Package ‘pre’

June 10, 2022

Title  Prediction Rule Ensembles
Version  1.0.5
Author  Marjolein Fokkema [aut, cre],
        Benjamin Christoffersen [aut]
Maintainer  Marjolein Fokkema <m.fokkema@fsw.leidenuniv.nl>
Description  Derives prediction rule ensembles (PREs). Largely follows the
            procedure for deriving PREs as described in Friedman & Popescu (2008;
            <DOI:10.1214/07-AOAS148>), with adjustments and improvements. The
            main function pre() derives prediction rule ensembles consisting of
            rules and/or linear terms for continuous, binary, count, multinomial,
            and multivariate continuous responses. Function gpe() derives
            generalized prediction ensembles, consisting of rules, hinge and linear
            functions of the predictor variables.
URL  https://github.com/marjoleinF/pre
BugReports  https://github.com/marjoleinF/pre/issues
Depends  R (>= 3.5.0)
Imports  earth, Formula, glmnet, graphics, methods, partykit (>=
            1.2-0), rpart, stringr, survival, Matrix, MatrixModels
Suggests  interp, datasets, doParallel, foreach, glmertree, grid,
           mlbench, testthat, mboost, ggplot2, caret, pROC, knitr,
           rmarkdown, mice
License  GPL-2 | GPL-3
Encoding  UTF-8
LazyData  true
RoxygenNote  7.1.2
VignetteBuilder  knitr
NeedsCompilation  no
Repository  CRAN
Date/Publication  2022-06-10 21:20:08 UTC
bsnullinteract

Compute bootstrapped null interaction prediction rule ensembles

Description

bsnullinteract generates bootstrapped null interaction models, which can be used to derive a reference distribution of the test statistic calculated with interact.

Usage

bsnullinteract(
  object,
  nsamp = 10,
  parallel = FALSE,
  penalty.par.val = "lambda.1se",
)
bsnullinteract

verbose = FALSE,
...
)

Arguments

object          object of class pre.
nsamp           numeric. Number of bootstrapped null interaction models to be derived.
parallel        logical. Should parallel foreach be used to generate initial ensemble? Must
                register parallel beforehand, such as doMC or others.
penalty.par.val character or numeric. Value of the penalty parameter $\lambda$ to be employed for
                selecting the final ensemble. The default "lambda.min" employs the $\lambda$ value
                within 1 standard error of the minimum cross-validated error. Alternatively,
                "lambda.min" may be specified, to employ the $\lambda$ value with minimum cross-
                validated error, or a numeric value $> 0$ may be specified, with higher values
                yielding a sparser ensemble. To evaluate the trade-off between accuracy and
                sparsity of the final ensemble, inspect pre_object$glmnet.fit and plot(pre_object$glmnet.fit).
verbose         logical. should progress be printed to the command line?
...

Further arguments to be passed to predict.pre.

Details

Note that computation of bootstrapped null interaction models is computationally intensive. The
default number of samples is set to 10, but for reliable results argument nsamp should be set to a
higher value (e.g., $\geq 100$).

See also section 8.3 of Friedman & Popescu (2008).

Value

A list of length nsamp with null interaction models, to be used as input for interact.

References

Software, 92(12), 1-30. doi: 10.18637/jss.v092.i12

See Also

pre, interact

Examples

set.seed(42)
airq.ens <- pre(Ozone ~ ., data=airquality[complete.cases(airquality),])
nullmods <- bsnullinteract(airq.ens)
interact(airq.ens, nullmods = nullmods, col = c("#7FBFF5", ",#8CC876"))
Description
caret_pre_model is deprecated and provided for backwards compatibility only. The object provides a model setup for function train of package caret. It allows for tuning arguments sampfrac, maxdepth, learnrate, mtry, use.grad and penalty.par.val.

Usage
caret_pre_model

Format
An object of class list of length 17.

Details
Object caret_pre_model is deprecated, and only included in package pre for backward compatibility. Parameters of function pre() can be tuned by using method "pre" in caret's function train(). See vignette on tuning for more information and examples: vignette("Tuning", package = "pre")

Examples
## Object caret_pre_model is only included in package pre for backward compatibility
## By now, function pre can be optimized in the default way by using the method "pre"
## in caret's function train(). More information and instructions on tuning parameters
## of function pre() are provided in the vignette about tuning, which can be accessed
## from R by typing:
##
## vignette("Tuning", package = "pre")
##

carrillo

Data on personality characteristics and depressive symptom severity

Description
Dataset from a study by Carrillo et al. (2001), who assessed the extent to which the subscales of the NEO Personality Inventory (NEO-PI; Costa and McCrae 1985) could predict depressive symptomatology, as measured by the Beck Depression Inventory (BDI; Beck, Steer, and Carbin 1988). The NEO-PI assesses five major personality dimensions (Neuroticism, Extraversion, Openness to Experience, Agreeableness and Conscientiousness). Each of these dimensions consist of six specific subtraits (facets). The NEO-PI and BDI were administered to 112 Spanish respondents. Respondents' age in years and sex were also recorded and included in the dataset.
Usage
data(carrillo)

