Package 'brif'

January 13, 2023

Type Package

Title A Tree and Forest Tool for Classification and Regression

Version 1.4.0

Date 2022-12-25

Maintainer Yanchao Liu <yanchaoliu@wayne.edu>

Description Build decision trees and random forests for classification and regression. The implementation strikes a balance between minimizing computing efforts and maximizing the expected predictive accuracy, thus scales well to large data sets. Multi-threading is available through 'OpenMP' <https://gcc.gnu.org/wiki/openmp>.

License GPL (>= 2)

Imports Rcpp (>= 1.0.9)

LinkingTo Rcpp

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation yes

Author Yanchao Liu [aut, cre] (<https://orcid.org/0000-0002-3256-9336>)

Repository CRAN

Date/Publication 2023-01-13 08:10:02 UTC

R topics documented:

brif-package 2
brif 2
brif.default 3
brif.formula 6
brif.trainpredict 7
brifTree 10
brifTree.default 10
brifTree.formula 11
brif_write_data 12
predict.brif 13

printBrifTree 14
printRules 15
rfpredict 15
rftrain 16
rftrainpredict 16
stratpar 17

Index **18**

`brif-package` *brif: A tree and forest tool for classification and regression*

Description

Build decision trees and random forests for classification and regression. The implementation strikes a balance between minimizing computing efforts and maximizing the expected predictive accuracy, thus scales well to large data sets. Multi-threading is available through 'OpenMP'.

Available functions

Use `brif` to build a random forest and (optionally) make predictions. Use `brifTree` to build a single decision tree. Use `printRules` to print out the decision rules of a tree. Use `predict.brif` to make predictions using a brif model (tree or forest).

Author(s)

Yanchao Liu

`brif` *Build a model (and make predictions)*

Description

Depending on the arguments supplied, the function `brif.formula`, `brif.default` or `brif.trainpredict` will be called.

Usage

```
brif(x, ...)
```

Arguments

`x`	a data frame or a `formula` object.
`...`	arguments passed on to `brif.formula`, `brif.default` or `brif.trainpredict`.

Value

a data frame, a vector or a list. If newdata is supplied, prediction results for newdata will be returned in a data frame or a vector, depending on the problem type (classification or regression) and the type argument; otherwise, an object of class "brif" is returned, which is to be used in the function predict.brif for making predictions. See brif.default for components of the "brif" object.

Examples

```
trainset <- sample(1:nrow(iris), 0.5*nrow(iris))
validset <- setdiff(1:nrow(iris), trainset)

# Train and predict at once
pred_scores <- brif(Species~., data = iris, subset = trainset,
                    newdata = iris[validset, 1:4], type = 'score')
pred_labels <- brif(Species~., data = iris, subset = trainset,
                    newdata = iris[validset, 1:4], type = 'class')

# Confusion matrix
table(pred_labels, iris[validset, 5])

# Accuracy
sum(pred_labels == iris[validset, 5])/length(validset)

# Train using the formula format
bf <- brif(Species~., data = iris, subset = trainset)

# Or equivalently, train using the data.frame format
bf <- brif(iris[trainset, c(5,1:4)])

# Make a prediction
pred_scores <- predict(bf, iris[validset, 1:4], type = 'score')
pred_labels <- predict(bf, iris[validset, 1:4], type = 'class')

# Regression
bf <- brif(mpg ~., data = mtcars)
pred <- predict(bf, mtcars[2:11])
plot(pred, mtcars$mpg)
abline(0, 1)

# Optionally, delete the model object to release memory
rm(list = c("bf"))
gc()
```

brif.default *Build a model taking a data frame as input*

Description

Build a model taking a data frame as input

Usage

```
## Default S3 method:
brif(
  x,
  n_numeric_cuts = 31,
  n_integer_cuts = 31,
  max_integer_classes = 20,
  max_depth = 20,
  min_node_size = 1,
  ntrees = 200,
  ps = 0,
  max_factor_levels = 30,
  seed = 0,
  bagging_method = 0,
  bagging_proportion = 0.9,
  split_search = 4,
  search_radius = 5,
  verbose = 0,
  nthreads = 2,
  ...
)
```

Arguments

`x` a data frame containing the training data set. The first column is taken as the target variable and all other columns are used as predictors.

`n_numeric_cuts` an integer value indicating the maximum number of split points to generate for each numeric variable.

`n_integer_cuts` an integer value indicating the maximum number of split points to generate for each integer variable.

