Package ‘BT’

January 20, 2025

Title (Adaptive) Boosting Trees Algorithm

Version 0.4

Date 2023-08-19

Author Gireg Willame [aut, cre, cph]

Maintainer Gireg Willame <gireg.willame@gmail.com>

Depends R (>= 4.0)

Imports rpart, stats, statmod, parallel

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Description Performs (Adaptive) Boosting Trees for Poisson distributed response variables, using log-link function.
The code approach is similar to the one used in 'gbm'/'gbm3'. Moreover, each tree in the expansion is built thanks to the 'rpart' package.
This package is based on following books and articles
Denuit, M., Hainaut, D., Trufin, J. (2019) <doi:10.1007/978-3-030-25820-7>
Denuit, M., Hainaut, D., Trufin, J. (2019) <doi:10.1007/978-3-030-57556-4>
Denuit, M., Hainaut, D., Trufin, J. (2019) <doi:10.1007/978-3-030-25827-6>
Denuit, M., Hainaut, D., Trufin, J. (2022) <doi:10.1080/03461238.2022.2037016>
Denuit, M., Huyghe, J., Trufin, J. (2022) <https://dial.uclouvain.be/pr/boreal/fr/object/boreal%3A244325/datastream/PDF_01/view>
Denuit, M., Trufin, J., Verdebout, T. (2022) <https://dial.uclouvain.be/pr/boreal/fr/object/boreal%3A268577>.

URL https://github.com/GiregWillame/BT/

BugReports https://github.com/GiregWillame/BT/issues/

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Config/testthat/edition 3

VignetteBuilder knitr

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2023-08-19 09:22:33 UTC

Contents

BT *(Adaptive) Boosting Trees (ABT/BT) Algorithm.*

Description

Performs the (Adaptive) Boosting Trees algorithm. This code prepares the inputs and calls the function `BT_call`. Each tree in the process is built thanks to the `rpart` function. In case of cross-validation, this function prepares the folds and performs multiple calls to the fitting function `BT_call`.

Usage

```
BT(
  formula = formula(data),
  data = list(),
  tweedie.power = 1,
  ABT = TRUE,
  n.iter = 100,
  train.fraction = 1,
  interaction.depth = 4,
  shrinkage = 1,
  bag.fraction = 1,
  colsample.bytree = NULL,
  keep.data = TRUE,
  is.verbose = FALSE,
  cv.folds = 1,
  folds.id = NULL,
  n.cores = 1,
```

```
  tree.control = rpart.control(xval = 0, maxdepth = (if (!is.null(interaction.depth)) {

       interaction.depth
 } else {
     10
 }), cp = -Inf, minsplit = 2),
  weights = NULL,
  seed = NULL,
  ...
)
```

Arguments

formula — a symbolic description of the model to be fit. Note that the offset isn't supported in this algorithm. Instead, everything is performed with a log-link function and a direct relationship exist between response, offset and weights.

data — an optional data frame containing the variables in the model. By default the variables are taken from `environment(formula)`, typically the environment from which `BT` is called. If `keep.data=TRUE` in the initial call to `BT` then `BT` stores a copy with the object (up to the variables used).

tweedie.power — Experimental parameter currently not used - Set to 1 referring to Poisson distribution.

ABT — a boolean parameter. If `ABT=TRUE` an adaptive boosting tree algorithm is built whereas if `ABT=FALSE` an usual boosting tree algorithm is run. By default, it is set to `TRUE`.

n.iter — the total number of iterations to fit. This is equivalent to the number of trees and the number of basis functions in the additive expansion. Please note that the initialization is not taken into account in the `n.iter`. More explicitly, a weighted average initializes the algorithm and then `n.iter` trees are built. Moreover, note that the `bag.fraction`, `colsample.bytree`, ... are not used for this initializing phase. By default, it is set to 100.

train.fraction — the first `train.fraction * nrows(data)` observations are used to fit the `BT` and the remainder are used for computing out-of-sample estimates (also known as validation error) of the loss function. By default, it is set to 1 meaning no out-of-sample estimates.

interaction.depth — the maximum depth of variable interactions: 1 builds an additive model, 2 builds a model with up to two-way interactions, etc. This parameter can also be interpreted as the maximum number of non-terminal nodes. By default, it is set to 4. Please note that if this parameter is `NULL`, all the trees in the expansion are built based on the `tree.control` parameter only, independently of the `ABT` value. This option is devoted to advanced users only and allows them to benefit from the full flexibility of the implemented algorithm.