Format
A data frame with 112 observations and 26 variables

Details
- neuroticism facet and total scores: n1, n2, n3, n4, n5, n6, ntot
- extraversion facet and total scores: e1, e2, e3, e4, e5, e6, etot
- openness to experience facet and total scores: open1, open2, open3, open4, open5, open6, opentot
- altruism total score: altot
- conscientiousness total score: contot
- depression symptom severity: bdi
- sex: sexo
- age in years: edad

References

Examples
data("carrillo")
summary(carrillo)

calcgpe <- coef.gpe
calcgpe

Description
coef function for gpe

Usage
## S3 method for class 'gpe'
coef(object, penalty.par.val = "lambda.1se", ...)

coef.gpe

Coefficients for a General Prediction Ensemble (gpe)
Arguments

object argument of class pre
penalty.par.val character or numeric. Value of the penalty parameter \( \lambda \) to be employed for selecting the final ensemble. The default "lambda.min" employs the \( \lambda \) value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the \( \lambda \) value with minimum cross-validated error, or a numeric value > 0 may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect pre_object$glmnet.fit and plot(pre_object$glmnet.fit).

... Further arguments to be passed to coef.cv.glmnet.

See Also

coef.pre, gpe

coef.pre Coefficients for the final prediction rule ensemble

Description

cof.pre returns coefficients for prediction rules and linear terms in the final ensemble

Usage

## S3 method for class 'pre'
coef(object, penalty.par.val = "lambda.1se", ...)

Arguments

object argument of class pre
penalty.par.val character or numeric. Value of the penalty parameter \( \lambda \) to be employed for selecting the final ensemble. The default "lambda.min" employs the \( \lambda \) value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the \( \lambda \) value with minimum cross-validated error, or a numeric value > 0 may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect pre_object$glmnet.fit and plot(pre_object$glmnet.fit).

... Further arguments to be passed to coef.cv.glmnet.
Details

In some cases, duplicated variable names may appear in the model. For example, the first variable is a factor named 'V1' and there are also variables named 'V10' and/or 'V11' and/or 'V12' (etc). Then for the binary factor V1, dummy contrast variables will be created, named 'V10', 'V11', 'V12' (etc). As should be clear from this example, this yields duplicated variable names, which may yield problems, for example in the calculation of predictions and importances, later on. This can be prevented by renaming factor variables with numbers in their name, prior to analysis.

Value

returns a dataframe with 3 columns: coefficient, rule (rule or variable name) and description (NA for linear terms, conditions for rules).

See Also

pre, plot.pre, cvpre, importance.pre, predict.pre, interact, print.pre

Examples

set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airquality[complete.cases(airquality),])
coefs <- coef(airq.ens)
Arguments

- **object**: object of class pre
- **penalty.par.val**: character or numeric. Value of the penalty parameter \( \lambda \) to be employed for selecting the final ensemble. The default "lambda.min" employs the \( \lambda \) value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the \( \lambda \) value with minimum cross-validated error, or a numeric value > 0 may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect \( \text{pre_object$glmnet.fit} \) and plot(\( \text{pre_object$glmnet.fit} \)).
- **colors**: vector of contiguous colors to be used for plotting. If colors = NULL (default), colorRampPalette is used to generate a sequence of 200 colors going from red to white to blue. A different set of plotting colors can be specified here, for example: cm.colors(100), rainbow_hcl(100) (the latter requires package colorspace), or colorRampPalette(c("red", "yellow", "green")(100).
- **fig.plot**: plotting region to be used for correlation plot. See fig under par.
- **fig.legend**: plotting region to be used for legend. See fig under par.
- **legend.breaks**: numeric vector of breakpoints to be depicted in the plot’s legend. Should be a sequence from -1 to 1.

Examples

```r
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airquality[complete.cases(airquality),])
corplot(airq.ens)
```

```
cvpre
Full k-fold cross validation of a prediction rule ensemble (pre)
```

Description

cvpre performs k-fold cross validation on the dataset used to create the specified prediction rule ensemble, providing an estimate of predictive accuracy on future observations.