`max_integer_classes`
an integer value. If the target variable is integer and has more than max_integer_classes unique values in the training data, then the target variable will be grouped into max_integer_classes bins. If the target variable is numeric, then the smaller of max_integer_classes and the number of unique values number of bins will be created on the target variables and the regression problem will be solved as a classification problem.

`max_depth` an integer specifying the maximum depth of each tree. Maximum is 40.

`min_node_size` an integer specifying the minimum number of training cases a leaf node must contain.

`ntrees` an integer specifying the number of trees in the forest.

`ps` an integer indicating the number of predictors to sample at each node split. Default is 0, meaning to use sqrt(p), where p is the number of predictors in the input.

max_factor_levels
: an integer. If any factor variables has more than max_factor_levels, the program stops and prompts the user to increase the value of this parameter if the too-many-level factor is indeed intended.

seed
: an integer specifying the seed used by the internal random number generator. Default is 0, meaning not to set a seed but to accept the set seed from the calling environment.

bagging_method
: an integer indicating the bagging sampling method: 0 for sampling without replacement; 1 for sampling with replacement (bootstrapping).

bagging_proportion
: a numeric scalar between 0 and 1, indicating the proportion of training observations to be used in each tree.

split_search
: an integer indicating the choice of the split search method. 0: randomly pick a split point; 1: do a local search; 2: random pick subject to regulation; 3: local search subject to regulation; 4 or above: a mix of options 0 to 3.

search_radius
: an positive integer indicating the split point search radius. This parameter takes effect only in the self-regulating local search (split_search = 2 or above).

verbose
: an integer (0 or 1) specifying the verbose level.

nthreads
: an integer specifying the number of threads used by the program. This parameter takes effect only on systems supporting OpenMP.

...
: additional arguments.

Value

an object of class `brif`, which is a list containing the following components. Note: this object is not intended for any use other than that by the function `predict.brif`. Do not apply the `str` function on this object because the output can be long and meaningless especially when ntrees is large. Use `summary` to get a peek of its structure. Use `printRules` to print out the decision rules of a particular tree. Most of the data in the object is stored in the tree_leaves element (which is a list of lists by itself) of this list.

p
: an integer scalar, the number of variables (predictors) used in the model

var_types
: an character vector of length (p+1) containing the variable names, including the target variable name as its first element

var_labels
: an character vector of length (p+1) containing the variable types, including that of the target variable as its first element

n_bcols
: an integer vector of length (p+1), containing the numbers of binary columns generated for each variable

ntrees
: an integer scalar indicating the number of trees in the model

index_in_group
: an integer vector specifying the internal index, for each variable, in its type group

numeric_cuts
: a list containing split point information on numeric variables

integer_cuts
: a list containing split point information on integer variables

factor_cuts
: a list containing split point information on factor variables

n_num_vars an integer scalar indicating the numeric variables in the model

n_int_vars an integer scalar indicating the integer variables in the model

n_fac_vars an integer scalar indicating the factor variables in the model

tree_leaves a list containing all the leaves in the forest

yc a list containing the target variable encoding scheme

brif.formula *Build a model (and make predictions) with formula*

Description

Build a model (and make predictions) with formula

Usage

```
## S3 method for class 'formula'
brif(
  formula,
  data,
  subset,
  na.action = stats::na.pass,
  newdata = NULL,
  type = c(”score”, ”class”),
  ...
)
```

Arguments

formula an object of class "formula": a symbolic description of the model to be fitted.

data an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which brif.formula is called.

subset an optional vector specifying a subset (in terms of index numbers, not actual data) of observations to be used in the fitting process.

na.action a function which indicates what should happen when the data contain NAs.

newdata a data frame containing the data set for prediction. Default is NULL. If newdata is supplied, prediction results will be returned.

type a character string specifying the prediction format, which takes effect only when newdata is supplied. Available values include "score" and "class". Default is "score".

... additional algorithmic parameters. See brif.default for a complete list.

Value

an object of class brif to be used by predict.brif.

Examples

```
bf <- brif(Species ~ ., data = iris)
pred <- predict(bf, iris[,1:4])
```

---

brif.trainpredict *Train a model and use it to predict new cases*

Description

If the model is built to predict for just one test data set (newdata), then this function should be used instead of the brif and predict.brif pipeline. Transporting the model object between the training and prediction functions through saving and loading the brif object takes a subtantial amount of time, and using the pred.trainpredict function eliminates such time-consuming operations. This function will be automatically invoked by the brif function when the newdata argument is supplied there. If GPU is used for training (GPU = 1 or 2), the total execution time of this function includes writing and reading temporary data files. To see timing of different steps, use verbose = 1. Note: Using GPU for training can improve training time only when the number of rows in the training data is extremely large, e.g., over 1 million. Even in such cases, GPU = 2 (hybrid mode) is recommended over GPU = 1 (force using GPU).