shrinkage — a shrinkage parameter (in the interval (0,1]) applied to each tree in the expansion. Also known as the learning rate or step-size reduction. By default, it is set to 1.

bag.fraction — the fraction of independent training observations randomly selected to propose the next tree in the expansion. This introduces randomness into the model fit. If bag.fraction<1 then running the same model twice will result in similar but different fits. Please note that if this parameter is used the BTErrors$training.error corresponds to the normalized in-bag error and the out-of-bag improvements are computed and stored in BTErrors$oob.improvement. See BTFit for more details. By default, it is set to 1.

colsample.bytree
: each tree will be trained on a random subset of colsample.bytree number of features. Each tree will consider a new random subset of features from the formula, adding variability to the algorithm and reducing computation time. colsample.bytree will be bounded between 1 and the number of features considered in the formula. By default, it is set to NULL meaning no effect.

keep.data
: a boolean variable indicating whether to keep the data frames. This is particularly useful if one wants to keep track of the initial data frames and is further used for predicting in case any data frame is specified. Note that in case of cross-validation, if keep.data=TRUE the initial data frames are saved whereas the cross-validation samples are not. By default, it is set to FALSE.

is.verbose
: if is.verbose=TRUE, the BT will print out the algorithm progress. By default, it is set to FALSE.

cv.folds
: a positive integer representing the number of cross-validation folds to perform. If cv.folds>1 then BT, in addition to the usual fit, will perform a cross-validation and calculate an estimate of generalization error returned in BTErrors$cv.error. By default, it is set to 1 meaning no cross-validation.

folds.id
: an optional vector of values identifying what fold each observation is in. If supplied, this parameter prevails over cv.folds. By default, folds.id = NULL meaning that no folds are defined.

n.cores
: the number of cores to use for parallelization. This parameter is used during the cross-validation. This parameter is bounded between 1 and the maximum number of available cores. By default, it is set to 1 leading to a sequential approach.

tree.control
: for advanced user only. It allows to define additional tree parameters that will be used at each iteration. See rpart.control for more information.

weights
: optional vector of weights used in the fitting process. These weights must be positive but do not need to be normalized. By default, it is set to NULL which corresponds to an uniform weight of 1 for each observation.

seed
: optional number used as seed. Please note that if cv.folds>1, the parLapply function is called. Therefore, the seed (if defined) used inside each fold will be a multiple of the seed parameter.

...
: not currently used.

Details

The NA values are currently dropped using na.omit.

Value

a BTFit object.

Author(s)

Gireg Willame <gireg.willame@gmail.com>

This package is inspired by the gbm3 *package. For more details, see* https://github.com/gbm-developers/gbm3/.

References

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2022). **Response versus gradient boosting trees, GLMs and neural networks under Tweedie loss and log-link**. Accepted for publication in *Scandinavian Actuarial Journal*.

M. Denuit, J. Huyghe and J. Trufin (2022). **Boosting cost-complexity pruned trees on Tweedie responses: The ABT machine for insurance ratemaking**. Paper submitted for publication.

M. Denuit, J. Trufin and T. Verdebout (2022). **Boosting on the responses with Tweedie loss functions**. Paper submitted for publication.

See Also

BTFit, BTCVFit, BT_call, BT_perf, predict.BTFit, summary.BTFit, print.BTFit, .BT_cv_errors.