Usage

```r
cvpre(
    object,
    k = 10,
    penalty.par.val = "lambda.1se",
    pclass = 0.5,
    foldids = NULL,
    verbose = FALSE,
    parallel = FALSE,
    print = TRUE,
    ...
)
```
Arguments

object  An object of class pre.

k  integer. The number of cross validation folds to be used.

penalty.par.val  character or numeric. Value of the penalty parameter $\lambda$ to be employed for selecting the final ensemble. The default "lambda.min" employs the $\lambda$ value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the $\lambda$ value with minimum cross-validated error, or a numeric value $> 0$ may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect pre_object$glmnet.fit and plot(pre_object$glmnet.fit).

pclass numeric. Only used for binary classification. Cut-off value for the predicted probabilities that should be used to classify observations to the second class.

foldids numeric vector of length(nrow(object$data)) (the number of observations in the training data used to fit the original ensemble). Defaults to NULL, resulting in the original training observations being randomly assigned to one of the $k$ folds. Depending on sample size, the number of factors in the data, the number of factor levels and their distributions, the default may yield errors. See 'Details'.

verbose logical. Should progress of the cross validation be printed to the command line?

parallel logical. Should parallel foreach be used? Must register parallel beforehand, such as doMC or others.

print logical. Should accuracy estimates be printed to the command line?

... Further arguments to be passed to predict.pre.

Details

The random sampling employed by default may yield folds including all observations with a given level of a given factor. This results in an error, as it requires predictions for factor levels to be computed that were not observed in the training data, which is impossible. By manually specifying the foldids argument, users can make sure all class levels are represented in each of the $k$ training partitions.

Value

Calculates cross-validated estimates of predictive accuracy and prints these to the command line. For survival regression, accuracy is not calculated, as there is currently no agreed-upon way to best quantify accuracy in survival regression models. Users can compute their own accuracy estimates using the (invisibly returned) cross-validated predictions ($cvpreds$). Invisibly, a list of three objects is returned: accuracy (containing accuracy estimates), cvpreds (containing cross-validated predictions) and fold_indicators (a vector indicating the cross validation fold each observation was part of). For (multivariate) continuous outcomes, accuracy is a list with elements $\text{MSE}$ (mean squared error on test observations) and $\text{MAE}$ (mean absolute error on test observations). For (binary and multiclass) classification, accuracy is a list with elements $\text{SEL}$ (mean squared error on predicted probabilities), $\text{AEL}$ (mean absolute error on predicted probabilities), $\text{MCR}$ (average misclassification error rate) and $\text{table}$ (proportion table with (mis)classification rates).
See Also

pre, plot.pre, coef.pre, importance.pre, predict.pre, interact, print.pre

Examples

```r
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airquality[complete.cases(airquality),])
airq.cv <- cvpre(airq.ens)
```

```
explain

Explain predictions from final prediction rule ensemble

Description

explain shows which rules apply to which observations and visualizes the contribution of rules and linear predictors to the predicted values

Usage

explain(
  object,  # object of class pre.
  newdata,  # optional dataframe of new (test) observations, including all predictor variables used for deriving the prediction rule ensemble.
  penalty.par.val = "lambda.1se",  # penalty parameter value
  response = 1L,  # index of response variable
  plot = TRUE,  # plot results
  intercept = FALSE,  # include intercept in plot
  center.linear = FALSE,  # center linear predictors
  plot.max.nobs = 4,  # maximum number of observations to plot
  plot.dim = c(2, 2),  # plot dimensions
  plot.obs.names = TRUE,  # plot observation names
  pred.type = "response",  # prediction type
  digits = 3L,  # number of digits
  cex = 0.8,  # text size
  ylab = "Contribution to linear predictor",  # y-axis label
  bar.col = c("#E495A5", "#39BEB1"),  # bar colors
  rule.col = "darkgrey",  # rule color
  ...
)
```

Arguments

- `object`: object of class pre.
- `newdata`: optional dataframe of new (test) observations, including all predictor variables used for deriving the prediction rule ensemble.
penalty.par.val

character or numeric. Value of the penalty parameter $\lambda$ to be employed for selecting the final ensemble. The default "lambda.min" employs the $\lambda$ value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the $\lambda$ value with minimum cross-validated error, or a numeric value $> 0$ may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect `pre_object$glmnet.fit` and `plot(pre_object$glmnet.fit)`. 

response

numeric or character vector of length one. Specifies the name or number of the response variable (for multivariate responses) or the name or number of the factor level (for multinomial responses) for which explanations and contributions should be computed and/or plotted. Only used for `pre` fitted to multivariate or multinomial responses.

plot

logical. Should explanations be plotted?

intercept

logical. Specifies whether intercept should be included in explaining predictions.

center.linear

logical. Specifies whether linear terms should be centered with respect to the training sample mean before computing their contribution to the predicted value. If `intercept = TRUE`, this will also affect the intercept. That is, the value of the intercept returned will differ from that of the value returned by the `print` method.

plot.max.nobs

numeric. Specifies maximum number of observations for which explanations will be plotted. The default (4) plots the explanation for the first four observations supplied in `newdata`.

plot.dim

numeric vector of length 2. Specifies the number of rows and columns in the resulting plot.

plot.obs.names

logical vector of length 1, NULL, or character vector of length `nrow(data)` supplying the names that should be used for individual observations’ plots. If `TRUE` (default), `rownames(newdata)` will be used as titles. If `NULL`, `paste("Observation", 1:nrow(newdata))` will be used as titles. If `FALSE`, no titles will be plotted.

pred.type

character. Specifies the type of predicted values to be computed, returned and provided in the plot(s). Note that the computed contributions must be additive and are therefore always on the scale of the linear predictor.

digits

integer. Specifies the number of digits used in depicting the predicted values in the plot.

cex

numeric. Specifies the relative text size of title, tick and axis labels.
ylab

character. Specifies the label for the horizontal (y-) axis.

bar.col

character vector of length two. Specifies the colors to be used for plotting the positive and negative contributions to the predictions, respectively.

rule.col

character. Specifies the color to be used for plotting the rule descriptions. If `NULL`, rule descriptions are not plotted.