Usage

```
## S3 method for class 'trainpredict'
brif(
  x,
  newdata,
  type = c("score", "class"),
  n_numeric_cuts = 31,
  n_integer_cuts = 31,
  max_integer_classes = 20,
  max_depth = 20,
  min_node_size = 1,
  ntrees = 200,
  ps = 0,
  max_factor_levels = 30,
  seed = 0,
  bagging_method = 0,
  bagging_proportion = 0.9,
  vote_method = 1,
  split_search = 4,
  search_radius = 5,
  verbose = 0,
  nthreads = 2,
  CUDA = 0,
  CUDA_blocksize = 128,
  CUDA_n_lb_GPU = 20480,
  cubrif_main = "cubrif_main.exe",
  tmp_file_prefix = "cbf",
  ...
)
```

Arguments

`x` a data frame containing the training data set. The first column is taken as the target variable and all other columns are used as predictors.

`newdata` a data frame containing the new data to be predicted. All columns in x (except for the first column which is the target variable) must be present in newdata and the data types must match.

`type` a character string specifying the prediction format. Available values include "score" and "class". Default is "score".

`n_numeric_cuts` an integer value indicating the maximum number of split points to generate for each numeric variable.

`n_integer_cuts` an integer value indicating the maximum number of split points to generate for each integer variable.

`max_integer_classes` an integer value. If the target variable is integer and has more than max_integer_classes unique values in the training data, then the target variable will be grouped into max_integer_classes bins. If the target variable is numeric, then the smaller of max_integer_classes and the number of unique values number of bins will be created on the target variables and the regression problem will be solved as a classification problem.

`max_depth` an integer specifying the maximum depth of each tree. Maximum is 40.

`min_node_size` an integer specifying the minimum number of training cases a leaf node must contain.

`ntrees` an integer specifying the number of trees in the forest.

`ps` an integer indicating the number of predictors to sample at each node split. Default is 0, meaning to use sqrt(p), where p is the number of predictors in the input.

`max_factor_levels` an integer. If any factor variables has more than max_factor_levels, the program stops and prompts the user to increase the value of this parameter if the too-many-level factor is indeed intended.

`seed` an integer specifying the seed used by the internal random number generator. Default is 0, meaning not to set a seed but to accept the set seed from the calling environment.

`bagging_method` an integer indicating the bagging sampling method: 0 for sampling without replacement; 1 for sampling with replacement (bootstrapping).

`bagging_proportion`
a numeric scalar between 0 and 1, indicating the proportion of training observations to be used in each tree.

`vote_method` an integer (0 or 1) specifying the voting method in prediction. 0: each leaf contributes the raw count and an average is taken on the sum over all leaves; 1: each leaf contributes an intra-node fraction which is then averaged over all leaves with equal weight.

`split_search` an integer indicating the choice of the split search method. 0: randomly pick a split point; 1: do a local search; 2: random pick subject to regulation; 3: local search subject to regulation; 4 or above: a mix of options 0 to 3.

`search_radius` an positive integer indicating the split point search radius. This parameter takes effect only in regulated search (split_search = 2 or above).

`verbose` an integer (0 or 1) specifying the verbose level.

`nthreads` an integer specifying the number of threads used by the program. This parameter takes effect only on systems supporting OpenMP.

`CUDA` an integer (0, 1 or 2). 0: Do not use GPU. 1: Use GPU to build the forest. 2: Hybrid mode: Use GPU to split a node only when the node size is greater than CUDA_n_lb_GPU.

`CUDA_blocksize` a positive integer specifying the CUDA thread block size, must be a multiple of 64 up to 1024.

`CUDA_n_lb_GPU` a positive integer. The number of training cases must be greater than this number to enable the GPU computing when GPU = 2.

`cubrif_main` a string containing the path and name of the cubrif executable (see https://github.com/profyliu/cubrif for how to build it).

`tmp_file_prefix`
a string for the path and prefix of temporary files created when CUDA is used.

`...` additional arguments.

Value

a data frame or a vector containing the prediction results. See `predict.brif` for details.