Examples

```
## Load dataset.
dataset <- BT::BT_Simulated_Data

## Fit a Boosting Tree model.
BT_algo <- BT(formula = Y_normalized ~ Age + Sport + Split + Gender, # formula
              data = dataset, # data
              ABT = FALSE, # Classical Boosting Tree
              n.iter = 200,
              train.fraction = 0.8,
              interaction.depth = 3,
              shrinkage = 0.01,
              bag.fraction = 0.5,
              colsample.bytree = 2, # 2 explanatory variable used at each iteration.
              keep.data = FALSE, # Do not keep a data copy.
              is.verbose = FALSE, # Do not print progress.
              cv.folds = 3, # 3-cv will be performed.
              folds.id = NULL ,
              n.cores = 1,
              weights = ExpoR, # <=> Poisson model on response Y with ExpoR in offset.
              seed = NULL)
```

```
## Determine the model performance and plot results.
best_iter_val <- BT_perf(BT_algo, method='validation')
best_iter_oob <- BT_perf(BT_algo, method='OOB', oobag.curve = TRUE)
best_iter_cv <- BT_perf(BT_algo, method ='cv', oobag.curve = TRUE)

best_iter <- best_iter_val

## Variable influence and plot results.
# Based on the first iteration.
variable_influence1 <- summary(BT_algo, n.iter = 1)
# Using all iterations up to best_iter.
variable_influence_best_iter <- summary(BT_algo, n.iter = best_iter)

##  Print results : call, best_iters and summarized relative influence.
print(BT_algo)

## Model predictions.
# Predict on the link scale, using only the best_iter tree.
pred_single_iter <- predict(BT_algo, newdata = dataset,
                            n.iter = best_iter, type = 'link', single.iter = TRUE)
# Predict on the response scale, using the first best_iter.
pred_best_iter <- predict(BT_algo, newdata = dataset,
                          n.iter = best_iter, type = 'response')
```

BTCVFit *BTCVFit*

Description

These are objects representing CV fitted boosting trees.

Details

CV (Adaptive) Boosting Tree Model Object.

Value

a list of `BTFit` objects with each element corresponding to a specific BT fit on a particular fold

Structure

The following components must be included in a legitimate `BTCVFit` object.

Author(s)

Gireg Willame <gireg.willame@gmail.com>

This package is inspired by the `gbm3` *package. For more details, see* https://github.com/gbm-developers/gbm3/.

References

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2022). **Response versus gradient boosting trees, GLMs and neural networks under Tweedie loss and log-link**. Accepted for publication in *Scandinavian Actuarial Journal*.

M. Denuit, J. Huyghe and J. Trufin (2022). **Boosting cost-complexity pruned trees on Tweedie responses: The ABT machine for insurance ratemaking**. Paper submitted for publication.

M. Denuit, J. Trufin and T. Verdebout (2022). **Boosting on the responses with Tweedie loss functions**. Paper submitted for publication.

See Also

`BT`.

`BTFit` *BTFit*

Description

These are objects representing fitted boosting trees.

Details

Boosting Tree Model Object.

Value

- `BTInit` — an object of class `BTInit` containing the initial fitted value `initFit`, the initial `training.error` and the initial `validation.error` if any.
- `BTErrors` — an object of class `BTErrors` containing the vectors of errors for each iteration performed (excl. the initialization). More precisely, it contains the `training.error`, `validation.error` if `train.fraction`<1 and the `oob.improvement` if `bag.fraction` < 1. Moreover, if a cross-validation approach was performed, a vector of cross-validation errors `cv.error` as a function of boosting iteration is also stored in this object.
- `BTIndivFits` — an object of class `BTIndivFits` containing the list of each individual tree fitted at each boosting iteration.
- `distribution` — the Tweedie power (and so the distribution) that has been used to perform the algorithm. It will currently always output 1.

var.names	a vector containing the names of the explanatory variables.
response	the name of the target/response variable.
w	a vector containing the weights used.
seed	the used seed, if any.
BTData	if keep.data=TRUE, an object of class BTData containing the training.set and validation.set (can be NULL if not used). These data frames are reduced to the used variables, that are the response and explanatory variables. Note that in case of cross-validation, even if keep.data=TRUE the folds will not be kept. In fact, only the data frames related to the original fit (i.e. on the whole training set) will be saved.
BTParams	an object of class BTParams containing all the (Adaptive) boosting tree parameters. More precisely, it contains the ABT, train.fraction, shrinkage, interaction.depth, bag.fraction, n.iter, colsample.bytree and tree.control parameter values.
keep.data	the keep.data parameter value.
is.verbose	the is.verbose parameter value.
fitted.values	the training set fitted values on the score scale using all the n.iter (and initialization) iterations.
cv.folds	the number of cross-validation folds. Set to 1 if no cross-validation performed.
call	the original call to the BT algorithm.
Terms	the model.frame terms argument.
folds	a vector of values identifying to which fold each observation is in. This argument is not present if there is no cross-validation. On the other hand, it corresponds to folds.id if it was initially defined by the user.
cv.fitted	a vector containing the cross-validation fitted values, if a cross-validation was performed. More precisely, for a given observation, the prediction will be furnished by the cv-model for which this specific observation was out-of-fold. See predict.BTCVFit for more details.

Structure

The following components must be included in a legitimate BTFit object.