... Further arguments to be passed to `predict.pre` and `predict.cv.glmnet`. 

explain

penalty.par.val

character or numeric. Value of the penalty parameter $\lambda$ to be employed for selecting the final ensemble. The default "lambda.min" employs the $\lambda$ value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the $\lambda$ value with minimum cross-validated error, or a numeric value $> 0$ may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect `pre_object$glmnet.fit` and `plot(pre_object$glmnet.fit)`.
Details

Provides a graphical depiction of the contribution of rules and linear terms to the individual predictions (if `plot = TRUE`). Invisibly returns a list with objects `predictors` and `contribution`. `predictors` contains the values of the rules and linear terms for each observation in `newdata`, for those rules and linear terms included in the final ensemble with the specified value of `penalty.par.val`. `contribution` contains the values of `predictors`, multiplied by the estimated values of the coefficients in the final ensemble selected with the specified value of `penalty.par.val`. All contributions are calculated w.r.t. the intercept, by default. Thus, if a given rule applies to an observation in `newdata`, the contribution of that rule equals the estimated coefficient of that rule. If a given rule does not apply to an observation in `newdata`, the contribution of that rule equals 0. For linear terms, contributions can be centered, or not (the default). Thus, by default the contribution of a linear terms for an observation in `newdata` equals the observation’s value of the linear term, times the estimated coefficient of the linear term. If `center.linear = TRUE`, the contribution of a linear term for an observation in `newdata` equals (the value of the linear term, minus the mean value of the linear term in the training data) times the estimated coefficient for the linear term.

References


See Also

`pre`, `plot.pre`, `coef.pre`, `importance.pre`, `cvpre`, `interact`, `print.pre`

Examples

```r
airq <- airquality[complete.cases(airquality), ]
set.seed(1)
train <- sample(1:nrow(airq), size = 100)
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airq[train, ])
airq.ens.exp <- explain(airq.ens, newdata = airq[-train, ])
airq.ens.exp$predictors
airq.ens.exp$contribution

## Can also include intercept in explanation:
airq.ens.exp <- explain(airq.ens, newdata = airq[-train, ])

## Fit PRE with linear terms only to illustrate effect of center.linear:
set.seed(42)
airq.ens2 <- pre(Ozone ~ ., data = airq[train, ], type = "linear")
## When not centered around their means, Month has negative and
## Day has positive contribution:
explain(airq.ens2, newdata = airq[-train, ][1:2, ],
       penalty.par.val = "lambda.min")$contribution
## After mean centering, contributions of Month and Day have switched
## sign (for these two observations):
explain(airq.ens2, newdata = airq[-train, ][1:2, ],
```
Derive a General Prediction Ensemble (gpe)

Description

Provides an interface for deriving sparse prediction ensembles where basis functions are selected through L1 penalization.

Usage

gpe(
  formula,
  data,
  base_learners = list(gpe_trees(), gpe_linear()),
  weights = rep(1, times = nrow(data)),
  sample_func = gpe_sample(),
  verbose = FALSE,
  penalized_trainer = gpe_cv.glmnet(),
  model = TRUE
)

Arguments

- **formula**: Symbolic description of the model to be fit of the form y ~ x1 + x2 + ... + xn. If the output variable (left-hand side of the formula) is a factor, an ensemble for binary classification is created. Otherwise, an ensemble for prediction of a continuous variable is created.
- **data**: data.frame containing the variables in the model.
- **base_learners**: List of functions which has formal arguments formula, data, weights, sample_func, verbose, and family and returns a vector of characters with terms for the final formula passed to cv.glmnet. See gpe_linear, gpe_trees, and gpe_earth.
- **weights**: Case weights with length equal to number of rows in data.
- **sample_func**: Function used to sample when learning with base learners. The function should have formal argument n and weights and return a vector of indices. See gpe_sample.
- **verbose**: TRUE if comments should be posted throughout the computations.
- **penalized_trainer**: Function with formal arguments x, y, weights, family which returns a fit object. This can be changed to test other "penalized trainers" (like other function that perform an L1 penalty or L2 penalty and elastic net penalty). Not using cv.glmnet may cause other function for gpe objects to fail. See gpe_cv.glmnet.
- **model**: TRUE if the data should added to the returned object.
Details

Provides a more general framework for making a sparse prediction ensemble than pre. By default, a similar fit to pre is obtained. In addition, multivariate adaptive regression splines (Friedman, 1991) can be included with gpe_earth. See examples.

