

Package: YARF (via r-universe)

October 16, 2026

Type Package

Title Yet Another Random Forests Package

Version 1.2.0

Description A customizable, asynchronous, and parallelized random forest implementation. Supports user-defined splitting and aggregation behavior, missingness incorporated in attributes, and random-forest-based imputation.

X-schema.org-keywords random-forest, decision-trees, ensemble-learning, machine-learning, statistical-learning, supervised-learning, classification, regression, variable-importance, interaction-detection, proximity-analysis, out-of-bag, missing-data, missing-data-imputation, custom-split-rules, asynchronous-computing, parallel-computing, Java, JavaScript, r-package

License GPL-3

URL <https://github.com/kapelner/YARF>, <https://kapelner.github.io/YARF/>

BugReports <https://github.com/kapelner/YARF/issues>

Depends R (>= 3.2.0)

Imports Bolstad2, checkmate, ggplot2, graphics, gridExtra, pander, R6, readr, rlang, rJava (>= 0.9-8), stats, stringr, YARFJARs, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

SystemRequirements Java (>= 1.8.0)

Encoding UTF-8

Config/roxygen2/version 8.0.0.9000

Config/testthat/edition 3

Config/Needs/website pkgdown

Config/pak/sysreqs make default-jdk libicu-dev libx11-dev

Repository <https://kapelner.r-universe.dev>

Date/Publication 2026-09-03 13:11:24 UTC

RemoteUrl <https://github.com/kapelner/YARF>

RemoteRef HEAD

RemoteSha 6677cdf14228609a67a334a7ace6f0792b1526c6

RemoteSubdir YARF

Contents

automobile	3
benchmark_datasets	3
calcAUC	4
compute_raw_proximity_info	4
cov_importance_test	5
first_order_interaction_investigator	6
get_tree_num_nodes_leaves_max_depths	7
illustrate_trees	8
model_fit_test	10
predict.YARF	11
prediction_nodes	11
print.YARF	12
proximity_info	13
prune_YARF_model	13
query_variable_counts	14
query_variable_proportions_across_trees	15
set_YARF_num_cores	15
shared_initial_substring	16
summary.YARF	17
tree_average_proximity_info	17
YARF	18
YARF_all_oob_results_matrix	23
YARF_convergence	24
yarf_duplicate	24
YARF_predict_all_trees	25
YARF_progress	26
YARF_serialize	26
YARF_set_aggregation_method	27
YARF_stop	28
YARF_update_with_oob_results	28
YARF_update_with_oob_test_results	29
YARF_update_with_oob_validation_results	30
YARFBAG	30
YARFCART	35
YARFMissForest	39
YARFROC	40

Index	42
--------------	-----------

automobile	<i>Automobile data</i>
------------	------------------------

Description

A 201-row, 25-variable automobile dataset distributed with YARF for examples and demonstrations.

Format

A data frame with 201 rows and 25 variables.

Source

Distributed with the YARF project.

benchmark_datasets	<i>Benchmark regression datasets</i>
--------------------	--------------------------------------

Description

Nine benchmark data frames distributed with YARF: 'ankara', 'baseball', 'boston', 'compactiv', 'ozone', 'pole', 'triazine', 'wine.red', and 'wine.white'. They are retained for reproducibility of package benchmarks.

Format

Data frames whose final column is the benchmark response. Dataset dimensions range from 186 to 8,192 rows and from 9 to 59 variables.

Source

Distributed with the YARF project.

codeAUC

Calculates the area under the curve

Description

Calculates the area under the curve

Usage

```
codeAUC(roc_results, plot = TRUE)
```

Arguments

`roc_results` The results from the function YARFROC.
`plot` Make a plot of the ROC with the AUC shaded in? Default is TRUE.

Value

The area under the curve approximated via numerical integration.

code_compute_raw_proximity_info

Computes information about the "proximity" of observations within the YARF model. Given two datasets, information is computed for all pairs of observations. Information returned is the prediction nodes (for all trees) for both objects and their common node in the tree structure plus much information about the nodes.

Description

Computes information about the "proximity" of observations within the YARF model. Given two datasets, information is computed for all pairs of observations. Information returned is the prediction nodes (for all trees) for both objects and their common node in the tree structure plus much information about the nodes.

Usage

```
code_compute_raw_proximity_info(
  yarf_mod,
  X1 = NULL,
  X2 = NULL,
  oob_only_1 = TRUE,
  oob_only_2 = TRUE,
  verbose = TRUE
)
```

Arguments

yarf_mod	A YARF model object.
X1	A $n \times p$ matrix where each row is an observation. If NULL (the default), then this parameter is set to the model's training data.
X2	A $n \times p$ matrix where each row is an observation. If NULL (the default), then this parameter is set to the model's training data.
oob_only_1	TRUE (the default) will only return proximity information for trees where each observation is out-of-bag. If X1 is not NULL, this parameter has no effect on output.
oob_only_2	TRUE (the default) will only return proximity information for trees where each observation is out-of-bag. If X2 is not NULL, this parameter has no effect on output.
verbose	Print out periodic messages on progress of the computation of the proximity information. The default is TRUE.

Value

A list indexed by the row number of X1 whose elements are a list indexed by the row number of X2 whose elements are a list indexed by tree number whose elements are a list consisting of the following information: (a) X1 prediction node (b) X2 prediction node, (c) shared node, (d) locations of all three nodes indicating left/right directions, (e) depth differences to shared node where 0 indicates they are in the same node, (f) parent node's cost for all three (where the parent cost of the root is the null model cost)

Author(s)

Adam Kapelner

cov_importance_test	<i>Tests the effect of H_0: one, or two, ... or all covariates are not predictive (out of sample) via a permutation test. We permute the covariates column(s) and build YARF models num_permutation_samples times. The p-val is determined by a permutation-like test.</i>
---------------------	---

Description

Tests the effect of H_0 : one, or two, ... or all covariates are not predictive (out of sample) via a permutation test. We permute the covariates column(s) and build YARF models num_permutation_samples times. The p-val is determined by a permutation-like test.

Usage

```
cov_importance_test(
  yarf_mod,
  covariates = NULL,
  num_permutation_samples = 100,
  plot = TRUE
)
```

Arguments

yarf_mod	The fit model to test against permuted training data.
covariates	The covariates to permute as a vector of indices or names. If multiple covariates they are permuted as a single block to not break collinearity. If this parameter is specified as NULL, we test all the covariates by permuting y instead. Default is NULL.
num_permutation_samples	How many different YARF models should be built - one for each different permutation of the variables tests. The is the resolution of the test. The default is 100.
plot	Plot a histogram of the permutation samples with the original sample displayed? Default is TRUE.

Value

A list with the following components: permutation_samples_of_error whose value is a vector with entries being the error values for each permutation sample, observed_error_estimate is the error value of the original model built with unpermuted data and pval is the estimated significance level of the permutation test.

Author(s)

Adam Kapelner

first_order_interaction_investigator

Investigate First Order Interactions

Description

This function investigates all two-way interactions in the forest by measuring how many times splits of one variable is nested in splits of another variable.

Usage

```
first_order_interaction_investigator(
  yarf_mod,
  plot = TRUE,
  only_non_self_interactions = FALSE,
  num_var_plot = 50
)
```

Arguments

yarf_mod	The YARF model in which to query
plot	This controls whether or not the function plots the interaction strengths. If so, it plots in order of strength from highest to lowest. Default is TRUE.
only_non_self_interactions	Should we only show interactions between different variables? Default is FALSE to show interactions of variables with themselves.
num_var_plot	How many first order interactions to plot? All would be p-choose-2 which can be large. The default is 50. Use Inf to plot all p-choose-2.