Author(s)

Gireg Willame <gireg.willame@gmail.com>

This package is inspired by the gbm3 *package. For more details, see* https://github.com/gbm-developers/gbm3/.

References

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2022). **Response versus gradient boosting trees, GLMs and neural networks under Tweedie loss and log-link**. Accepted for publication in *Scandinavian Actuarial Journal*.

M. Denuit, J. Huyghe and J. Trufin (2022). **Boosting cost-complexity pruned trees on Tweedie responses: The ABT machine for insurance ratemaking**. Paper submitted for publication.

M. Denuit, J. Trufin and T. Verdebout (2022). **Boosting on the responses with Tweedie loss functions**. Paper submitted for publication.

See Also

BT.

`BT_call` *(Adaptive) Boosting Trees (ABT/BT) fit.*

Description

Fit a (Adaptive) Boosting Trees algorithm. This is for "power" users who have a large number of variables and wish to avoid calling `model.frame` which can be slow in this instance. This function is in particular called by BT. It is mainly split in two parts, the first one considers the initialization (see `BT_callInit`) whereas the second performs all the boosting iterations (see `BT_callBoosting`). By default, this function does not perform input checks (those are all done in BT) and all the parameters should be given in the right format. We therefore suppose that the user is aware of all the choices made.

Usage

```
BT_call(
  training.set,
  validation.set,
  tweedie.power,
  respVar,
  w,
  explVar,
  ABT,
  tree.control,
  train.fraction,
  interaction.depth,
  bag.fraction,
  shrinkage,
  n.iter,
  colsample.bytree,
  keep.data,
  is.verbose
```

```
)

BT_callInit(training.set, validation.set, tweedie.power, respVar, w)

BT_callBoosting(
  training.set,
  validation.set,
  tweedie.power,
  ABT,
  tree.control,
  interaction.depth,
  bag.fraction,
  shrinkage,
  n.iter,
  colsample.bytree,
  train.fraction,
  keep.data,
  is.verbose,
  respVar,
  w,
  explVar
)
```

Arguments

training.set	a data frame containing all the related variables on which one wants to fit the algorithm.
validation.set	a held-out data frame containing all the related variables on which one wants to assess the algorithm performance. This can be NULL.
tweedie.power	Experimental parameter currently not used - Set to 1 referring to Poisson distribution.
respVar	the name of the target/response variable.
w	a vector of weights.
explVar	a vector containing the name of explanatory variables.
ABT	a boolean parameter. If ABT=TRUE an adaptive boosting tree algorithm is built whereas if ABT=FALSE an usual boosting tree algorithm is run.
tree.control	allows to define additional tree parameters that will be used at each iteration. See rpart.control for more information.
train.fraction	the first train.fraction * nrows(data) observations are used to fit the BT and the remainder are used for computing out-of-sample estimates (also known as validation error) of the loss function. It is mainly used to report the value in the BTFit object.
interaction.depth	the maximum depth of variable interactions: 1 builds an additive model, 2 builds a model with up to two-way interactions, etc. This parameter can also be interpreted as the maximum number of non-terminal nodes. By default, it is set to 4.

Please note that if this parameter is NULL, all the trees in the expansion are built based on the tree.control parameter only. This option is devoted to advanced users only and allows them to benefit from the full flexibility of the implemented algorithm.

bag.fraction
: the fraction of independent training observations randomly selected to propose the next tree in the expansion. This introduces randomness into the model fit. If bag.fraction<1 then running the same model twice will result in similar but different fits. BT uses the R random number generator, so set.seed ensures the same model can be reconstructed. Please note that if this parameter is used the BTErrors$training.error corresponds to the normalized in-bag error.

shrinkage
: a shrinkage parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction.

n.iter
: the total number of iterations to fit. This is equivalent to the number of trees and the number of basis functions in the additive expansion. Please note that the initialization is not taken into account in the n.iter. More explicitly, a weighted average initializes the algorithm and then n.iter trees are built. Moreover, note that the bag.fraction, colsample.bytree, ... are not used for this initializing phase.

colsample.bytree
: each tree will be trained on a random subset of colsample.bytree number of features. Each tree will consider a new random subset of features from the formula, adding variability to the algorithm and reducing computation time. colsample.bytree will be bounded between 1 and the number of features considered.

keep.data
: a boolean variable indicating whether to keep the data frames. This is particularly useful if one wants to keep track of the initial data frames and is further used for predicting in case any data frame is specified. Note that in case of cross-validation, if keep.data=TRUE the initial data frames are saved whereas the cross-validation samples are not.

is.verbose
: if is.verbose=TRUE, the BT will print out the algorithm progress.