Other custom base learners can be implemented. See gpe_trees, gpe_linear or gpe_earth for details of the setup. The sampling function given by sample_func can also be replaced by a custom sampling function. See gpe_sample for details of the setup.

Value

An object of class gpe.

References


See Also

pre, gpe_trees, gpe_linear, gpe_earth, gpe_sample, gpe_cv.glmnet

Examples

## Not run:
## Obtain similar fit to function pre:
gpe.rules <- gpe(Ozone ~ ., data = airquality[complete.cases(airquality),],
  base_learners = list(gpe_linear(), gpe_trees()))
gpe.rules

## Also include products of hinge functions using MARS:
gpe.hinge <- gpe(Ozone ~ ., data = airquality[complete.cases(airquality),],
  base_learners = list(gpe_linear(), gpe_trees(), gpe_earth()))

## End(Not run)
Arguments

... arguments to cv.glmnet. x, y, weights and family will not be used.

Value

Returns a function with formal arguments x, y, weights, family and returns a fit object.

See Also

gpe

description

gpe_rules_pre generates a learner which generates rules like pre, which can be supplied to the gpe base_learner argument.

Usage

gpe_rules_pre(
  learnrate = 0.01,
  par.init = FALSE,
  mtry = Inf,
  maxdepth = 3L,
  ntrees = 500,
  tree.control = ctree_control(),
  use.grad = TRUE,
  removeduplicates = TRUE,
  removecomplements = TRUE,
  tree.unbiased = TRUE
)

Arguments

learnrate numeric value > 0. Learning rate or boosting parameter.

par.init logical. Should parallel foreach be used to generate initial ensemble? Only used when learnrate == 0. Note: Must register parallel beforehand, such as doMC or others. Furthermore, setting par.init = TRUE will likely only increase computation time for smaller datasets.

mtry positive integer. Number of randomly selected predictor variables for creating each split in each tree. Ignored when tree.unbiased=FALSE.
maxdepth  positive integer. Maximum number of conditions in rules. If length(maxdepth) == 1, it specifies the maximum depth of of each tree grown. If length(maxdepth) == ntrees, it specifies the maximum depth of every consecutive tree grown. Alternatively, a random sampling function may be supplied, which takes argument ntrees and returns integer values. See also maxdepth_sampler.

ntrees  positive integer value. Number of trees to generate for the initial ensemble.

tree.control  list with control parameters to be passed to the tree fitting function, generated using ctree_control, mob_control (if use.grad = FALSE), or rpart.control (if tree.unbiased = FALSE).

use.grad  logical. Should gradient boosting with regression trees be employed when learnrate > 0? If TRUE, use trees fitted by ctree or rpart as in Friedman (2001), but without the line search. If use.grad = FALSE, glmtree instead of ctree will be employed for rule induction, yielding longer computation times, higher complexity, but possibly higher predictive accuracy. See Details for supported combinations of family, use.grad and learnrate.

removeduplicates  logical. Remove rules from the ensemble which are identical to an earlier rule?

removecomplements  logical. Remove rules from the ensemble which are identical to (1 - an earlier rule)?

tree.unbiased  logical. Should an unbiased tree generation algorithm be employed for rule generation? Defaults to TRUE, if set to FALSE, rules will be generated employing the CART algorithm (which suffers from biased variable selection) as implemented in rpart. See details below for possible combinations with family, use.grad and learnrate.

Examples

## Obtain same fits with pre and gpe
set.seed(42)
gpe.mod <- gpe(Ozone ~ ., data = airquality[complete.cases(airquality),],
    base_learners = list(gpe_rules_pre(), gpe_linear()))
gpe.mod
set.seed(42)
pre.mod <- pre(Ozone ~ ., data = airquality[complete.cases(airquality),])
pre.mod

---

**gpe_sample**

*Sampling Function Generator for gpe*

**Description**

Provides a sample function for *gpe*.

**Usage**

gpe_sample(sampfrac = 0.5)
gpe_trees

Arguments

sampfrac Fraction of \( n \) to use for sampling. It is the \( \eta/N \) in Friedman & Popescu (2008).

Value

Returns a function that takes an \( n \) argument for the number of observations and a weights argument for the case weights. The function returns a vector of indices.

References


See Also

gpe

gpe_trees

Learner Functions Generators for gpe

Description

Functions to get "learner" functions for gpe.