Value

A list with two components: (1) `interaction_counts` is a $p \times p$ sized matrix of interaction counts between the variables and (2) `sorted_counts` is a vector of the top interactions in order. Returns invisibly.

Author(s)

Adam Kapelner

```
get_tree_num_nodes_leaves_max_depths
```

Gets information about each tree: the number of nodes, number of leaves and maximum depth.

Description

Gets information about each tree: the number of nodes, number of leaves and maximum depth.

Usage

```
get_tree_num_nodes_leaves_max_depths(yarf_mod)
```

Arguments

yarf_mod	The YARF model in which to query
----------	----------------------------------

Value

A list with three integer vector components all of which have length equal to the number of trees completes: (1) `num_nodes` has entries equal to the number of nodes in each tree, (2) `num_leaves` has entries equal to the number of leaves (terminal nodes) in each tree and (3) `max_depths` has entries equal to the maximum depth of each tree.

Author(s)

Adam Kapelner

<code>illustrate_trees</code>	<i>Illustrates trees to PNG files in the local directory.</i>
-------------------------------	---

Description

Illustrates trees to PNG files in the local directory.

Usage

```
illustrate_trees(
  yarf_mod,
  trees = NULL,
  max_depth = -1,
  background_rgb_color = c(255, 255, 255),
  line_rgb_color = c(100, 100, 100),
  text_rgb_color = c(0, 0, 0),
  font_family = "Arial",
  font_size = 12,
  margin_in_px = 70,
  character_width_in_px = 4.2,
  length_in_px_per_half_split = 20,
  depth_in_px_per_split = 100,
  print_at_split_node_script = NULL,
  print_at_leaf_script = NULL,
  file_format = "png",
  use_real_names = TRUE,
  title = "yarf_mod_tree",
  open_file = FALSE
)
```

Arguments

<code>yarf_mod</code>	The YARF model whose trees are to be illustrated
<code>trees</code>	A vector of the tree indices (a subset of 1... <code>num_trees</code>)
<code>max_depth</code>	Illustrate the tree only up to this depth. Default is -1 indicating no restriction.

background_rgb_color	A vector with 3 elements in 0, 1, ..., 255 specifying the RGB color of the background. Default is c(255, 255, 255) for white.
line_rgb_color	A vector with 3 elements in 0, 1, ..., 255 specifying the RGB color of the lines. Default is c(100, 100, 100) for grey.
text_rgb_color	A vector with 3 elements in 0, 1, ..., 255 specifying the RGB color of the text. Default is c(0, 0, 0) for black.
font_family	The font family of the text in the illustration. This string is passed in as the first argument in a new java.awt.Font object. You can find a list of legal arguments in the Java 8 documentation. Default is arial.
font_size	The font size in the illustrations in pixels. Default is 12.
margin_in_px	The margin in pixels of the entire illustration (equal on all sides). Default is 70.
character_width_in_px	Estimate as to the size of the character width in pixels. Differs based on fonts.
length_in_px_per_half_split	The length of half of a split in pixels. Default is 20.
depth_in_px_per_split	The length of the depth of a split in pixels. Default is 100.
print_at_split_node_script	A custom javascript function which creates a special custom message to be printed at split nodes (see below). The default is NULL which prints no special message. <pre>function printAtSplitNode(node){ //node is of type YARF.YARFNode ... return message; //a string }</pre>
print_at_leaf_script	A custom javascript function which creates a special custom message to be printed at leaf nodes (see below). The default is NULL which prints no special message. <pre>function printAtLeaf(node){ //node is of type YARF.YARFNode ... return message; //a string }</pre>
file_format	The image format of the resulting illustration. Legal values are "jpg", "gif", "png". Default is "png".
use_real_names	In the illustrations of split rules, set this to TRUE to use the names in the data frame. and FALSE to use "X_1", "X_2", etc. Default is TRUE.
title	The name of the file. Note that "_00t.png" will be added where "t" is the tree index and "00" is an appropriate number of leading zeroes. Default is "yarf_mod_tree".
open_file	Should the first file be opened by the operating system? Default is FALSE.

Author(s)

Adam Kapelner

model_fit_test	<i>Convenience method for testing a model's fit. The strategy is simple: one builds a YARF model from the residuals of the model you wish to fit and performs an omnibus test of all covariates against those residuals. This effecticely answers the question: "is there out-of-sample predictive information left over after this model was fit?" The p-val is determined by a permutation-like test.</i>
----------------	---

Description

Convenience method for testing a model's fit. The strategy is simple: one builds a YARF model from the residuals of the model you wish to fit and performs an omnibus test of all covariates against those residuals. This effecticely answers the question: "is there out-of-sample predictive information left over after this model was fit?" The p-val is determined by a permutation-like test.

Usage

```
model_fit_test(
  X,
  y,
  model_fit = NULL,
  num_permutation_samples = 100,
  plot = TRUE,
  ...
)
```

Arguments

X	The training data frame
y	The vector of training responses
model_fit	An already-fit model that you wish to test if there is any predictive power left over. This model's class must respond to the "predict" S3 method. The default is NULL which indicates the standard linear model (missingness not allowed there).
num_permutation_samples	The resolution of the test. See cov_importance_test . The default is 100.
plot	Plot the result as a histogram. Default is TRUE.
...	Additional parameters to be passed to YARF .

Value

The same as [cov_importance_test](#).

Author(s)

Adam Kapelner

predict.YARF	<i>Computes a response prediction for new data observation(s).</i>
--------------	--

Description

Predictions are an aggregation from each of the individual trees' predictions based on the function passed into the model creation step. The default aggregation is the sample average for regression and the modal class for classification. You can set a custom aggregation method via the YARF_set_aggregation_method function.

Usage

```
## S3 method for class 'YARF'
predict(object, new_data, ...)
```

Arguments

object	A YARF model object.
new_data	A n* x p matrix where each row is an observation you wish to predict for. Here, p is the same length as the original training data.
...	Other methods passed to predict

Value

The yhat prediction for the new_data record(s).

Author(s)

Kapelner

prediction_nodes	<i>Returns the leaf nodes that are responsible for predictions for a given dataset.</i>
------------------	---

Description

Returns the leaf nodes that are responsible for predictions for a given dataset.

Usage

```
prediction_nodes(yarf_mod, X = NULL, oob_only = TRUE)
```

Arguments

yarf_mod	A YARF model object.
X	A $n \times p$ data frame where each row is an observation you wish to obtain prediction information. Here, p is the same length as the original training data and the column names match the original training data. If a matrix object is passed in, it is automatically cast to a data frame. This may not always give you what you want. If NULL (the default), then this parameter is set to the model's training data.
oob_only	If X is NULL, setting this to TRUE (the default) will only return node information for trees where each observation is out-of-bag. If X is not NULL, this parameter has no effect on output.

Value

A list where values are rJava YARFNode objects for all the prediction nodes. The first key indexes the row number in X and the second, the tree number.

Author(s)

Kapelner

print.YARF	<i>Alias for summary.</i>
------------	---------------------------

Description

Alias for summary.

Usage

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

Arguments

x	A YARF model object.
...	Other methods passed to print

Author(s)

Adam Kapelner

proximity_info	<i>Computes information about the "proximity" of observations within the YARF model. Given two datasets, information is computed for all pairs of observation.</i>
----------------	--

Description

Computes information about the "proximity" of observations within the YARF model. Given two datasets, information is computed for all pairs of observation.

