

⁺This command includes features that are part of [StataNow](#).

Description	Quick start	Menu	Syntax
Options	Remarks and examples	Stored results	Also see

Description

`h2oml gbbinclass` implements gradient boosting classification for binary responses. You can validate your model by using validation data or cross-validation, and you can tune hyperparameters and stop early to improve model performance on new data. This command provides only measures of performance. See [\[H2OML\] h2oml postestimation](#) for commands to compute and explain predictions, examine variable importance, and perform other postestimation analyses.

For an introduction to decision trees and the gradient boosting machine (GBM) method, see [\[H2OML\] Intro](#).

Quick start

Before running the `h2oml gbbinclass` command, an H2O cluster must be initialized and data must be imported to an H2O frame; see [\[H2OML\] H2O setup](#) and *Prepare your data for H2O machine learning in Stata* in [\[H2OML\] h2oml](#).

Perform gradient boosting binary classification of binary response `y1` on predictors `x1` through `x100`

```
h2oml gbbinclass y1 x1-x100
```

As above, but also report measures of fit for the validation frame named `valid`, and set an H2O random-number seed for reproducibility

```
h2oml gbbinclass y1 x1-x100, validframe(valid) h2orseed(123)
```

As above, but instead of a validation frame, use 3-fold cross-validation

```
h2oml gbbinclass y1 x1-x100, cv(3) h2orseed(123)
```

As above, but set the number of trees to 30, the maximum tree depth to 10, the learning rate to 0.01, and the predictor sampling rate to 0.6

```
h2oml gbbinclass y1 x1-x100, cv(3) h2orseed(123) ntrees(30) ///
maxdepth(10) lrate(0.01) pedsamprate(0.6)
```

As above, but use the default exhaustive grid search to select the optimal number of trees and the maximum tree depth that minimize the log-loss metric

```
h2oml gbbinclass y1 x1-x100, cv(3) h2orseed(123) lrate(0.01) ///
pedsamprate(0.6) ntrees(10(5)100) maxdepth(3(1)10) ///
tune(metric(logloss))
```

As above, but use a random grid search, set an H2O random-number seed for this search, and limit the maximum search time to 200 seconds

```
h2oml gbbinclass y1 x1-x100, cv(3) h2orseed(123) lrate(0.01) ///
pedsamprate(0.6) ntrees(10(5)100) maxdepth(3(1)10) ///
tune(metric(logloss) grid(random, h2orseed(456)) maxtime(200))
```

As above, but specify a learning-rate decay of 0.9, and tune the number of bins for the categorical and continuous predictors

```
h2oml gbbinclass y1 x1-x100, cv(3) h2orseed(123) lrate(0.01)    ///  
  lratedecay(0.9) predsamprate(0.6) ntrees(10(5)100)         ///  
  maxdepth(3(1)10) binscont(15(5)50) binscat(500(5)1100)    ///  
  tune(metric(logloss) grid(random, h2orseed(456)) maxtime(200))
```

Menu

Statistics > H2O machine learning

Syntax

```
h2oml gbbinclass response_bin predictors [ , options ]
```

response_bin and *predictors* correspond to column names of the current H2O frame.

<i>options</i>	Description
Model	
<code>validframe(<i>framename</i>)</code>	specify the name of the H2O frame containing the validation dataset that will be used to evaluate the performance of the model
<code>cv[(# [, <i>cvmethod</i>])]</code>	specify the number of folds and method for cross-validation
<code>cv(<i>colname</i>)</code>	specify the name of the variable (H2O column) for cross-validation that identifies the fold to which each observation is assigned
<code>balanceclasses</code>	balance the distribution of classes (categories of the response variable) by oversampling the minority class
<code>h2orseed(#)</code>	set H2O random-number seed for GBM
<code>encode(<i>encode_type</i>)</code>	specify H2O encoding type for categorical predictors; default is <code>encode(enum)</code>
<code>stop[(# [, <i>stop_opts</i>])]</code>	specify the number of training iterations and other criteria for stopping GBM training if the stopping metric does not improve
<code>maxtime(#)</code>	specify the maximum run time in seconds for GBM; by default, no time restriction is imposed
<code>scoreevery(#)</code>	specify that metrics be scored after every # trees during training
<code>monotone(<i>predictors</i> [, <i>mon_opts</i>])</code>	specify monotonicity constraints on the relationship between the response and the specified predictors
Hyperparameter	
<code>ntrees(# <i>numlist</i>)</code>	specify the number of trees to build the GBM model; default is <code>ntrees(50)</code>
<code>lrate(# <i>numlist</i>)</code>	specify the learning rate of each tree; default is <code>lrate(0.1)</code>
<code>lratedecay(# <i>numlist</i>)</code>	specify the rate by which the learning rate specified in <code>lrate()</code> is decaying after adding each tree to the GBM; default is <code>lratedecay(1)</code>
<code>maxdepth(# <i>numlist</i>)</code>	specify the maximum depth of each tree; default is <code>maxdepth(5)</code>
<code>minobsleaf(# <i>numlist</i>)</code>	specify the minimum number of observations per child for splitting a leaf node; default is <code>minobsleaf(10)</code>
<code>predsamprate(# <i>numlist</i>)</code>	specify the sampling rate for randomly selecting a fraction of predictors to build a tree; default is <code>predsamprate(1)</code>
<code>samprate(# <i>numlist</i>)</code>	specify the sampling rate for randomly selecting a fraction of observations to build a tree; default is <code>samprate(1)</code>
<code>minsplitthreshold(# <i>numlist</i>)</code>	specify the threshold for the minimum relative improvement needed for a node split; default is <code>minsplitthreshold(1e-05)</code>
<code>binscat(# <i>numlist</i>)</code>	specify the number of bins to build the histogram for node splits for categorical predictors (enum columns in H2O); default is <code>binscat(1024)</code>