Usage

gpe_trees(
  ..., remove_duplicates_complements = TRUE,
  mtry = Inf, ntrees = 500, maxdepth = 3L,
  learnrate = 0.01, parallel = FALSE,
  use_grad = TRUE,
  tree.control = ctree_control(mtry = mtry, maxdepth = maxdepth)
)

gpe_linear(..., winsfrac = 0.025, normalize = TRUE)

gpe_earth(
  ..., degree = 3, nk = 8, normalize = TRUE,
  ntrain = 100, learnrate = 0.1, cor_thresh = 0.99
)
Arguments

... Currently not used.
remove_duplicates_complements
   TRUE. Should rules with complementary or duplicate support be removed?
mtry
   Number of input variables randomly sampled as candidates at each node for random forest like algorithms. The argument is passed to the tree methods in the partykit package.
ntrees
   Number of trees to fit. Will not have an effect if tree.control is used.
maxdepth
   Maximum depth of trees. Will not have an effect if tree.control is used.
learnrate
   Learning rate for methods. Corresponds to the $\nu$ parameter in Friedman & Popescu (2008).
parallel
   TRUE. Should basis functions be found in parallel?
use_grad
   TRUE. Should binary outcomes use gradient boosting with regression trees when learnrate > 0? That is, use ctree instead of glmtree as in Friedman (2001) with a second order Taylor expansion instead of first order as in Chen and Guestrin (2016).
tree.control
   ctree_control with options for the ctree function.
winsfrac
   Quantile to winsorize linear terms. The value should be in [0, 0.5]
normalize
   TRUE. Should value be scaled by .4 times the inverse standard deviation? If TRUE, gives linear terms the same influence as a typical rule.
degree
   Maximum degree of interactions in earth model.
nk
   Maximum number of basis functions in earth model.
ntrain
   Number of models to fit.
cor_thresh
   A threshold on the pairwise correlation for removal of basis functions. This is similar to remove_duplicates_complements. One of the basis functions in pairs where the correlation exceeds the threshold is excluded. NULL implies no exclusion. Setting a value closer to zero will decrease the time needed to fit the final model.

Details

gpe_trees provides learners for tree method. Either ctree or glmtree from the partykit package will be used.
gpe_linear provides linear terms for the gpe.
gpe_earth provides basis functions where each factor is a hinge function. The model is estimated with earth.

Value

A function that has formal arguments formula, data, weights, sample_func, verbose, family, ... The function returns a vector with character where each element is a term for the final formula in the call to cv.glmnet.
importance.pre

References


See Also
gpe, rTerm, lTerm, eTerm

importance.pre

Calculate importances of baselearners and input variables in a prediction rule ensemble (pre)

Description

importance.pre calculates importances for rules, linear terms and input variables in the prediction rule ensemble (pre), and creates a bar plot of variable importances.

Usage

## S3 method for class 'pre'
importance(
  x,
  standardize = FALSE,
  global = TRUE,
  penalty.par.val = "lambda.1se",
  gamma = NULL,
  quantprobs = c(0.75, 1),
  round = NA,
  plot = TRUE,
  ylab = "Importance",
  main = "Variable importances",
  abbreviate = 10L,
  diag.xlab = TRUE,
  diag.xlab.hor = 0,
  diag.xlab.vert = 2,
  cex.axis = 1,
legend = "topright",
...
)

Arguments

x an object of class pre
standardize logical. Should baselearner importances be standardized with respect to the outcome variable? If TRUE, baselearner importances have a minimum of 0 and a maximum of 1. Only used for ensembles with numeric (non-count) response variables.
global logical. Should global importances be calculated? If FALSE, local importances will be calculated, given the quantiles of the predictions F(x) in quantprobs.
penalty.par.val character or numeric. Value of the penalty parameter $\lambda$ to be employed for selecting the final ensemble. The default "lambda.min" employs the $\lambda$ value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the $\lambda$ value with minimum cross-validated error, or a numeric value $> 0$ may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect pre_object$glmnet.fit and plot(pre_object$glmnet.fit).
gamma Mixing parameter for relaxed fits. See coef.cv.glmnet.
quantprobs optional numeric vector of length two. Only used when global = FALSE. Probabilities for calculating sample quantiles of the range of F(X), over which local importances are calculated. The default provides variable importances calculated over the 25% highest values of F(X).
round integer. Number of decimal places to round numeric results to. If NA (default), no rounding is performed.
plot logical. Should variable importances be plotted?
ylab character string. Plotting label for y-axis. Only used when plot = TRUE.
main character string. Main title of the plot. Only used when plot = TRUE.
abbreviate integer or logical. Number of characters to abbreviate x axis names to. If FALSE, no abbreviation is performed.
diag.xlab logical. Should variable names be printed diagonally (that is, in a 45 degree angle)? Alternatively, variable names may be printed vertically by specifying diag.xlab = FALSE and las = 2.
diag.xlab.hor numeric. Horizontal adjustment for lining up variable names with bars in the plot if variable names are printed diagonally.
diag.xlab.vert positive integer. Vertical adjustment for position of variable names, if printed diagonally. Corresponds to the number of character spaces added after variable names.
cex.axis numeric. The magnification to be used for axis annotation relative to the current setting of cex.
legend logical or character. Should legend be plotted for multinomial or multivariate responses and if so, where? Defaults to "topright", which puts the legend in the top-right corner of the plot. Alternatively, "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center" and FALSE (which omits the legend) can be specified.