Usage

```
proximity_info(yarf_mod, X, prox_single_node_calc_script = NULL)
```

Arguments

yarf_mod	A YARF model object.
X	A $n \times p$ data frame where each row is an observation. If NULL (the default), then this parameter is set to the model's training data.
prox_single_node_calc_script	Script ...

Value

(For now, a list of raw information for rows in each dataset)

Author(s)

Matt Olson

prune_YARF_model	<i>Prunes the trees in a YARF mode. This is an irreversible modification to the tree structure.</i>
------------------	---

Description

Prunes the trees in a YARF mode. This is an irreversible modification to the tree structure.

Usage

```
prune_YARF_model(yarf_mod, prune_if_script, trees_to_prune = NULL)
```

Arguments

yarf_mod	The YARF model whose trees are to be pruned.
prune_if_script	A custom javascript function (see below) which determines the pruning of a node (i.e. deletes the node's children and sets the node to a leaf and then sets a $\hat{y}$ using the assign function). <pre>function pruneIf(node){ //node is of type YARF.YARFNode ... return boolean //true if this node is to be pruned }</pre>
trees_to_prune	An array of indices which specify which of the trees to prune. The default is NULL which means all of them.

Author(s)

Adam Kapelner

query_variable_counts *Function which queries the YARF model about how many times the variables are used when splitting*

Description

Function which queries the YARF model about how many times the variables are used when splitting

Usage

```
query_variable_counts(yarf_mod, type = "splits")
```

Arguments

yarf_mod	The model in which to query
type	Either "trees" which counts the number of trees where each variable was used one (or more) times or "splits" which counts the number of times each variable was used across the whole model.

Value

A vector with length p , the number of variables considered in this YARF model, with either a binary value for type = "trees" or a count value for type = "splits".

Author(s)

Adam Kapelner

```
query_variable_proportions_across_trees
```

Function which queries the YARF model about the proportion of times the variables are used when splitting. This is a convenience method which merely normalizes the result of query_variable_counts.

Description

Function which queries the YARF model about the proportion of times the variables are used when splitting. This is a convenience method which merely normalizes the result of query_variable_counts.

Usage

```
query_variable_proportions_across_trees(yarf_mod, type = "splits")
```

Arguments

yarf_mod	The model in which to query
type	Either "trees" which computes the proportion of trees where each variable was used one (or more) times or "splits" which computes the proportion of times each variable was used across the whole model.

Value

A vector with length p, the number of variables considered in this YARF model, with values between 0 and 1.

Author(s)

Adam Kapelner

set_YARF_num_cores	<i>Sets the number of cores for YARF to use for all operations which are multithreaded (e.g. model construction, prediction, etc)</i>
--------------------	---

Description

Sets the number of cores for YARF to use for all operations which are multithreaded (e.g. model construction, prediction, etc)

Usage

```
set_YARF_num_cores(num_cores)
```

Arguments

num_cores The number of cores to use

Author(s)

Adam Kapelner

shared_initial_substring

Returns the shared initial portion of a string

Description

Given two strings, return the identical portion beginning from the first character and on.

Usage

```
shared_initial_substring(s1, s2)
```

Arguments

s1 The first string

s2 The second string

Value

The shared substring

Author(s)

Adam Kapelner

Examples

```
shared_initial_substring("money can't buy me love", "money can buy me love")  
#> "money can"
```

`summary.YARF`*Prints a summary of the model to the console*

Description

Prints a summary of the model to the console

Usage

```
## S3 method for class 'YARF'  
summary(object, ...)
```

Arguments

<code>object</code>	A YARF model object.
<code>...</code>	Other methods passed to summary

Author(s)

Adam Kapelner

`tree_average_proximity_info`*Computes information about the "proximity" of observations within the YARF model. Given two datasets, information is computed for all pairs of observations. Information is averaged over the trees in the model.*

Description

Computes information about the "proximity" of observations within the YARF model. Given two datasets, information is computed for all pairs of observations. Information is averaged over the trees in the model.

Usage

```
tree_average_proximity_info(raw)
```

Arguments

<code>raw</code>	The information returned from <code>compute_raw_proximity_info()</code> .
------------------	---

Value

A list indexed by the row number of X1 whose elements are a list indexed by the row number of X2 whose elements are an array consisting of the following information: (a) proportion of trees these two observations were in the same node, (b) average node 1/2/shared depth (c) average node 1/2/shared cost (d) average node 1/2/shared parent cost (e) average node 1/2/shared size (f) average depth difference between node 1/2 and shared

Author(s)

Kapelner

YARF

Builds a YARF Model. There are many customizations available.

Description

Builds a YARF Model. There are many customizations available.

Usage

```
YARF(
  X = NULL,
  y = NULL,
  Xy = NULL,
  Xother = NULL,
  allow_missingness_in_y = FALSE,
  num_trees = NULL,
  bootstrap_indices = NULL,
  n_max_per_tree = NULL,
  other_indices = NULL,
  mtry = NULL,
  nodesize = NULL,
  mtry_script = NULL,
  split_vals_script = NULL,
  make_node_to_leaf_script = NULL,
  cost_single_node_calc_script = NULL,
  cost_both_children_calc_script = NULL,
  node_assign_script = NULL,
  after_node_birth_function_script = NULL,
  shared_scripts = NULL,
  use_missing_data = TRUE,
  replace_missing_data_with_x_j_bar = FALSE,
  no_missing_data_split_rule = "RANDOM",
  serialize = FALSE,
  seed = NULL,
  wait = TRUE,
  calculate_oob_error = TRUE,
```

```

    fit_until_convergence = FALSE,
    oob_cost_calculation_script = NULL,
    tolerance = 0.01,
    verbose = TRUE,
    debug_log = FALSE
)

```

Arguments

<code>x</code>	The data frame of training data
<code>y</code>	The vector of training responses which is either numeric (for regression or factor (for classification)).
<code>xy</code>	The data frame of training data where the last column is responses
<code>xother</code>	Other data that is used in the training but the RF doesn't split on it
<code>allow_missingness_in_y</code>	If TRUE, missingness in the response variable, <code>y</code> , is allowed. If the missings are not handled (somehow) in the custom functions, YARF will crash!! Default is FALSE.
<code>num_trees</code>	The number of trees in the RF. Default is NULL which sets the value to 500 if <code>fit_until_convergence</code> is set to FALSE and 10,000 if <code>fit_until_convergence</code> is set to TRUE. If the model is fit asynchronously via the <code>wait</code> parameter being set to TRUE, this number represents a vague contract between the user and the software as the model fitting can be (a) halted by the user or the model can be (b) set to converge automatically (by setting the <code>fit_until_convergence</code> parameter to TRUE). In the latter case, this parameter functions as the maximum number of trees that can be created and thus, it is recommended that this number is very large, hence the default of 10,000.
<code>bootstrap_indices</code>	A list with keys 1, 2, ..., <code>num_trees</code> where each value is the indices of the training data you wish to use for each tree. The default is NULL indicating the default algorithm of sampling 1, ..., <code>n</code> with replacement (i.e. the non-parametric bootstrap default). Needless to say indices specified here will not be part of the out-of-bag collection of indices. You may specify more than enough (i.e. more than <code>num_trees</code>) elements in this list. Only the first <code>num_trees</code> will be used. No warning message will be displayed if you include too many. If this parameter is specified, there is no need to specify <code>n_max_per_tree</code>.
<code>n_max_per_tree</code>	An upper limit on the number of observations used to build each tree. If this parameter is specified, there is no need to specify <code>bootstrap_indices</code> . Default is NULL for the entire dataset, <code>n</code> .
<code>other_indices</code>	An optional list with keys 1,2,..., <code>num_trees</code> where each value is indices of the training data you wish to use in some custom way for each tree using a custom function. If not custom function is specified which makes use of this, it will be ignored. However, indices specified here will not be part of the out-of-bag collection of indices. If this is not your wish, the elements in the vectors specified in this list's values should be a subset of those in the values of

