twoway bar**, excluding `horizontal`, `vertical`, and `base()`.

`dot` plots the variable importance as a dot plot. `dot` is not allowed with `bar`.

`dotopts(`*dot_opts*`)` affects the rendition of the dot plot. *dot_opts* are any of the options documented in [G-2] **graph twoway dot**, excluding `horizontal`, `vertical`, and `base()`.

`valuelabel` displays the values of the variable importance on the graph.

`valuelabelopts(`*label_opts*`)` affects the labeling of variable importance values. *label_opts* include any of the options documented in [G-3] ***marker_label_options***, excluding `mlabel()`.

twoway_options are any of the options documented in [G-3] ***twoway_options***, excluding `by()`, `horizontal`, and `vertical`. These include options for titling the graph (see [G-3] ***title_options***) and options for saving the graph to disk (see [G-3] ***saving_option***).

The following options are available with `h2omlgraph permimp` but are not shown in the dialog box:

`train`, `valid`, `test`, `test()`, and `frame()` specify the H2O frame for which permutation importance is reported. Only one of `train`, `valid`, `test`, `test()`, or `frame()` is allowed.

`train` specifies that permutation importance be reported using training results. This is the default when validation is not performed during estimation and when a postestimation frame has not been set with `h2omlpostestframe`.

`valid` specifies that permutation importance be reported using validation results. This is the default when validation is performed during estimation and when a postestimation frame has not been set with `h2omlpostestframe`. `valid` may be specified only when the `validframe()` option is specified with `h2oml` *gbm* or `h2oml` *rf*.

`test` specifies that permutation importance be computed on the testing frame specified with `h2omlpostestframe`. This is the default when a testing frame is specified with `h2omlpostestframe`. `test` may be specified only after a testing frame is set by using `h2omlpostestframe`. `test` is necessary only when a subsequent `h2omlpostestframe` command is used to set a default postestimation frame other than the testing frame.

`test(`*framename*`)` specifies that permutation importance be computed using data in testing frame *framename* and is rarely used. This option is most useful when running a single postestimation command on the named frame. If multiple postestimation commands are to be run on the same test frame, it is more computationally efficient and convenient to specify the testing frame by using `h2omlpostestframe` instead of specifying `test(`*framename*`)` with individual postestimation commands.

`frame(`*framename*`)` specifies that permutation importance be computed using the data in H2O frame *framename*.

`framelabel(`*string*`)` specifies the label to be used for the frame in the output.

Remarks and examples

We assume you have read *Interpretation and explanation* in [H2OML] **Intro**.

To understand and explain machine learning predictions, we usually ask a question: Which predictors drive the predictions?

Explainability methods, particularly variable importance, help us identify the most influential predictors. In tree-based models, such as random forests, a common approach relies on impurity-based importance, which quantifies how much each predictor contributes to splitting the training data and reducing uncertainty, as measured by entropy or MSE. For more details, see [H2OML] **h2omlgraph varimp**.

While useful, this method has some drawbacks. It reflects how frequently a predictor is used to split the data but not necessarily how essential that predictor is for improving prediction accuracy. As a result, impurity-based measures can be biased toward predictors with more categories. We explore this further in example 1.

Permutation variable importance, first introduced by Breiman (2001) for random forests and later extended to a model-agnostic form by Fisher, Rudin, and Dominici (2019), is designed to answer a more fundamental question: What happens to a prediction when we break the relationship between a predictor and the response?

Intuitively, a predictor is considered important if scrambling its values affects the model's ability to make accurate predictions. Permutation variable importance permutes one predictor at a time and measures the resulting change in model performance.

We summarize the main steps below:

1. For $j = 1, 2, \ldots, B$, where B is the number of permutation repititions, do the following:
   i. Use the Fisher–Yates algorithm (Knuth 1998) to randomly shuffle predictor X_i, where $1 \leq i \leq p$ and p is the number of predictors.
   ii. Use a machine learning model to generate predictions on the permuted data.
   iii. Compute the performance metric M_j^{perm}.
   iv. Calculate the absolute difference between the performance metric on the original data and the permuted data: $|M^{\text{orig}} - M_j^{\text{perm}}|$.

