# h2omlgraph prcurve — Produce precision–recall curve plot

Description | Quick start | Menu | Syntax
Options | Remarks and examples | References | Also see

## Description

h2omlgraph prcurve plots the precision–recall curve after binary classification performed by h2oml gbbinclass and h2oml rfbinclass. With binary classification, the predicted probability for each observation is compared with a threshold value to determine whether the observation is predicted to be in the positive class or the negative class. Thus, for different threshold values, different numbers of observations are classified as positive and negative. Metrics based on the predicted classes, including precision (the proportion of correct predictions out of all observations predicted to be in the positive class) and recall (the true-positive rate), also depend on the selected threshold. Plotting the precision versus the recall for a variety of threshold values produces the precision–recall curve, which allows us to evaluate the tradeoff between precision and recall for a model.

The precision–recall curve is useful for evaluating model performance, especially for models fit to imbalanced response variables. A large area under the precision–recall curve (AUCPR) indicates good fit with both precision and recall being high.

## Quick start

Plot the precision–recall curve

```
h2omlgraph prcurve
```

Same as above, but plot the curve based on the validation data

```
h2omlgraph prcurve, valid
```

Same as above, but remove the reference line

```
h2omlgraph prcurve, valid norefline
```

## Menu

Statistics > H2O machine learning

# Syntax

`h2omlgraph prcurve` [ , *options* ]

| *options* | Description |
|---|---|
| Main | |
| models(*namelist*) | specify the name or a list of names of the stored estimation results |
| savedata(*filename*[ , replace ]) | save plot data to *filename* |
| Plot options | |
| rlopts(*line_options*) | affect rendition of reference line |
| norefline | suppress plotting reference line |
| lineopts(*line_options*) | affect rendition of all precision–recall curves |
| line#opts(*line_options*) | affect rendition of the precision–recall curve for model # |
| *twoway_options* | any options other than by() documented in [G-3] ***twoway_options*** |
| train | specify that precision and recall be reported using training results |
| valid | specify that precision and recall be reported using validation results |
| cv | specify that precision and recall be reported using cross-validation results |
| test | specify that precision and recall be computed using the testing frame |
| test(*framename*) | specify that precision and recall be computed using data in testing frame *framename* |
| frame(*framename*) | specify that precision and recall be computed using data in H2O frame *framename* |
| framelabel(*string*) | label frame as *string* in the output |

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

# Options

Main

models(*namelist*) specifies the name or a list of names of the stored estimation results for which the precision–recall curve is being plotted. For each model, the displayed curve corresponds to the default frame of that model when the h2omlpostestframe command has not been used to set a postestimation frame.

savedata(*filename*[ , replace ]) saves the plot data to a Stata data file (.dta file). replace specifies to overwrite the existing file.

Plot options

rlopts(*line_options*) affects the rendition of the reference line. See [G-3] ***line_options***.

norefline suppresses plotting the reference line. The reference line of the precision–recall curve is determined by the proportion of the response variable in the positive class, that is, the ratio of the number of positives to the total number of observations.

lineopts(*line_options*) affects the rendition of all precision–recall curves. See [G-3] ***line_options***.

`line#opts(`*line_options*`)` affects the rendition of the precision–recall curve for model #. See [G-3] ***line_options***.

*twoway_options* are any of the options documented in [G-3] ***twoway_options***, excluding `by()`. 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 prcurve` but are not shown in the dialog box:

`train`, `valid`, `cv`, `test`, `test()`, and `frame()` specify the H2O frame for which precision and recall are reported. Only one of `train`, `valid`, `cv`, `test`, `test()`, or `frame()` is allowed.

`train` specifies that precision and recall be reported using training results. This is the default when neither validation nor cross-validation is performed during estimation and when a postestimation frame has not been set with `h2omlpostestframe`.

`valid` specifies that precision and recall 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*.

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

`test` specifies that precision and recall 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 with `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 precision and recall 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, `h2omlpostestframe` provides a more convenient and computationally efficient process for doing this.

`frame(`*framename*`)` specifies that precision and recall be computed using the data in H2O frame *framename*.

`framelabel(`*string*`)` specifies the label to be used for the frame in the output. This option is not allowed with the `cv` option.

# Remarks and examples

After performing binary classification, the receiver operating characteristic (ROC) curve, introduced in [H2OML] **h2omlgraph roc**, is a common tool for evaluating model performance. However, the ROC curve is not reliable when the data are imbalanced (when the data contain very few positive classes). For imbalanced data, a small false-positive rate and a large true-positive rate are expected. Consequently, the ROC curve will be close to the upper-left corner and will indicate good fit rather than reflecting the true performance of the model. The precision–recall curve is designed to mitigate this problem by plotting the precision (the proportion of correct predictions out of all observations predicted to be in the positive class) versus the recall (the proportion of correct predictions out of all observations actually in the positive class; also known as the true-positive rate) (Davis and Goadrich 2006). The precision–recall curve is more reliable for imbalanced data compared with the ROC curve because the false-positive rate in the ROC curve is replaced with precision, which does not rely on the number of true negatives. (The number of true negatives will be large for imbalanced data and will strongly influence the false-positive rate.)