	bootstrap_indices. The default is NULL indicating you do not wish to specify any "other" data records.
mtry	The number of variables tried at every split. The default is NULL which indicates the out-of-box RF default which is $\text{floor}(p / 3)$ for regression and $\text{floor}(\sqrt{p})$ for classification. If you want to use all possible features, set this parameter to "all". If you want a custom function, leave this NULL and see next parameter.
nodesize	The minimum number of observations in a node. YARF will stop splitting at this point. If NULL the out-of-the-box default of 5 for regression and 1 for classification will be used.
mtry_script	A custom javascript function which selects the variables to be greedily searched (see below) The default is NULL which employs the mtry argument. If you specify your function please randomize the order of the returned attributes to arbitrate ties. <pre>function tryVars(node){ //node is of type YARF.YARFNode ... return int_array //a subset of 0,...,p-1, indices indicating the variables to perform the exhaustive search on }</pre>
split_vals_script	A custom javascript function which selects the split values to be greedily searched in feature j. The default is NULL which employs the midpoints of all sorted values. <pre>function tryVals(node, j){ //node is of type YARF.YARFNode and j is the feature number in 0,...,p-1 ... return double_array //a vector of split vals to greedily assess }</pre>
make_node_to_leaf_script	A custom javascript function to be used to calculate nodesize (see below). The default is NULL where nodesize will be calculated as a static constant (see the nodesize argument). <pre>function makeNodeIntoLeaf(node){ //node is of type YARF.YARFNode ... return boolean //where true makes this node into a leaf }</pre>
cost_single_node_calc_script	A custom cost calculation for a potential node (when considering a split) in Javascript (see below). The default is NULL which means the out-of-the-box default of sum of squared error relative to the sample average (if regression) and sum of entropy (if classification). You may find it convenient to also make a node assignment here. If so, make sure you specify the node assignment function as a blank function (not NULL). <pre>function nodeCost(node){ //node is of type YARF.YARFNode ... return double //where a higher number indicates a higher cost }</pre>

cost_both_children_calc_script

A custom cost calculation in Javascript for an entire split considering both the putative left and right children nodes (see below). The default is NULL which means the out-of-the-box default for Random Forests which is sum of left and right nodes' costs for regression and average of left and right nodes' cost (relative to the number of observations in each node).

```
function totalChildrenCost(leftNode, rightNode){ //both nodes are of type YARF.YARFNode
...
return double //where a higher number indicates a higher cost
}
```

node_assign_script

A custom node assignment function in Javascript (see below). This function is run after RF greedily finds the "lowest cost" split. The default is NULL corresponding to the sample average of the node responses in regression or the modal class during classification.

```
function assignYhatToNode(node){ //node is of type YARF.YARFNode
...
return double //assigned as this node's predicted value ("y_hat")
}
```

after_node_birth_function_script

A custom function in Javascript which is executed after a node is given birth to. The default is NULL which implies nothing special is done, the Random Forest default. This is particularly useful to record extra information in the node (e.g. by writing a hash to the other_info field in the node).

```
function nodeAfterNodeBirth(node){ //node is of type YARF.YARFNode
...
}
```

shared_scripts Custom Javascript code that are always in scope when running all your custom methods. The default is NULL for no shared scripts.

use_missing_data

Use the "missing-incorporated-in-attributes" strategy to fit data with missingness. The default is TRUE.

replace_missing_data_with_x_j_bar

Replace missing predictor values with their training-column means before fitting.

no_missing_data_split_rule

What rule should be used for missing data seen in testing for nodes that did not see missing data in training? Choices: "RANDOM" and "CONDITIONAL_ON_QUANTILE". "RANDOM" will assign a split rule for missing data at such nodes purely arbitrarily and with equal probability, i.e. 50 will condition this probability on the quantile of the split value amongst the observed (non-missing) data. For example, if the split value lies at the 75th percentile, then with probability .75, this node's rule for missing data will be to go left, and with probability .25 to go right. Note that for either of these options, the randomization happens at training time and thus during prediction the rule will be deterministic. The default is "RANDOM".

<code>serialize</code>	Should the YARF model be saved? The default is FALSE as this is costly in processing time and memory. This can only be set to TRUE if <code>wait = TRUE</code> . If TRUE, we will automatically serialize after other operations that add data (such as the OOB evaluation).
<code>seed</code>	Set a random seed for reproducibility.
<code>wait</code>	Should we hang R to wait for the YARF model to complete? The default is TRUE.
<code>calculate_oob_error</code>	Should we also calculate the OOB error? Default is TRUE. Automatically is turned off if <code>wait</code> becomes FALSE.
<code>fit_until_convergence</code>	Default is FALSE. If TRUE, then the <code>wait</code> parameter is set to FALSE regardless of the user-specified value. Then, the Random Forest model is fit until "convergence" as defined below: After each tree is completed, the OOB cost is computed. If the cost decreases the algorithm does not engage. However, upon the first increase in cost, the model checks for convergence by calculating the average change in cost and the standard deviation of change in cost. If a 95% confidence interval of the average cost is within the window created by plus or minus tolerance (i.e. from zero), then the algorithm has "converged". Upon convergence, the model is stopped and a message is printed to the console. Note that specification of the <code>oob_cost_calculation_script</code> and a proper tolerance level is essential. It is strongly recommended to set the <code>num_trees</code> parameter large, otherwise the model can halt before it has converged since <code>num_trees</code> represents an upper limit. If left unspecified, <code>num_trees</code> is set to 10,000 when <code>fit_until_convergence</code> set to TRUE. If the user wishes to view the convergence in real time, we recommend the YARF_progress function. However, this function will lock the console.
<code>oob_cost_calculation_script</code>	This parameter will determine the out-of-bag cost of the RF forest model. Default is NULL. If it is NULL, then the cost will be $1-R^2$ for regression models and misclassification error for classification models. If non-null, an optional custom Javascript function which calculates the cost of a prediction given the true value of the prediction (see below). It is likely similar to <code>cost_single_node_calc_script</code>. It is recommended to share code between them by writing a function included in the <code>shared_scripts</code> argument which can be referenced when calculating OOB results. <pre>function oobCost(y_hat, y){ //y_hat is the predicted value and y is the true value (both are of type double) ... return double //where a larger number indicates a higher cost to the error between y and y_hat. } </pre> If this script is specified (i.e. the argument is non-NULL), this will also affect the output of the YARF_update_with_oob_results function. This script can be reset after the model is built if necessary.

tolerance	This parameter is ignored unless <code>fit_until_convergence</code> is set to <code>TRUE</code> . If so, this controls the tolerance used when assessing the convergence of the RF model. Default is <code>0.1</code> . If <code>oob_cost_calculation_script</code> is specified, this should also be specified as <code>0.1</code> may not be appropriate for a custom oob cost function.
verbose	Should we print out messages verbosely during construction? Default is <code>FALSE</code> .
debug_log	Should we print out messages from Java? Default is <code>FALSE</code> .