The number of permutations B can be specified by using the `reps()` option; the default is $B = 10$. You may specify a single predictor or a list of predictors for which to compute variable importance. By default, permutation importance is computed for all predictors. The performance metric can be specified by using the `metric()` option.

In contrast to `h2omlgraph varimp`, which relies on training data, permutation importance can be evaluated on testing data. This helps avoid overfitting and improves the generalizability of the model explanations.

▷ Example 1: Comparing variable importance methods

In this example, we compare the impurity-based variable importance (`h2omlgraph varimp`) with the permutation importance on the simulated `effect` dataset, described in example 13 of [H2OML] **h2oml**. We aim to examine whether permutation importance is more reliable when the dataset contains a categorical predictor with many categories.

Recall that in this dataset the variables with the prefix `important` identify those variables that affect the response, whereas variables with the prefix `noise` represent random noise. In addition, we set Stata's random seed and generate a new variable, `noise_cat`, that is unrelated to the response but has many categories.

```
. use https://www.stata-press.com/data/r19/effect
(Simulated data with many nuisance predictors)
. set seed 19
. generate noise_cat = runiformint(1, 100)
```

Next we initialize an H2O cluster and import the dataset as an H2O frame `sim`. We also split the `sim` frame into training and validation with 60% of observations in the training sample. We also specify the random-number seed for reproducibility and make the `train` frame be the current H2O frame for estimation.

```
. h2o init
. _h2oframe put, into(sim)
. _h2oframe split sim, into(train test) split(0.6 0.4) rseed(19)
. _h2oframe change train
```

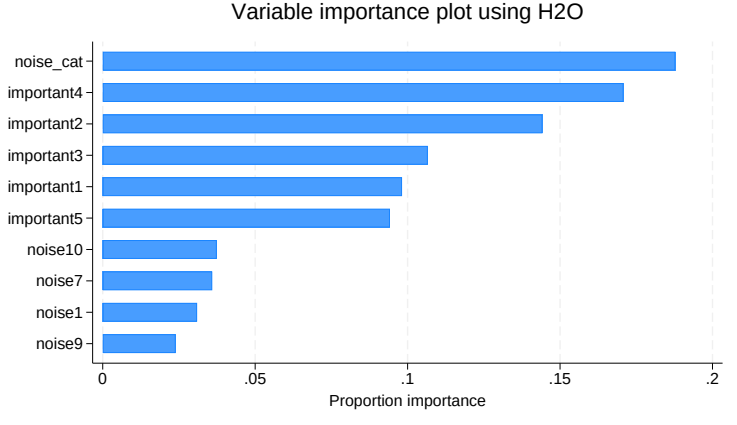
For illustration purposes, we run gradient boosting binary classification using the default hyperparameters. In practice, before attempting to explain a model, we recommend tuning hyperparameters to obtain a high-performing model.

```
. h2oml gbbinclass y important1-important5 noise_cat noise1-noise10, h2orseed(19)
Progress (%): 0 100
Gradient boosting binary classification using H2O
Response: y
Loss:     Bernoulli
Frame:                                  Number of observations:
  Training: train                                 Training =    607
Model parameters
Number of trees      =  50              Learning rate       =     .1
             actual =  50               Learning rate decay =      1
Tree depth:                             Pred. sampling rate =      1
           Input max =   5              Sampling rate       =      1
                 min =   5              No. of bins cat.    =  1,024
                 avg = 5.0              No. of bins root    =  1,024
                 max =   5              No. of bins cont.   =     20
Min. obs. leaf split =  10              Min. split thresh.  = .00001
Metric summary
```

Metric	Training
Log loss	.352606
Mean class error	.1068004
AUC	.9557621
AUCPR	.9573292
Gini coefficient	.9115242
MSE	.1034411
RMSE	.3216226

First, we plot the impurity-based variable importance by using the `h2omlgraph varimp` command.