4 h2oml gbbinclass — Gradient boosting binary classification⁺

<code>binsroot(# <i>numlist</i>)</code>	specify the number of bins to build the histogram for root node splits for continuous predictors (<code>real</code> and <code>int</code> columns in H2O); default is <code>binsroot(1024)</code>
<code>binscont(# <i>numlist</i>)</code>	specify the number of bins to build the histogram for node splits for continuous predictors (<code>real</code> and <code>int</code> columns in H2O); default is <code>binscont(20)</code>

Tuning

<code>tune(<i>tune_opts</i>)</code>	specify hyperparameter tuning options for selecting the best-performing model
-------------------------------------	---

Only one of `validframe()` or `cv[]` is allowed.

If neither `validframe()` nor `cv[]` is specified, the evaluation metrics are reported for the training dataset.

When *numlist* is specified in one or more hyperparameter options, tuning is performed for those hyperparameters.

`collect` is allowed; see [U] [11.1.10 Prefix commands](#).

See [U] [20 Estimation and postestimation commands](#) for more capabilities of estimation commands.

<i>cvmethod</i>	Description
<code>random</code>	randomly split the training dataset into folds; the default
<code>modulo</code>	evenly split the training dataset into folds using the modulo operation
<code>stratify</code>	evenly distribute observations from the different classes of the response to all folds

<i>stop_opts</i>	Description
<code>metric(<i>metric_option</i>)</code>	specify the stopping metric for training or grid search
<code>tolerance(#)</code>	specify the tolerance value by which a model must improve before the training or grid search stops; default is <code>tolerance(1e-3)</code>

<i>tune_opts</i>	Description
<code>metric(<i>metric_option</i>)</code>	specify the metric for selecting the best-performing model
<code>grid(<i>gridspec</i>)</code>	specify whether to perform an exhaustive or random search for all hyperparameter combinations
<code>maxmodels(#)</code>	specify the maximum number of models considered in the grid search; default is all configurations
<code>maxtime(#)</code>	specify the maximum run time for the grid search in seconds; default is no time limit
<code>stop[(# [, <i>stop_opts</i>])]</code>	specify the number of iterations and other criteria for stopping GBM training if the stopping metric does not improve in the grid search
<code>parallel(#)</code>	specify the number of models to build in parallel during the grid search; default is <code>parallel(1)</code> , sequential model building
<code>nooutput</code>	suppress the table summarizing hyperparameter tuning

If any of `maxmodels()`, `maxtime()`, or `stop[]` is specified, then `grid(random)` is implied.

Options

Model

`validframe()`, `cv()`, `balanceclasses`, `h2orseed()`, `encode()`, `stop()`, `maxtime()`, `scoreevery()`, and `monotone()`; see [H2OML] [h2oml gbm](#).

Hyperparameter

`ntrees()`, `lrate()`, `lratedecay()`, `maxdepth()`, `minobsleaf()`, `predsamprate()`, `samprate()`, `minsplitthrshold()`, `binscat()`, `binsroot()`, and `binscont()`; see [H2OML] [h2oml gbm](#).

Tuning

`tune()`; see [H2OML] [h2oml gbm](#).
stata.com

Remarks and examples

For examples, see [Remarks and examples](#) in [H2OML] [h2oml gbm](#).