Examples

```
trainset <- sample(1:nrow(iris), 0.5*nrow(iris))
validset <- setdiff(1:nrow(iris), trainset)

pred_score <- brif.trainpredict(iris[trainset, c(5,1:4)], iris[validset, c(1:4)], type = 'score')
pred_label <- colnames(pred_score)[apply(pred_score, 1, which.max)]
```

brifTree *Build a single brif tree of a given depth*

Description

This is a wrapper for brif to build a single tree of a given depth. See brifTree.default and brifTree.formula for details.

Usage

```
brifTree(x, ...)
```

Arguments

x	a data frame or a formula object.
...	arguments passed on to brifTree.formula or brifTree.default.

Value

an object of class brif. See brif.default for details.

Examples

```
# Build a single tree
bt <- brifTree(Species ~., data = iris, depth = 3)

# Print out the decision rules
printRules(bt)

# Get the accuracy on the training set
sum(predict(bt, newdata = iris, type = 'class') == iris[,'Species'])/nrow(iris)
```

brifTree.default *Build a single brif tree taking a data frame as input*

Description

This function invokes brif.default with appropriately set parameters to generate a single tree with the maximum expected predictive accuracy.

Usage

```
## Default S3 method:
brifTree(
  x,
  depth = 3,
  n_cuts = 2047,
  max_integer_classes = 20,
  max_factor_levels = 30,
  seed = 0,
  ...
)
```

Arguments

`x`	a data frame containing the training data. The first column is treated as the target variable.
`depth`	a positive integer indicating the desired depth of the tree.
`n_cuts`	a positive integer indicating the maximum number of split points to generate on each numeric or integer variable. A large value is preferred for a single tree.
`max_integer_classes`	a positive integer. See `brif.default` for details.
`max_factor_levels`	a positive integer. See `brif.default` for details.
`seed`	a non-negative positive integer specifying the random number generator seed.
`...`	other relevant arguments.

Value

an object of class `brif`. See `brif.default` for details.

`brifTree.formula` *Build a single brif tree taking a formula as input*

Description

Build a single brif tree taking a formula as input

Usage

```
## S3 method for class 'formula'
brifTree(
  formula,
  data,
  subset,
  na.action = stats::na.pass,
  depth = 3,
  n_cuts = 2047,
  max_integer_classes = 20,
  max_factor_levels = 30,
  seed = 0,
  ...
)
```

Arguments

`formula` an object of class "`formula`": a symbolic description of the model to be fitted.

`data` an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `brif.formula` is called.

`subset` an optional vector specifying a subset (in terms of index numbers, not actual data) of observations to be used in the fitting process.

`na.action` a function which indicates what should happen when the data contain NAs.

`depth` a positive integer indicating the desired depth of the tree.

`n_cuts` a positive integer indicating the maximum number of split points to generate on each numeric or integer variable. A large value is preferred for a single tree.

`max_integer_classes`
a positive integer. See `brif.default` for details.

`max_factor_levels`
a positive integer. See `brif.default` for details.

`seed` a non-negative positive integer specifying the random number generator seed.

`...` other relevant arguments.

Value

an object of class `brif` to be used by `predict.brif`.

`brif_write_data` *Write data set to file*

Description

Write data set to file

Usage

```
brif_write_data(df, resp_col_num = 1, outfile = "data")
```

Arguments

`df` a data frame

`resp_col_num` an integer indicating the column number (in df) of the response variable. For test data without the response column, use 0 here.

`outfile` a character string specifying the file name prefix of output files

Value

a list of four elements. n: number of rows, p: number of predictors, data_file: name of the data file, config_file: name of the configuration file

---

`predict.brif` *Make predictions using a brif model*

Description

Make predictions for `newdata` using a brif model `object`.

Usage

```
## S3 method for class 'brif'
predict(
  object,
  newdata = NULL,
  type = c("score", "class"),
  vote_method = 1,
  nthreads = 2,
  ...
)
```

Arguments

`object` an object of class "brif" as returned by the brif training function.

`newdata` a data frame. The predictor column names and data types must match those supplied for training. The order of the predictor columns does not matter though.

`type` a character string indicating the return content. For a classification problem, "score" means the by-class probabilities and "class" means the class labels (i.e., the target variable levels). For regression, the predicted values are returned.

`vote_method` an integer (0 or 1) specifying the voting method in prediction. 0: each leaf contributes the raw count and an average is taken on the sum over all leaves; 1: each leaf contributes an intra-node fraction which is then averaged over all leaves with equal weight.