Value

A list that (a) reiterates all of the above arguments passed in and (b) `pred_type` is the guess as to modeling type: regression or classification (c) `java_YARF` is the Java object (d) `y_levels` is the unique values of the response variable (e) `model_matrix_training_data` is the data matrix with factor variables converted to dummies that is fed into YARF as the X variables (f) `training_data_features` is the names of the features (g) `predictors_which_are_factors` is a list of the variables that are factors and (h) `n` and `p` which are the dimensions of the training data model matrix and (i) various other convenient data and internal data of use to other functions in this package.

Author(s)

Adam Kapelner

YARF_all_oob_results_matrix

All OOB Results Matrix

Description

Computes the out-of-bag (OOB) predictions for all observations in training data by tree, returning a matrix. This is useful for debugging or other analyses

Usage

```
YARF_all_oob_results_matrix(yarf_mod)
```

Arguments

<code>yarf_mod</code>	The <code>yarf</code> model object such that <code>YARF_update_with_oob_validation_results</code> has been run before on a user-determined proportion of OOB samples as data validation.
-----------------------	--

Author(s)

Adam Kapelner

YARF_convergence	<i>Plot YARF convergence</i>
------------------	------------------------------

Description

Prints out a messages reflecting the progress of the YARF model construction until completion.

Usage

```
YARF_convergence(yarf_mod, time_delay_in_seconds = 5, trail_pts = 50)
```

Arguments

yarf_mod	The yarf model object
time_delay_in_seconds	Frequency of messages in seconds. Default is 5 seconds. Minimum is 0.1. NULL indicates it is shown once and then it completes.
trail_pts	If non-null, it will plot a secondary window with only the trees in the last trail_trees number of progress updates defined by time_delay_in_seconds versus error which allows assessing the convergence more closely. Default is 50.

Details

If time_delay_in_seconds is not NULL, running this function once will slow down your model construction since it will forever be calculating OOB statistics after each tree (until completion or until the user stops the function).

Author(s)

Adam Kapelner

yarf_duplicate	<i>Duplicates the settings of YARF model. Wait is TRUE, all verbose logging is turned off and serializing is turned off as well.</i>
----------------	--

Description

Duplicates the settings of YARF model. Wait is TRUE, all verbose logging is turned off and serializing is turned off as well.

Usage

```
yarf_duplicate(yarf_mod, X = NULL, y = NULL)
```

Arguments

yarf_mod	The YARF model with settings you wish to duplicate
X	The training data as a data frame. Default is NULL indicating the original training data.
y	The training responses as numeric / integer / factor vector. Default is NULL indicating the original training responses.

Value

A newly fitted YARF model with the original settings and the supplied training data.

Author(s)

Adam Kapelner

YARF_predict_all_trees

Computes a response prediction for new data observation(s) for all trees (unaggregated).

Description

Computes a response prediction for new data observation(s) for all trees (unaggregated).

Usage

```
YARF_predict_all_trees(yarf_mod, new_data)
```

Arguments

yarf_mod	A YARF model object.
new_data	A n* x p data frame where each row is an observation you wish to predict for. Here, p is the same length as the original training data and new_data has the same column names. If a matrix object is passed in, it is automatically cast to a data frame. This may not always give you what you want.

Value

A matrix of yhat predictions for the new_data record(s) for all trees (size n* x num_trees).

Author(s)

Adam Kapelner

YARF_progress	<i>Prints out a message reflecting the progress of the YARF model construction</i>
---------------	--

Description

Prints out a message reflecting the progress of the YARF model construction

Usage

```
YARF_progress(yarf_mod, console_message = TRUE)
```

Arguments

yarf_mod	The yarf model object
console_message	Should we print a message to console? Default is TRUE.

Value

The yarf object with new progress data

Author(s)

Adam Kapelner

YARF_serialize	<i>Serializes the model so the user can use save and save.image to write it to a file that can be then loaded into another and/or future R session.</i>
----------------	---

Description

Serializes the model so the user can use save and save.image to write it to a file that can be then loaded into another and/or future R session.

Usage

```
YARF_serialize(yarf_mod)
```

Arguments

yarf_mod	The yarf model object
----------	-----------------------

Author(s)

Adam Kapelner

YARF_set_aggregation_method*Sets the Tree Aggregation Method*

Description

This function sets custom code to be run when the Forest is making a decision from its many trees. This function should be run before you run `predict` or `YARF_update_with_oob_results` so that your custom aggregation can be employed. This function is optional as the default aggregation method corresponds to (a) the sample average for regression and (b) the modal category for classification. One further note of warning: once this script has been set, it will be retained in the YARF model until this function is run again for this YARF model object with a new aggregation script. Set `aggregation_script = NULL` if you wish to reset.

Usage

```
YARF_set_aggregation_method(yarf_mod, aggregation_script)
```

Arguments

`yarf_mod` The yarf model object

`aggregation_script`

A custom javascript function which aggregates the predictions in the trees for one observations into one scalar prediction (see below).

```
function aggregateYhatsIntoOneYhat(y_hats, yarf){ //y_hats is an array of doubles of size num_trees //and yarf provides access to the entire random forest object if needed (of type YARF.YARF)
```

```
...
```

```
return double //this is the final predicted value aggregated from all tree predictions
```

```
}
```

Value

The yarf model object (invisibly)

Author(s)

Adam Kapelner

YARF_stop	<i>Halts the model building.</i>
-----------	----------------------------------

Description

Halts the model building.

Usage

```
YARF_stop(yarf_mod)
```

Arguments

yarf_mod	The yarf model object
----------	-----------------------

Author(s)

Adam Kapelner

YARF_update_with_oob_results	<i>Computes the out-of-bag (OOB) predictions for the training data. This gives a good sense of out-of-sample performance in the future.</i>
------------------------------	---

Description

Computes the out-of-bag (OOB) predictions for the training data. This gives a good sense of out-of-sample performance in the future.

Usage

```
YARF_update_with_oob_results(  
  yarf_mod,  
  oob_cost_calculation_script = NULL,  
  indices = NULL  
)
```

Arguments

yarf_mod	The yarf model object
oob_cost_calculation_script	An optional custom Javascript function which calculates the cost of a prediction given the true value of the prediction (see below). It is likely similar to <code>cost_single_node_calc_script</code> . It is recommended to share code between them by writing a function included in <code>shared_scripts</code> which is passed into YARF upon construction and can be referenced when calculating OOB results.

```

function oobCost(y_hat, y){ //y_hat is the predicted value and y is the true value
  (both are of type double)
  ...
  return double //where a larger number indicates a higher cost to the error be-
  tween y and y_hat.
}

```

indices The indices to compute OOB performance. This is generally a private argument for use by the validation / test functions.

Author(s)

Adam Kapelner

YARF_update_with_oob_test_results

Computes the out-of-bag (OOB) predictions for the test subset of the training data. This function should only be run once. If it run again you risk adapting your model to the test data which will lead to overfitting and hence a biased out-of-sample performance estimate.

Description

Computes the out-of-bag (OOB) predictions for the test subset of the training data. This function should only be run once. If it run again you risk adapting your model to the test data which will lead to overfitting and hence a biased out-of-sample performance estimate.

Usage

```
YARF_update_with_oob_test_results(yarf_mod)
```

Arguments

yarf_mod The yarf model object such that YARF_update_with_oob_validation_results has been run before on a user-determined proportion of OOB samples as data validation.

Author(s)

Adam Kapelner

```
YARF_update_with_oob_validation_results
```

Computes the out-of-bag (OOB) predictions for the validation subset of the training data. This allows the user to try many different models while still leaving test data for true out-of-sample performance estimation.

Description

Computes the out-of-bag (OOB) predictions for the validation subset of the training data. This allows the user to try many different models while still leaving test data for true out-of-sample performance estimation.