... further arguments to be passed to barplot (only used when plot = TRUE).

Details

See also sections 6 and 7 of Friedman & Popescu (2008).

Value

A list with two dataframes: $baseimps, giving the importances for baselearners in the ensemble, and $varimps, giving the importances for all predictor variables.

References


See Also

pre

Examples

```
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airquality[complete.cases(airquality),])
# calculate global importances:
importance(airq.ens)
# calculate local importances (default: over 25% highest predicted values):
importance(airq.ens, global = FALSE)
# calculate local importances (custom: over 25% lowest predicted values):
importance(airq.ens, global = FALSE, quantprobs = c(0, .25))
```
interact  

**Calculate interaction statistics for variables in a prediction rule ensemble (pre)**

### Description

interact calculates test statistics for assessing the strength of interactions between a set of user-specified input variable(s), and all other input variables.

### Usage

```r
interact(
  object,
  varnames = NULL,
  penalty.par.val = "lambda.1se",
  gamma = NULL,
  nullmods = NULL,
  quantprobs = c(0.05, 0.95),
  plot = TRUE,
  col = c("darkgrey", "lightgrey"),
  ylab = "Interaction strength",
  main = "Interaction test statistics",
  se.linewidth = 0.05,
  legend.text = c("observed", "null model median"),
  parallel = FALSE,
  k = 10,
  verbose = FALSE,
  ...
)
```

### Arguments

- **object**: an object of class `pre`.
- **varnames**: character vector. Names of variables for which interaction statistics should be calculated. If `NULL`, interaction statistics for all predictor variables with non-zero coefficients will be calculated (which may take a long time).
- **penalty.par.val**: character or numeric. Value of the penalty parameter $\lambda$ to be employed for selecting the final ensemble. The default "lambda.min" employs the $\lambda$ value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the $\lambda$ value with minimum cross-validated error, or a numeric value $> 0$ may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect `pre_object$glmnet.fit` and `plot(pre_object$glmnet.fit)`.
- **gamma**: Mixing parameter for relaxed fits. See `coef.cv.glmnet`.
nullmods object with bootstrapped null interaction models, resulting from application of bsnullinteract.

quantprobs numeric vector of length two. Probabilities that should be used for plotting the range of bootstrapped null interaction model statistics. Only used when nullmods argument is specified and plot = TRUE. The default yields sample quantiles corresponding to .05 and .95 probabilities.

plot logical. Should interaction statistics be plotted?

col character vector of length one or two. The first value specifies the color to be used for plotting the interaction statistic from the training data, the second color is used for plotting the interaction statistic from the bootstrapped null interaction models. Only used when plot = TRUE. Only the first element will be used if nullmods = NULL.

ylab character string. Label to be used for plotting y-axis.

main character. Main title for the bar plot.

se.linewidth numeric. Width of the whiskers of the plotted standard error bars (in inches).

legend.text character vector of length two to be used for plotting the legend. Only used when nullmods is specified. If FALSE, no legend is plotted.

parallel logical. Should parallel foreach be used? Must register parallel beforehand, such as doMC or others.

k integer. Calculating interaction test statistics is computationally intensive, so calculations are split up in several parts to prevent memory allocation errors. If a memory allocation error still occurs, increase k.

verbose logical. Should progress information be printed to the command line?

... Further arguments to be passed to barplot.

Details

Can be computationally intensive, especially when nullmods is specified, in which case setting parallel = TRUE may improve speed.

Value

Function interact() returns and plots interaction statistics for the specified predictor variables. If nullmods is not specified, it returns and plots only the interaction test statistics for the specified fitted prediction rule ensemble. If nullmods is specified, the function returns a list, with elements $fittedH2, containing the interaction statistics of the fitted ensemble, and $nullH2, which contains the interaction test statistics for each of the bootstrapped null interaction models.

If plot = TRUE (the default), a barplot is created with the interaction test statistic from the fitted prediction rule ensemble. If nullmods is specified, bars representing the median of the distribution of interaction test statistics of the bootstrapped null interaction models are plotted. In addition, error bars representing the quantiles of the distribution (their value specified by the quantprobs argument) are plotted. These allow for testing the null hypothesis of no interaction effect for each of the input variables.

Note that the error rates of null hypothesis tests of interaction effects have not yet been studied in detail, but results are likely to get more reliable when the number of bootstrapped null interaction
models is larger. The default of the bsnullinteract function is to generate 10 bootstrapped null interaction datasets, to yield shorter computation times. To obtain a more reliable result, however, users are advised to set the nsamp argument ≥ 100.

See also section 8 of Friedman & Popescu (2008).