Value

a BTFit object.

Author(s)

Gireg Willame <gireg.willame@gmail.com>

This package is inspired by the gbm3 *package. For more details, see* https://github.com/gbm-developers/gbm3/.

References

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2022). **Response versus gradient boosting trees, GLMs and neural networks under Tweedie loss and log-link**. Accepted for publication in *Scandinavian Actuarial Journal*.

M. Denuit, J. Huyghe and J. Trufin (2022). **Boosting cost-complexity pruned trees on Tweedie responses: The ABT machine for insurance ratemaking**. Paper submitted for publication.

M. Denuit, J. Trufin and T. Verdebout (2022). **Boosting on the responses with Tweedie loss functions**. Paper submitted for publication.

See Also

BTFit, BTCVFit, BT_perf, predict.BTFit, summary.BTFit, print.BTFit, .BT_cv_errors.

BT_devTweedie *Deviance function for the Tweedie family.*

Description

Compute the deviance for the Tweedie family case.

Usage

```
BT_devTweedie(y, mu, tweedieVal, w = NULL)
```

Arguments

y	a vector containing the observed values.
mu	a vector containing the fitted values.
tweedieVal	a numeric representing the Tweedie Power. It has to be a positive number outside of the interval]0,1[.
w	an optional vector of weights.

Details

This function computes the Tweedie related deviance. The latter is defined as:

$$d(y, mu, w) = w(y - mu)^2, if tweedieVal = 0;$$
$$d(y, mu, w) = 2w(y log(y/mu) + mu - y), if tweedieVal = 1;$$
$$d(y, mu, w) = 2w(log(mu/y) + y/mu - 1), if tweedieVal = 2;$$
$$d(y, mu, w) = 2w(max(y, 0)^(2-p)/((1-p)(2-p)) - y mu^(1-p)/(1-p) + mu^(2-p)/(2-p)), else.$$

Value

A vector of individual deviance contribution.

Author(s)

Gireg Willame <gireg.willame@gmail.com>

This package is inspired by the gbm3 *package. For more details, see* https://github.com/gbm-developers/gbm3/.

References

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2022). **Response versus gradient boosting trees, GLMs and neural networks under Tweedie loss and log-link**. Accepted for publication in *Scandinavian Actuarial Journal*.

M. Denuit, J. Huyghe and J. Trufin (2022). **Boosting cost-complexity pruned trees on Tweedie responses: The ABT machine for insurance ratemaking**. Paper submitted for publication.

M. Denuit, J. Trufin and T. Verdebout (2022). **Boosting on the responses with Tweedie loss functions**. Paper submitted for publication.

See Also

BT, BT_call.

BT_more — *Perform additional boosting iterations.*

Description

Method to perform additional iterations of the Boosting Tree algorithm, starting from an initial BTFit object. This does not support further cross-validation. Moreover, this approach is only allowed if keep.data=TRUE in the original call.

Usage

```
BT_more(BTFit_object, new.n.iter = 100, is.verbose = FALSE, seed = NULL)
```

Arguments

BTFit_object	a BTFit object.
new.n.iter	number of new boosting iterations to perform.
is.verbose	a logical specifying whether or not the additional fitting should run "noisely" with feedback on progress provided to the user.
seed	optional seed used to perform the new iterations. By default, no seed is set.

Value

Returns a new BTFit object containing the initial call as well as the new iterations performed.

Author(s)

Gireg Willame <gireg.willame@gmail.com>

This package is inspired by the gbm3 *package. For more details, see* https://github.com/gbm-developers/gbm3/.

References

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2022). **Response versus gradient boosting trees, GLMs and neural networks under Tweedie loss and log-link**. Accepted for publication in *Scandinavian Actuarial Journal*.

M. Denuit, J. Huyghe and J. Trufin (2022). **Boosting cost-complexity pruned trees on Tweedie responses: The ABT machine for insurance ratemaking**. Paper submitted for publication.

M. Denuit, J. Trufin and T. Verdebout (2022). **Boosting on the responses with Tweedie loss functions**. Paper submitted for publication.

See Also

BT, BTFit.