Usage

```
YARF_update_with_oob_validation_results(yarf_mod, validation_proportion = 0.8)
```

Arguments

<code>yarf_mod</code>	The yarf model object
<code>validation_proportion</code>	The percentage of the data to use for validation data (and the remaining is test data). The default is 80%.

Author(s)

Adam Kapelner

YARFBAG

A convenience method to build a Bagged Tree model via YARF.

Description

You can conceptualize this as a random forest without the feature that de-correlates the trees, namely trying random subsets of variables at each split. Here, all splits are considered. You can also think about this as model averaging many CART trees built from bootstrap samples (or specific samples you choose).

Usage

```
YARFBAG(  
  X = NULL,  
  y = NULL,  
  Xy = NULL,  
  Xother = NULL,
```

```

allow_missingness_in_y = FALSE,
num_trees = NULL,
bootstrap_indices = NULL,
n_max_per_tree = NULL,
other_indices = NULL,
nodesize = NULL,
mtry_script = NULL,
split_vals_script = NULL,
make_node_to_leaf_script = NULL,
cost_single_node_calc_script = NULL,
cost_both_children_calc_script = NULL,
node_assign_script = NULL,
after_node_birth_function_script = NULL,
shared_scripts = NULL,
use_missing_data = TRUE,
replace_missing_data_with_x_j_bar = FALSE,
serialize = FALSE,
seed = NULL,
wait = TRUE,
calculate_oob_error = TRUE,
fit_until_convergence = FALSE,
oob_cost_calculation_script = NULL,
tolerance = 0.01,
verbose = TRUE,
debug_log = FALSE
)

```

Arguments

<code>x</code>	The data frame of training data
<code>y</code>	The vector of training responses
<code>xy</code>	The data frame of training data where the last column is responses
<code>xother</code>	Other data that is used in the training but the RF doesn't split on it
<code>allow_missingness_in_y</code>	If TRUE, missingness in the response variable, <code>y</code> , is allowed. If this is not handled in the custom functions, YARF will crash. Default is FALSE.
<code>num_trees</code>	The number of trees in the RF. Default is NULL which sets the value to 500 if <code>fit_until_convergence</code> is set to FALSE and 10,000 if <code>fit_until_convergence</code> is set to TRUE. If the model is fit asynchronously via the <code>wait</code> parameter being set to TRUE, this number represents a vague contract between the user and the software as the model fitting can be (a) halted by the user or the model can be (b) set to converge automatically (by setting the <code>fit_until_convergence</code> parameter to TRUE). In the latter case, this parameter functions as the maximum number of trees that can be created and thus, it is recommended that this number is very large, hence the default of 10,000.

<code>bootstrap_indices</code>	A list with keys 1, 2, ..., <code>num_trees</code> where each value is the indices of the training data you wish to use for each tree. The default is NULL indicating the default algorithm of sampling 1, ..., <code>n</code> with replacement (i.e. the non-parametric bootstrap default). Needless to say indices specified here will not be part of the out-of-bag collection of indices. You may specify more than enough (i.e. more than <code>num_trees</code>) elements in this list. Only the first <code>num_trees</code> will be used. No warning message will be displayed if you include too many.
<code>n_max_per_tree</code>	An upper limit on the number of observations used to build each tree. If this parameter is specified, there is no need to specify <code>bootstrap_indices</code> . Default is NULL for the entire dataset, <code>n</code> .
<code>other_indices</code>	An optional list with keys 1,2,..., <code>num_trees</code> where each value is indices of the training data you wish to use in some custom way for each tree using a custom function. If not custom function is specified which makes use of this, it will be ignored. However, indices specified here will not be part of the out-of-bag collection of indices. If this is not your wish, the elements in the vectors specified in this list's values should be a subset of those in the values of <code>bootstrap_indices</code> . The default is NULL indicating you do not wish to specify any "other" data records.
<code>nodesize</code>	The minimum number of observations in a node. YARF will stop splitting at this point. If NULL the out-of-the-box default of 5 for regression and 1 for classification will be used.
<code>mtry_script</code>	A custom JavaScript function that selects candidate variables at each split.
<code>split_vals_script</code>	A custom javascript function which selects the split values to be greedily searched in feature <code>j</code>. The default is NULL which employs the midpoints of all sorted values. <pre>function tryVals(node, j){ //node is of type YARF.YARFNode and j is the feature number in 0,...,p-1 ... return double_array //a vector of split vals to greedily assess }</pre>
<code>make_node_to_leaf_script</code>	A custom javascript function to be used to calculate <code>nodesize</code> (see below). The default is NULL where <code>nodesize</code> will be calculated as a static constant (see the <code>nodesize</code> argument). <pre>function makeNodeIntoLeaf(node){ //node is of type YARF.YARFNode ... return boolean //where true makes this node into a leaf }</pre>
<code>cost_single_node_calc_script</code>	A custom cost calculation for a potential node (when considering a split) in Javascript (see below). The default is NULL which means the out-of-the-box default of sum of squared error relative to the sample average (if regression) and sum of entropy (if classification). You may find it convenient to also made a node

assignment here. If so, make sure you specify the node assignment function as a blank function (not NULL).

```
function nodeCost(node){ //node is of type YARF.YARFNode
...
return double //where a higher number indicates a higher cost
}
```

`cost_both_children_calc_script`

A custom cost calculation in Javascript for an entire split considering both the putative left and right children nodes (see below). The default is NULL which means the out-of-the-box default for Random Forests which is sum of left and right nodes' costs for regression and average of left and right nodes' cost (relative to the number of observations in each node).

```
function totalChildrenCost(leftNode, rightNode){ //both nodes are of type YARF.YARFNode
...
return double //where a higher number indicates a higher cost
}
```

`node_assign_script`

A custom node assignment function in Javascript (see below). This function is run after RF greedily finds the "lowest cost" split. The default is NULL corresponding to the sample average of the node responses in regression or the modal class during classification.

```
function assignYhatToNode(node){ //node is of type YARF.YARFNode
...
return double //assigned as this node's predicted value ("y_hat")
}
```

`after_node_birth_function_script`

A custom function in Javascript which is executed after a node is given birth to. The default is NULL which implies nothing special is done, the Random Forest default. This is particularly useful to record extra information in the node (e.g. by writing a hash to the `other_info` field in the node).

```
function nodeAfterNodeBirth(node){ //node is of type YARF.YARFNode
...
}
```

`shared_scripts` Custom Javascript code that are always in scope when running all your custom methods. The default is NULL for no shared scripts.

`use_missing_data`

Use the "missing-incorporated-in-attributes" strategy to fit data with missingness. The default is TRUE.

`replace_missing_data_with_x_j_bar`

Replace missing predictor values with their training-column means before fitting.