The computation of the precision and recall metrics relies on a threshold value. After binary classification, the predicted probability for each observation is compared with a threshold value to determine whether the observation is predicted to be in the positive class or the negative class. Observations with probabilities greater than the threshold are classified as positive, and the remaining observations are classified as negative. Different threshold values lead to different predicted classes. Therefore, as the threshold changes, the precision and recall also change.

The precision–recall curve plots the precision on the $y$ axis and the recall on the $x$ axis, where each metric is computed across a range of threshold values. When evaluating model performance, the closer the curve is to the upper-right corner, the better the performance. Similarly, the larger the AUCPR, the better the performance.

## ▷ Example 1: The precision–recall curve vs. the ROC

In this example, we compare ROC and precision–recall graphs for imbalanced data.

We use a popular credit card dataset available in Kaggle (Pozzolo et al. [2015], Pozzolo et al. [2018]) to predict whether a given credit card transaction is fraudulent.

The dataset contains 28 predictors, denoted `V1,...,V28`, which are obtained after a principal component analysis transformation. Due to confidentiality issues, the original predictors are not available. The response `fraud` is a binary variable that takes value 1 in the case of fraud and value 0 otherwise.

We start by opening the dataset in Stata and using the `tabulate` command to look at the distribution of the classes of `fraud`.

```
. use https://www.stata-press.com/data/r19/creditcard
(Credit card data)

. tabulate fraud

         Is |
 fraudulent |      Freq.     Percent        Cum.
------------+-----------------------------------
         No |    284,315       99.83       99.83
        Yes |        492        0.17      100.00
------------+-----------------------------------
      Total |    284,807      100.00
```

The data are highly imbalanced; only 0.17% of the response belongs to the class `yes`.

Next we put the data into an H2O frame. Recall that `h2o init` initiates an H2O cluster, `_h2oframe put` loads the current Stata dataset into an H2O frame, and `_h2oframe change` makes the specified frame the current H2O frame. We use the `_h2oframe split` command to randomly split the `credit` frame into a training frame (70% of observations) and a testing frame (30% of observations), which we name `train` and `test`, respectively. We also change the current frame to `train`. For details, see *Prepare your data for H2O machine learning in Stata* in [H2OML] **h2oml** and see [H2OML] **H2O setup**.

```
. h2o init
  (output omitted)

. _h2oframe put, into(credit)
Progress (%): 0 100
```

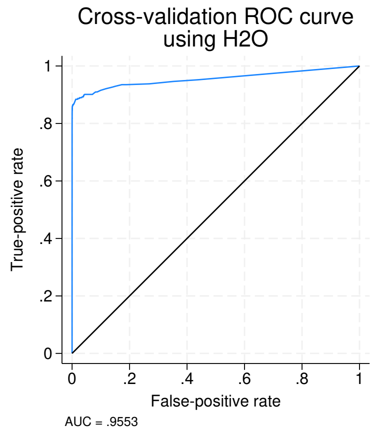
```
. _h2oframe split credit, into(train test) split(0.7 0.3) rseed(19)
. _h2oframe change train
```

We use random forest binary classification with 3-fold cross-validation to fit a model, and we specify `h2orseed()` for reproducibility. Because our goal is to compare ROC and precision–recall curves, we do not implement tuning. We store the estimation results by using the `h2omlest store` command.

```
. h2oml rfbinclass fraud v1-v28 amount, h2orseed(19) cv(3, modulo)
Progress (%): 0 0.4 1.4 4.5 10.4 17.4 25.0 29.4 34.4 38.4 43.0 50.4 56.4 62.0
> 66.5 70.9 75.0 76.4 83.4 88.9 96.4 100

Random forest binary classification using H2O

Response: fraud
Frame:                                  Number of observations:
  Training: train                                  Training = 199,612
                                           Cross-validation = 199,612
Cross-validation: Modulo                Number of folds     =       3

Model parameters

Number of trees       =   50
               actual =   50
Tree depth:                             Pred. sampling value =      -1
           Input max =   20             Sampling rate        =    .632
                 min =   19             No. of bins cat.     =   1,024
                 avg = 19.9             No. of bins root     =   1,024
                 max =   20             No. of bins cont.    =      20
Min. obs. leaf split =    1             Min. split thresh.   =  .00001

Metric summary

                                       Cross-
              Metric    Training   validation

            Log loss    .0057128     .0054806
    Mean class error    .0890433     .0904708
                 AUC     .940396     .9553414
              AUCPR     .8348062     .8391036
    Gini coefficient    .8807921     .9106828
                 MSE    .0004454     .0004531
                RMSE    .0211043     .0212871

. h2omlest store RF
```

Now we plot the ROC curve by using the `h2omlgraph roc` command.