BT_perf *Performance assessment.*

Description

Function to compute the performances of a fitted boosting tree.

Usage

```
BT_perf(
  BTFit_object,
  plot.it = TRUE,
  oobag.curve = FALSE,
  overlay = TRUE,
  method,
  main = ""
)
```

Arguments

- `BTFit_object` — a `BTFit` object resulting from an initial call to `BT`
- `plot.it` — a boolean indicating whether to plot the performance measure. Setting `plot.it = TRUE` creates two plots. The first one plots the `object$BTErrors$training.error` (in black) as well as the `object$BTErrors$validation.error` (in red) and/or the `object$BTErrors$cv.error` (in green) depending on the `method` and parametrization. These values are plotted as a function of the iteration number. The scale of the error measurement, shown on the left vertical axis, depends on the arguments used in the initial call to `BT` and the chosen `method`.
- `oobag.curve` — indicates whether to plot the out-of-bag performance measures in a second plot. Note that this option makes sense if the `bag.fraction` was properly defined in the initial call to `BT`.
- `overlay` — if set to `TRUE` and `oobag.curve=TRUE` then a right y-axis is added and the estimated cumulative improvement in the loss function is plotted versus the iteration number.
- `method` — indicates the method used to estimate the optimal number of boosting iterations. Setting `method = "OOB"` computes the out-of-bag estimate and `method = "validation"` uses the validation dataset to compute an out-of-sample estimate. Finally, setting `method = "cv"` extracts the optimal number of iterations using cross-validation, if `BT` was called with `cv.folds > 1`. If missing, a guessing method is applied.
- `main` — optional parameter that allows the user to define specific plot title.

Value

Returns the estimated optimal number of iterations. The method of computation depends on the `method` argument.

Author(s)

Gireg Willame <g.willame@detralytics.eu>

This package is inspired by the `gbm3` *package. For more details, see* https://github.com/gbm-developers/gbm3/.

References

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2022). **Response versus gradient boosting trees, GLMs and neural networks under Tweedie loss and log-link**. Accepted for publication in *Scandinavian Actuarial Journal*.

M. Denuit, J. Huyghe and J. Trufin (2022). **Boosting cost-complexity pruned trees on Tweedie responses: The ABT machine for insurance ratemaking**. Paper submitted for publication.

M. Denuit, J. Trufin and T. Verdebout (2022). **Boosting on the responses with Tweedie loss functions**. Paper submitted for publication.

See Also

BT, BT_call.

BT_Simulated_Data *Simulated Database.*

Description

A simulated database used for examples and vignettes. The variables are related to a motor insurance pricing context.

Usage

```
BT_Simulated_Data
```

Format

A simulated data frame with 50,000 rows and 7 columns, containing simulation of different policyholders:

Gender Gender, varying between male and female.

Age Age, varying from 18 to 65years old.

Split Noisy variable, not used to simulate the response variable. It allows to assess how the algorithm handle these features.

Sport Car type, varying between yes (sport car) or no.

ExpoR Yearly exposure-to-risk, varying between 0 and 1.

Y Yearly claim number, simulated thanks to Poisson distribution.

Y_normalized Yearly claim frequency, corresponding to the ratio between Y and ExpoR.

predict.BTFit *Predict method for BT Model fits.*

Description

Predicted values based on a boosting tree model object.

Usage

```
## S3 method for class 'BTFit'
predict(object, newdata, n.iter, type = "link", single.iter = FALSE, ...)
```

Arguments

object	a BTFit object.
newdata	data frame of observations for which to make predictions. If missing or not a data frame, if keep.data=TRUE in the initial fit then the original training set will be used.
n.iter	number of boosting iterations used for the prediction. This parameter can be a vector in which case predictions are returned for each iteration specified.
type	the scale on which the BT makes the predictions. Can either be "link" or "response". Note that, by construction, a log-link function is used during the fit.
single.iter	if single.iter=TRUE then predict.BTFit returns the predictions from the single tree n.iter.
...	not currently used.

Details

predict.BTFit produces a predicted values for each observation in newdata using the first n.iter boosting iterations. If n.iter is a vector then the result is a matrix with each column corresponding to the BT predictions with n.iter[1] boosting iterations, n.iter[2] boosting iterations, and so on.

As for the fit, the predictions do not include any offset term. In the Poisson case, please remind that a weighted approach is initially favored.