`serialize`

Should the YARF model be saved? The default is FALSE as this is costly in processing time and memory. This can only be set to TRUE if `wait = TRUE`. If TRUE, we will automatically serialize after other operations that add data (such as the OOB evaluation).

seed	Set a random seed for reproducibility.
wait	Should we hang R to wait for the YARF model to complete? The default is TRUE.
calculate_oob_error	Should we also calculate the OOB error? Default is TRUE. Automatically is turned off if wait becomes FALSE.
fit_until_convergence	Fit trees until the out-of-bag convergence criterion is met.
oob_cost_calculation_script	This parameter will determine the out-of-bag cost of the RF forest model. Default is NULL. If it is NULL, then the cost will be $1-R^2$ for regression models and misclassification error for classification models. If non-null, an optional custom Javascript function which calculates the cost of a prediction given the true value of the prediction (see below). It is likely similar to <code>cost_single_node_calc_script</code>. It is recommended to share code between them by writing a function included in the <code>shared_scripts</code> argument which can be referenced when calculating OOB results. <pre>function oobCost(y_hat, y){ //y_hat is the predicted value and y is the true value (both are of type double) ... return double //where a larger number indicates a higher cost to the error between y and y_hat. }</pre> If this script is specified (i.e. the argument is non-NULL), this will also affect the output of the YARF_update_with_oob_results function. This script can be reset after the model is built if necessary.
tolerance	This parameter is ignored unless <code>fit_until_convergence</code> is set to TRUE. If so, this controls the tolerance used when assessing the convergence of the RF model. Default is 0.1. If <code>oob_cost_calculation_script</code> is specified, this should also be specified as 0.1 may not be appropriate for a custom oob cost function.
verbose	Should we print out messages verbosely during construction? Default is FALSE.
debug_log	Should we print out messages from Java? Default is FALSE.

Value

A list that (a) reiterates all of the above arguments passed in and (b) `pred_type` is the guess as to modeling type: regression or classification (c) `java_YARF` is the Java object (d) `y_levels` is the unique values of the response variable (e) `model_matrix_training_data` is the data matrix with factor variables converted to dummies that is fed into YARF as the X variables (f) `training_data_features` is the names of the features (g) `predictors_which_are_factors` is a list of the variables that are factors and (h) `n` and `p` which are the dimensions of the training data model matrix and (i) various other convenient data and internal data of use to other functions in this package.

Author(s)

Adam Kapelner

YARFCART	<i>A convenience method to build a CART model via YARF. There are many customizations available.</i>
----------	--

Description

A convenience method to build a CART model via YARF. There are many customizations available.

Usage

```
YARFCART(
  X = NULL,
  y = NULL,
  Xy = NULL,
  Xother = NULL,
  allow_missingness_in_y = FALSE,
  bootstrap_indices = NULL,
  other_indices = NULL,
  mtry = NULL,
  nodesize = NULL,
  mtry_script = NULL,
  split_vals_script = NULL,
  make_node_to_leaf_script = NULL,
  cost_single_node_calc_script = NULL,
  cost_both_children_calc_script = NULL,
  node_assign_script = NULL,
  after_node_birth_function_script = NULL,
  shared_scripts = NULL,
  use_missing_data = TRUE,
  replace_missing_data_with_x_j_bar = FALSE,
  serialize = FALSE,
  seed = NULL,
  calculate_oob_error = TRUE,
  oob_cost_calculation_script = NULL,
  verbose = TRUE,
  debug_log = FALSE
)
```

Arguments

X	The data frame of training data
y	The vector of training responses
Xy	The data frame of training data where the last column is responses
Xother	Other data that is used in the training but the RF doesn't split on it
allow_missingness_in_y	If TRUE, missingness in the response variable, y, is allowed. If this is not handled in the custom functions, YARF will crash. Default is FALSE.

<code>bootstrap_indices</code>	A list with keys 1, 2, ..., <code>num_trees</code> where each value is the indices of the training data you wish to use for each tree. The default is NULL indicating the default algorithm of sampling 1, ..., <code>n</code> with replacement (i.e. the non-parametric bootstrap default). Needless to say indices specified here will not be part of the out-of-bag collection of indices. You may specify more than enough (i.e. more than <code>num_trees</code>) elements in this list. Only the first <code>num_trees</code> will be used. No warning message will be displayed if you include too many.
<code>other_indices</code>	An optional list with keys 1,2,..., <code>num_trees</code> where each value is indices of the training data you wish to use in some custom way for each tree using a custom function. If not custom function is specified which makes use of this, it will be ignored. However, indices specified here will not be part of the out-of-bag collection of indices. If this is not your wish, the elements in the vectors specified in this list's values should be a subset of those in the values of <code>bootstrap_indices</code> . The default is NULL indicating you do not wish to specify any "other" data records.
<code>mtry</code>	The number of variables tried at every split. The default is NULL which indicates the out-of-box RF default which is $\text{floor}(p / 3)$ for regression and $\text{floor}(\text{sqrt}(p))$ for classification. If you want a custom function, leave this NULL and see next parameter.
<code>nodesize</code>	The minimum number of observations in a node. YARF will stop splitting at this point. If NULL the out-of-the-box default of 5 for regression and 1 for classification will be used.
<code>mtry_script</code>	A custom javascript function which selects the variables to be greedily searched (see below) The default is NULL which employs the <code>mtry</code> argument. If you specify your function please randomize the order of the returned attributes to arbitrate ties. <pre>function tryVars(node){ //node is of type YARF.YARFNode ... return int_array //a subset of 0,...,p-1, indices indicating the variables to perform the exhaustive search on }</pre>
<code>split_vals_script</code>	A custom javascript function which selects the split values to be greedily searched in feature <code>j</code> . The default is NULL which employs the midpoints of all sorted values. <pre>function tryVals(node, j){ //node is of type YARF.YARFNode and j is the feature number in 0,...,p-1 ... return double_array //a vector of split vals to greedily assess }</pre>
<code>make_node_to_leaf_script</code>	A custom javascript function to be used to calculate <code>nodesize</code> (see below). The default is NULL where <code>nodesize</code> will be calculated as a static constant (see the <code>nodesize</code> argument). <pre>function makeNodeIntoLeaf(node){ //node is of type YARF.YARFNode</pre>

```

...
return boolean //where true makes this node into a leaf
}
cost_single_node_calc_script
A custom cost calculation for a potential node (when considering a split) in
Javascript (see below). The default is NULL which means the out-of-the-box
default of sum of squared error relative to the sample average (if regression) and
sum of entropy (if classification). You may find it convenient to also made a node
assignment here. If so, make sure you specify the node assignment function as
a blank function (not NULL).
function nodeCost(node){ //node is of type YARF.YARFNode
...
return double //where a higher number indicates a higher cost
}
cost_both_children_calc_script
A custom cost calculation in Javascript for an entire split considering both the
putative left and right children nodes (see below). The default is NULL which
means the out-of-the-box default for Random Forests which is sum of left and
right nodes' costs for regression and average of left and right nodes' cost (rela-
tive to the number of observations in each node).
function totalChildrenCost(leftNode, rightNode){ //both nodes are of type YARF.YARFNode
...
return double //where a higher number indicates a higher cost
}
node_assign_script
A custom node assignment function in Javascript (see below). This function is
run after RF greedily finds the "lowest cost" split. The default is NULL corre-
sponding to the sample average of the node responses in regression or the modal
class during classification.
function assignYhatToNode(node){ //node is of type YARF.YARFNode
...
return double //assigned as this node's predicted value ("y_hat")
}
after_node_birth_function_script
A custom function in Javascript which is executed after a node is given birth to.
The default is NULL which implies nothing special is done, the Random Forest
default. This is particularly useful to record extra information in the node (e.g.
by writing a hash to the other_info field in the node).
function nodeAfterNodeBirth(node){ //node is of type YARF.YARFNode
...
}
shared_scripts Custom Javascript code that are always in scope when running all your custom
methods. The default is NULL for no shared scripts.
use_missing_data
Use the "missing-incorporated-in-attributes" strategy to fit data with missing-
ness. The default is TRUE.