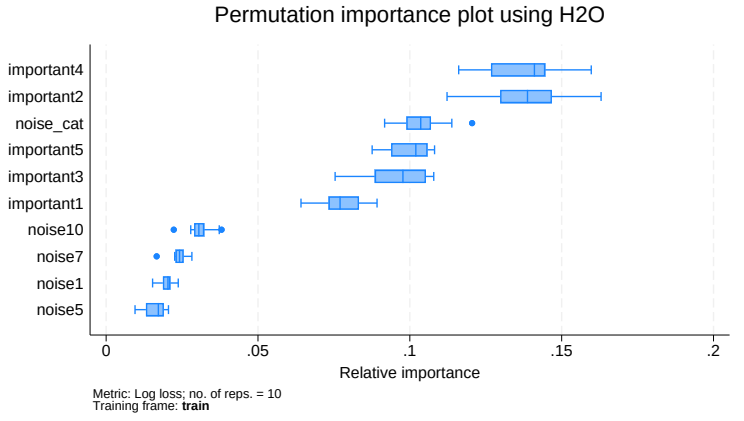
```
. h2omlgraph varimp
```



This method erroneously identifies the random noise `noise_cat` as the most important predictor.

Next, we plot the permutation importance for the training dataset, which is the default frame in this case. We also specify the random seed for reproducibility.

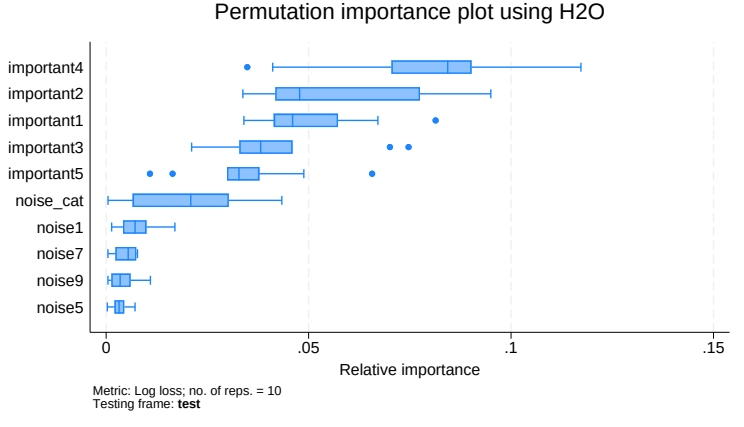
```
. h2omlgraph permimp, h2orseed(19)
```



The box plot for each predictor summarizes the importance over 10 permutations. The center line in each box represents the median relative importance of the corresponding predictor. Predictors are ordered in descending order based on their median importance.

We can see that for the training frame, permutation importance, similar to `h2omlgraph varimp`, misidentified the predictor `noise_cat` as important. However, as we mentioned earlier, with `h2omlgraph permimp` we can compute the permutation importance on the testing data, which improves the reliability of the explanation. This is illustrated in the plot below, where we use the `test()` option to specify the testing frame.

```
. h2omlgraph permimp, h2orseed(19) test(test)
```



On the testing frame, the importance of the noise predictor `noise_cat` is almost 0.

References

Breiman, L. 2001. Random forests. *Machine Learning* 45: 5–32. https://doi.org/10.1023/A:1010933404324.

Fisher, A., C. Rudin, and F. Dominici. 2019. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research* 20: 1–81.

Knuth, D. E. 1998. *Seminumerical Algorithms*. Vol. 2 of *The Art of Computer Programming*, 3rd ed. Reading, MA: Addison–Wesley.

Also see

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

[H2OML] **h2omlgraph varimp** — Produce variable importance plot