`nthreads` an integer specifying the number of threads used by the program. This parameter only takes effect on systems supporting OpenMP.

`...` additional arguments.

Details

Note: If a model is built just for making predictions on one test set (i.e., no need to save the model object for future use), then the `brif.trainpredict` should be used.

Value

a data frame or a vector containing the prediction results. For regression, a numeric vector of predicted values will be returned. For classification, if `type = "class"`, a character vector of the predicted class labels will be returned; if `type = "score"`, a data frame will be returned, in which each column contains the probability of the new case being in the corresponding class.

Examples

```
# Predict using a model built by brif
pred_score <- predict(brif(Species ~ ., data = iris), iris, type = 'score')
pred_label <- predict(brif(Species ~ ., data = iris), iris, type = 'class')

# Equivalently and more efficiently:
pred_score <- brif(Species ~., data = iris, newdata = iris, type = 'score')
pred_label <- brif(Species ~., data = iris, newdata = iris, type = 'class')

# Or, retrieve predicted labels from the scores:
pred_label <- colnames(pred_score)[apply(pred_score, 1, which.max)]
```

`printBrifTree` *Print the decision rules of a Brif tree*

Description

Print the decision rules of a Brif tree

Usage

```
printBrifTree(rf, which_tree)
```

Arguments

`rf`	an object of class 'brif', as returned by rftrain.
`which_tree`	an integer indicating the tree number

Value

No return value. The function is intended for producing a side effect, which prints the decision rules to the standard output.

printRules *Print the decision rules of a brif tree*

Description

Print the decision rules of a brif tree

Usage

```
printRules(object, which_tree = 0)
```

Arguments

object	an object of class "brif" as returned by the brif training function.
which_tree	a nonnegative integer indicating the tree number (starting from 0) in the forest to be printed.

Value

No return value. The function is called for side effect. The decision rules of the given tree is printed to the console output. Users can use sink to direct the output to a file.

Examples

```
# Build a single tree
bt <- brifTree(Species ~., data = iris, depth = 3)

# Print out the decision rules
printRules(bt)

# Get the training accuracy
sum(predict(bt, newdata = iris, type = 'class') == iris[,'Species'])/nrow(iris)
```

rfpredict *Predict new cases*

Description

This function is not intended for end users. Users should use the predict.brif function instead.

Usage

```
rfpredict(rf, rdf, vote_method, nthreads)
```

Arguments

`rf` an object of class 'brif', as returned by rftrain.

`rdf` a data frame containing the new cases to be predicted.

`vote_method` an integer (0 or 1) indicating the voting mechanism among leaf predictions.

`nthreads` an integer specifying the number of threads to be used in prediction.

Value

a data frame containing the predicted values.

`rftrain` *Train a random forest*

Description

This function is not intended for end users. Users should use the brif.formula or brif.default function.

Usage

```
rftrain(rdf, par)
```

Arguments

`rdf` a data frame. The first column is treated as the target variable.

`par` a list containing all parameters.

Value

a list, of class "brif", containing the trained random forest model.

`rftrainpredict` *Train a model and predict for newdata in one go*

Description

This function is not intended for end users. Users should use the function brif or brif.trainpredict and supply the newdata argument thereof.

Usage

```
rftrainpredict(rdf, rdf_new, par)
```

Arguments

`rdf`	a data frame containing the training data.
`rdf_new`	a data frame containing new cases to be predicted.
`par`	a list containing all parameters.

Value

a data frame containing the predicted values.

`stratpar` *Stratified permutation of rows by the first column*

Description

Stratified permutation of rows by the first column

Usage

```
stratpar(x, stride)
```

Arguments

`x`	a data frame to be permuted by row
`stride`	an integer indicating how many rows are to be groups in one block

Value

a data frame, which is a permutation of x

Index

as.data.frame, *6*, *12*

brif, *2*, 2, *7*, *10*
brif-package, 2
brif.default, *2, 3*, 3, *6*, *10–12*
brif.formula, *2*, 6
brif.trainpredict, *2*, 7, *14*
brif_write_data, 12
brifTree, *2*, 10
brifTree.default, *10*, 10
brifTree.formula, *10*, 11

formula, *2*, *6*, *10*, *12*

predict.brif, *2, 3*, *5*, *7*, *9*, *12*, 13
printBrifTree, 14
printRules, *2*, *5*, 15

rfpredict, 15
rftrain, 16
rftrainpredict, 16

sink, *15*
str, *5*
stratpar, 17
summary, *5*