```

<code>replace_missing_data_with_x_j_bar</code>	Replace missing predictor values with their training-column means before fitting.
<code>serialize</code>	Should the YARF model be saved? The default is FALSE as this is costly in processing time and memory. This can only be set to TRUE if <code>wait = TRUE</code> . If TRUE, we will automatically serialize after other operations that add data (such as the OOB evaluation).
<code>seed</code>	Set a random seed for reproducibility.
<code>calculate_oob_error</code>	Should we also calculate the OOB error? Default is TRUE. Automatically is turned off if <code>wait</code> becomes FALSE.
<code>oob_cost_calculation_script</code>	This parameter will determine the out-of-bag cost of the RF forest model. Default is NULL. If it is NULL, then the cost will be $1-R^2$ for regression models and misclassification error for classification models. If non-null, an optional custom Javascript function which calculates the cost of a prediction given the true value of the prediction (see below). It is likely similar to <code>cost_single_node_calc_script</code>. It is recommended to share code between them by writing a function included in the <code>shared_scripts</code> argument which can be referenced when calculating OOB results. <pre>function oobCost(y_hat, y){ //y_hat is the predicted value and y is the true value (both are of type double) ... return double //where a larger number indicates a higher cost to the error between y and y_hat. }</pre> If this script is specified (i.e. the argument is non-NULL), this will also affect the output of the YARF_update_with_oob_results function. This script can be reset after the model is built if necessary.
<code>verbose</code>	Should we print out messages verbosely during construction? Default is FALSE.
<code>debug_log</code>	Should we print out messages from Java? Default is FALSE.

Value

A list that (a) reiterates all of the above arguments passed in and (b) `pred_type` is the guess as to modeling type: regression or classification (c) `java_YARF` is the Java object (d) `y_levels` is the unique values of the response variable (e) `model_matrix_training_data` is the data matrix with factor variables converted to dummies that is fed into YARF as the X variables (f) `training_data_features` is the names of the features (g) `predictors_which_are_factors` is a list of the variables that are factors and (h) `n` and `p` which are the dimensions of the training data model matrix and (i) various other convenient data and internal data of use to other functions in this package.

Author(s)

Adam Kapelner

YARFMissForest	<i>MissForest Imputation</i>
----------------	------------------------------

Description

This function imputes missing data using the missForest algorithm which essentially iteratively runs the RF algorithm on each column with missingness until the predictions do not change significantly. We provide a means to run this imputation on both training data (with responses) and test data (without responses). The user can also enter algorithm hyperparameters like convergence settings.

Usage

```
YARFMissForest(
  Xtrain,
  ytrain,
  Xtest = NULL,
  maxiter = 10,
  converge_after_increasing = TRUE,
  verbose_missForest = TRUE,
  ...
)
```

Arguments

Xtrain	The training data as a data frame.
ytrain	The training responses as numeric / integer / factor vector.
Xtest	The test data as a data frame (optional, default NULL for not specified).
maxiter	The maximum number of iterations before stopping (default is 10, as in package missForest).
converge_after_increasing	The algorithm quits after imputation errors increase once. Default is TRUE but if not, the algorithm continues all <code>maxiter</code> runs and then returns the final imputations.
verbose_missForest	Prints out messages by iteration. Default is TRUE.
...	Settings to pass to YARF during the iterations.

Details

After each iteration the difference between the previous and the new imputed data matrix is assessed for the continuous and categorical parts. The stopping criterion is defined such that the imputation process is stopped as soon as both differences have become larger once. In case of only one type of variable the computation stops as soon as the corresponding difference goes up for the first time. However, the imputation last performed where both differences went up is generally less accurate than the previous one. Therefore, whenever the computation stops due to the stopping criterion (and not due to 'maxiter') the before last imputation matrix is returned.

Value

A list with objects: `Xtrain_imp` (and `Xtest_imp` if `Xtest` was specified) as well as `nrmse`s which is a record of the sum of the normalized rmse's over all numeric features by iteration and `npfc`s which is a record of the sum of the misclassification rates over all categorical features by iteration.

Author(s)

Adam Kapelner

YARFROC

ROC and ROC-like curves

Description

This generates data with a plot of an ROC curve or an ROC like curve for binary response

Usage

```
YARFROC(
  X,
  y,
  x_axis = "FPR",
  y_axis = "TPR",
  use_prop_data = 1,
  minimum_class_proportion = 0.02,
  desired_interval = 0.05,
  tolerance = 0.001,
  y_axis_fine_resolution = FALSE,
  plot = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

<code>X</code>	The covariate data frame
<code>y</code>	The binary response vector
<code>x_axis</code>	The rate to plot on the x-axis of the ROC like plot as a 3-character case-insensitive string. The default is "FPR" but it can be any of the following: "tpr" - true positive rate (AKA "sensitivity" or "recall") i.e. the number predicted positive among those truly positive "fnr" - false negative rate (AKA "miss rate") i.e. the number predicted negative among those truly positive "fpr" - false positive rate (AKA "fall-out") i.e. the number predicted positive among those truly negative "tnr" - true negative rate (AKA "specificity") i.e. the number predicted negative among those truly negative "ppv" - positive predictive value i.e. the number truly positive among those predicted positive "fdr" - false discovery rate

	i.e. the number truly negative among those predicted positive "fomr" - false omission rate i.e. the number truly positive among those predicted negative "npv" - negative predictive value i.e. the number truly negative among those predicted negative
y_axis	The rate to plot on the y-axis of the ROC like plot as a 3-character string. The default is "TPR" but it can be any of those listed in the documentation for the x_axis parameter. Must be different from x-axis.
use_prop_data	Relative multiplier controlling the resampled class proportions.
minimum_class_proportion	Smallest positive-class proportion considered while tracing the curve.
desired_interval	Target spacing between adjacent metric values.
tolerance	Recursion stops when candidate class proportions differ by less than this value.
y_axis_fine_resolution	Whether to make an additional pass emphasizing resolution on the y-axis.
plot	Whether to draw intermediate and final ROC-like plots.
verbose	Whether to print progress information.
...	Additional arguments passed to YARF construction. This cannot include bootstrap_indicies.

Value

A data frame containing all models with the percent of first class in `levels(y)` for the bootstrap indices to replicate any point on the plot.

Author(s)

Adam

Index

ankara (benchmark_datasets), 3
automobile, 3

baseball (benchmark_datasets), 3
benchmark_datasets, 3
boston (benchmark_datasets), 3

calcAUC, 4
compactiv (benchmark_datasets), 3
compute_raw_proximity_info, 4
cov_importance_test, 5, 10

first_order_interaction_investigator,
6

get_tree_num_nodes_leaves_max_depths,
7

illustrate_trees, 8

model_fit_test, 10

ozone (benchmark_datasets), 3

pole (benchmark_datasets), 3
predict.YARF, 11
prediction_nodes, 11
print.YARF, 12
proximity_info, 13
prune_YARF_model, 13

query_variable_counts, 14
query_variable_proportions_across_trees,
15

set_YARF_num_cores, 15
shared_initial_substring, 16
summary.YARF, 17

tree_average_proximity_info, 17
triazine (benchmark_datasets), 3

wine.red (benchmark_datasets), 3
wine.white (benchmark_datasets), 3

YARF, 10, 18
YARF_all_oob_results_matrix, 23
YARF_convergence, 24
yarf_duplicate, 24
YARF_predict_all_trees, 25
YARF_progress, 22, 26
YARF_serialize, 26
YARF_set_aggregation_method, 27
YARF_stop, 28
YARF_update_with_oob_results, 22, 28, 34,
38
YARF_update_with_oob_test_results, 29
YARF_update_with_oob_validation_results,
30
YARFBAG, 30
YARFCART, 35
YARFMissForest, 39
YARFROC, 40