Value

Returns a vector of predictions. By default, the predictions are on the score scale. If type = "response", then BT converts back to the same scale as the outcome. Note that, a log-link is supposed by construction.

Author(s)

Gireg Willame <gireg.willame@gmail.com>

This package is inspired by the gbm3 *package. For more details, see* https://github.com/gbm-developers/gbm3/.

References

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2022). **Response versus gradient boosting trees, GLMs and neural networks under Tweedie loss and log-link**. Accepted for publication in *Scandinavian Actuarial Journal*.

M. Denuit, J. Huyghe and J. Trufin (2022). **Boosting cost-complexity pruned trees on Tweedie responses: The ABT machine for insurance ratemaking**. Paper submitted for publication.

M. Denuit, J. Trufin and T. Verdebout (2022). **Boosting on the responses with Tweedie loss functions**. Paper submitted for publication.

See Also

`BT`, `BTFit`.

`print.BTFit` *Printing function.*

Description

Function to print the BT results.

Usage

```
## S3 method for class 'BTFit'
print(x, ...)
```

Arguments

`x`	a `BTFit` object.
`...`	arguments passed to `print.default`.

Details

Print the different input parameters as well as obtained results (best iteration/performance & relative influence) given the chosen approach.

Value

No value returned.

Author(s)

Gireg Willame <gireg.willame@gmail.com>

This package is inspired by the gbm3 *package. For more details, see* https://github.com/gbm-developers/gbm3/.

References

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions**, *Springer Actuarial*.

M. Denuit, D. Hainaut and J. Trufin (2022). **Response versus gradient boosting trees, GLMs and neural networks under Tweedie loss and log-link**. Accepted for publication in *Scandinavian Actuarial Journal*.

M. Denuit, J. Huyghe and J. Trufin (2022). **Boosting cost-complexity pruned trees on Tweedie responses: The ABT machine for insurance ratemaking**. Paper submitted for publication.

M. Denuit, J. Trufin and T. Verdebout (2022). **Boosting on the responses with Tweedie loss functions**. Paper submitted for publication.

See Also

BT, .BT_relative_influence, BT_perf.

---

summary.BTFit *Summary of a BTFit object.*

Description

Computes the relative influence of each variable in the BTFit object.

Usage

```
## S3 method for class 'BTFit'
summary(
  object,
  cBars = length(object$var.names),
  n.iter = object$BTParams$n.iter,
  plot_it = TRUE,
  order_it = TRUE,
  method = .BT_relative_influence,
  normalize = TRUE,
  ...
)
```

Arguments

object	a BTFit object.
cBars	the number of bars to plot. If order=TRUE only the variables with the cBars largest relative influence will appear in the barplot. If order=FALSE then the first cBars variables will appear in the barplot.
n.iter	the number of trees used to compute the relative influence. Only the first n.iter trees will be used.
plot_it	an indicator as to whether the plot is generated.
order_it	an indicator as to whether the plotted and/or returned relative influences are sorted.
method	the function used to compute the relative influence. Currently, only .BT_relative_influence is available (default value as well).
normalize	if TRUE returns the normalized relative influence.
...	additional argument passed to the plot function.

Details

Please note that the relative influence for variables having an original **negative** relative influence is forced to 0.

Value

Returns a data frame where the first component is the variable name and the second one is the computed relative influence, normalized to sum up to 100. Depending on the plot_it value, the relative influence plot will be performed.

Author(s)

Gireg Willame <gireg.willame@gmail.com>

This package is inspired by the gbm3 *package. For more details, see* https://github.com/gbm-developers/gbm3/.

References

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions**, *Springer Actuarial.*

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions**, *Springer Actuarial.*

M. Denuit, D. Hainaut and J. Trufin (2019). **Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions**, *Springer Actuarial.*

M. Denuit, D. Hainaut and J. Trufin (2022). **Response versus gradient boosting trees, GLMs and neural networks under Tweedie loss and log-link**. Accepted for publication in *Scandinavian Actuarial Journal.*

M. Denuit, J. Huyghe and J. Trufin (2022). **Boosting cost-complexity pruned trees on Tweedie responses: The ABT machine for insurance ratemaking**. Paper submitted for publication.

M. Denuit, J. Trufin and T. Verdebout (2022). **Boosting on the responses with Tweedie loss functions**. Paper submitted for publication.

See Also

`BT`, `.BT_relative_influence`.

Index