Stored results

`h2oml gbbinclass` stores the following in `e()`:

Scalars

<code>e(N_train)</code>	number of observations in the training frame
<code>e(N_valid)</code>	number of observations in the validation frame (with option <code>validframe()</code>)
<code>e(N_cv)</code>	number of observations in the cross-validation (with option <code>cv()</code>)
<code>e(n_cvfolds)</code>	number of cross-validation folds (with option <code>cv()</code>)
<code>e(k_predictors)</code>	number of predictors
<code>e(n_trees)</code>	number of trees
<code>e(n_trees_a)</code>	actual number of trees used in GBM
<code>e(maxdepth)</code>	maximum specified tree depth
<code>e(depth_min_a)</code>	achieved minimum tree depth
<code>e(depth_avg_a)</code>	achieved average depth among trees
<code>e(depth_max_a)</code>	achieved maximum tree depth
<code>e(minobsleaf)</code>	minimum specified number of observations for a child leaf
<code>e(lrate)</code>	learning rate
<code>e(lratedecay)</code>	learning rate decay
<code>e(samprate)</code>	observation sampling rate
<code>e(predsamprate)</code>	predictor sampling rate
<code>e(minsplitthr)</code>	minimum split improvement threshold
<code>e(binscat)</code>	number of bins for categorical predictors
<code>e(binsroot)</code>	number of bins for root node
<code>e(binscont)</code>	number of bins for continuous predictors
<code>e(h2orseed)</code>	H2O random-number seed
<code>e(maxtime)</code>	maximum run time
<code>e(balanceclass)</code>	1 if classes are balanced; 0 otherwise
<code>e(stop_iter)</code>	maximum iterations before stopping training without metric improvement
<code>e(stop_tol)</code>	tolerance for metric improvement before training stops
<code>e(scoreevery)</code>	number of trees before scoring metrics during training
<code>e(tune_h2orseed)</code>	random-number seed for tuning (with option <code>tune()</code>)
<code>e(tune_stop_iter)</code>	maximum iterations before stopping tuning without metric improvement (with option <code>tune()</code>)
<code>e(tune_stop_tol)</code>	tolerance for metric improvement before tuning stops (with option <code>tune()</code>)

<code>e(tune_maxtime)</code>	maximum run time for tuning grid search (with option <code>tune()</code>)
<code>e(tune_maxmodels)</code>	maximum number of models considered in tuning grid search (with option <code>tune()</code>)

Macros

<code>e(cmd)</code>	<code>h2oml gbbinclass</code>
<code>e(cmdline)</code>	command as typed
<code>e(subcmd)</code>	<code>gbbinclass</code>
<code>e(method)</code>	<code>gbm</code>
<code>e(method_type)</code>	classification
<code>e(class_type)</code>	binary
<code>e(method_full_name)</code>	Gradient boosting binary classification
<code>e(response)</code>	name of response
<code>e(predictors)</code>	names of predictors
<code>e(title)</code>	title in estimation output
<code>e(loss)</code>	name of the loss function
<code>e(train_frame)</code>	name of the training frame (with option <code>validframe()</code>)
<code>e(valid_frame)</code>	name of the validation frame (with option <code>cv()</code>)
<code>e(cv_method)</code>	fold assignment method (with option <code>cv()</code>)
<code>e(cv_varname)</code>	name of variable identifying cross-validation folds
<code>e(encoding_type)</code>	encoding type for categorical predictors
<code>e(monotone_inc)</code>	names of predictors with monotone increasing constraints
<code>e(monotone_dec)</code>	names of predictors with monotone decreasing constraints
<code>e(stop_metric)</code>	stopping metric for training
<code>e(tune_grid)</code>	grid search method used for tuning (with option <code>tune()</code>)
<code>e(tune_metric)</code>	name of the tuning metric (with option <code>tune()</code>)
<code>e(tune_stop_metric)</code>	stopping metric for tuning (with option <code>tune()</code>)
<code>e(properties)</code>	<code>nob noV</code>
<code>e(estat_cmd)</code>	program used to implement <code>h2omlestat</code>
<code>e(predict)</code>	program used to implement <code>h2omlpredict</code>
<code>e(marginsnotok)</code>	predictions disallowed by margins

Matrices

<code>e(metrics)</code>	training, validation, and cross-validation metrics
<code>e(hyperparam_table)</code>	minimum, maximum, and selected hyperparameter values

Also see

[H2OML] **h2oml postestimation** — Postestimation tools for `h2oml gbm` and `h2oml rf`⁺

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

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

[H2OML] **h2oml gbmclass** — Gradient boosting multiclass classification⁺

[H2OML] **h2oml gbregress** — Gradient boosting regression⁺

[H2OML] **h2oml rfbinclass** — Random forest binary classification⁺

[U] **20 Estimation and postestimation commands**

Stata, Stata Press, and Mata are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow and NetCourseNow are trademarks of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2023 StataCorp LLC, College Station, TX, USA. All rights reserved.

For suggested citations, see the FAQ on [citing Stata documentation](#).