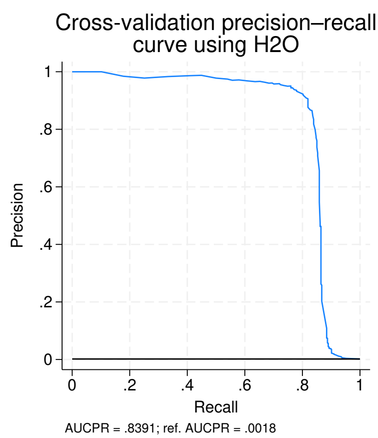
```
. h2omlgraph roc
```




As expected, the ROC curve fails to capture the imbalance in the response and shows good performance of the model.

On the other hand, the precision–recall curve, plotted below, shows an abrupt decrease in performance closer to the right side.

```
. h2omlgraph prcurve
```




The abrupt drop in precision when recall is greater than 0.8 suggests that the model's ability to distinguish between positive and negative classes diminishes substantially at certain thresholds.

The horizontal black line in the graph is the reference line. The reference line of the precision–recall curve is determined by the proportion of positive classes in the response (the ratio of the number of positives and the total number of observations). It corresponds to the model that always predicts a positive class.

Note that the h2omlgraph prcurve command by default plotted the precision and recall values based on cross-validation because the cv() option was specified and cross-validation was performed during estimation.

◁

## ▷ Example 2: Comparing models using the precision–recall curve

In example 1, we plotted the precision–recall curve for random forest binary classification. In practice, the precision–recall curve is often used to compare the performance of different models and methods on a testing frame. In this example, we compare the precision–recall curves for the random forest method and the gradient boosting machine (GBM) method.

We use the h2omlpostestframe command to set the testing frame for the random forest model estimated in example 1.

```
. h2omlpostestframe test
(testing frame test is now active for h2oml postestimation)
```

Then we perform gradient boosting binary classification and store the estimation results.

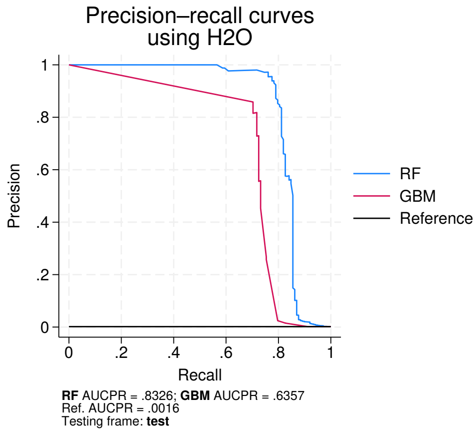
```
. h2oml gbbinclass fraud v1-v28 amount, h2orseed(19) cv(3, modulo)
Progress (%): 0 2.9 17.4 33.0 51.4 62.9 74.5 82.9 100
Gradient boosting binary classification using H2O
Response: fraud
Loss:     Bernoulli
Frame:                                       Number of observations:
  Training: train                                       Training = 199,612
                                                Cross-validation = 199,612
Cross-validation: Modulo                     Number of folds     =       3
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 | Cross-validation |
|---|---|---|
| Log loss | .0069067 | .0213072 |
| Mean class error | .0932605 | .1597576 |
| AUC | .9220793 | .8142659 |
| AUCPR | .8075749 | .5743456 |
| Gini coefficient | .8441585 | .6285319 |
| MSE | .0004101 | .0009271 |
| RMSE | .0202519 | .0304475 |

```
. h2omlest store GBM
. h2omlpostestframe test
(testing frame test is now active for h2oml postestimation)
```

To compare GBM and random forest, with default hyperparameters, we use `h2omlgraph prcurve` with the `models()` option.

```
. h2omlgraph prcurve, models(RF GBM)
```




Based on the graph above, random forest performs better than GBM.

◁

# References


Davis, J., and M. Goadrich. 2006. "The relationship between precision-recall and ROC curves". In *Proceedings of the 23rd International Conference on Machine Learning*, 233–240. New York: Association for Computing Machinery. https://doi.org/10.1145/1143844.1143874.

Pozzolo, A. D., G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi. 2018. Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems* 29: 3784–3797. https://doi.org/10.1109/tnnls.2017.2736643.

Pozzolo, A. D., O. Caelen, R. A. Johnson, and G. Bontempi. 2015. "Calibrating probability with undersampling for unbalanced classification". In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, 159–166. Piscataway, NJ: IEEE. https://doi.org/10.1109/SSCI.2015.33.


# Also see

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

[H2OML] **h2omlgraph roc** — Produce ROC curve plot