References


See Also
`pre`, `bsnullinteract`

Examples
```
set.seed(42)
airq.ens <- pre(Ozone ~ ., data=airquality[complete.cases(airquality),])
interact(airq.ens, c("Temp", "Wind", "Solar.R"))
```

---

### maxdepth_sampler

Sampling function generator for specifying varying maximum tree depth in a prediction rule ensemble (pre)

**Description**

`maxdepth_sampler` generates a random sampling function, governed by a pre-specified average tree depth.

**Usage**

`maxdepth_sampler(av.no.term.nodes = 4L, av.tree.depth = NULL)`

**Arguments**

- `av.no.term.nodes` integer of length one. Specifies the average number of terminal nodes in trees used for rule induction.
- `av.tree.depth` integer of length one. Specifies the average maximum tree depth in trees used for rule induction.
maxdepth_sampler

Details

The original RuleFit implementation varying tree sizes for rule induction. Furthermore, it defined tree size in terms of the number of terminal nodes. In contrast, function pre defines the maximum tree size in terms of a (constant) tree depth. Function maxdepth_sampler allows for mimicing the behavior of the original RuleFit implementation. In effect, the maximum tree depth is sampled from an exponential distribution with learning rate $1/(\bar{L} - 2)$, where $\bar{L} \geq 2$ represents the average number of terminal nodes for trees in the ensemble. See Friedman & Popescu (2008, section 3.3).

Value

Returns a random sampling function with single argument ntree, which can be supplied to the maxdepth argument of function pre to specify varying tree depths.

References


See Also

pre

Examples

```r
## RuleFit default is max. 4 terminal nodes, on average:
func1 <- maxdepth_sampler()
set.seed(42)
func1(10)
mean(func1(1000))

## Max. 16 terminal nodes, on average (equals average maxdepth of 4):
func2 <- maxdepth_sampler(average_maxdepth = 16)
set.seed(42)
func2(10)
mean(func2(1000))

## Max. tree depth of 3, on average:
func3 <- maxdepth_sampler(tree_depth = 3)
set.seed(42)
func3(10)
mean(func3(1000))

## Max. 2 of terminal nodes, on average (always yields maxdepth of 1):
func4 <- maxdepth_sampler(average_maxdepth = 2)
set.seed(42)
func4(10)
mean(func4(1000))

## Create rule ensemble with varying maxdepth:
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airquality[complete.cases(airquality),]
```
pairplot

Create partial dependence plot for a pair of predictor variables in a prediction rule ensemble (pre)

Description

pairplot creates a partial dependence plot to assess the effects of a pair of predictor variables on the predictions of the ensemble. Note that plotting partial dependence is computationally intensive. Computation time will increase fast with increasing numbers of observations and variables. For large datasets, package ‘plotmo’ (Milborrow, 2019) provides more efficient functions for plotting partial dependence and also supports ‘pre’ models.

Usage

pairplot(
  object, 
  varnames, 
  type = "both", 
  gamma = NULL, 
  penalty.par.val = "lambda.1se", 
  nvals = c(20L, 20L), 
  pred.type = "response", 
  ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>an object of class pre</td>
</tr>
<tr>
<td>varnames</td>
<td>character vector of length two. Currently, pairplots can only be requested for non-nominal variables. If varnames specifies the name(s) of variables of class &quot;factor&quot;, an error will be printed.</td>
</tr>
<tr>
<td>type</td>
<td>character string. Type of plot to be generated. type = &quot;heatmap&quot; yields a heatmap plot, type = &quot;contour&quot; yields a contour plot, type = &quot;both&quot; yields a heatmap plot with added contours, type = &quot;perspective&quot; yields a three dimensional plot.</td>
</tr>
<tr>
<td>gamma</td>
<td>Mixing parameter for relaxed fits. See coef.cv.glmnet.</td>
</tr>
<tr>
<td>penalty.par.val</td>
<td>character or numeric. Value of the penalty parameter λ to be employed for selecting the final ensemble. The default &quot;lambda.min&quot; employs the λ value within 1 standard error of the minimum cross-validated error. Alternatively, &quot;lambda.min&quot; may be specified, to employ the λ value with minimum cross-validated error, or a numeric value &gt; 0 may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect pre_object$glmnet.fit and plot(pre_object$glmnet.fit).</td>
</tr>
</tbody>
</table>
pairplot

**nvals**
optional numeric vector of length 2. For how many values of x1 and x2 should partial dependence be plotted? If NULL, all observed values for the two predictor variables specified will be used (see details).

**pred.type**
character string. Type of prediction to be plotted on z-axis. pred.type = "response" gives fitted values for continuous outputs and fitted probabilities for nominal outputs. pred.type = "link" gives fitted values for continuous outputs and linear predictor values for nominal outputs.

... Further arguments to be passed to image, contour or persp (depending on whether type is specified to be "heatmap", "contour", "both" or "perspective").

**Details**

By default, partial dependence will be plotted for each combination of 20 values of the specified predictor variables. When nvals = NULL is specified a dependence plot will be created for every combination of the unique observed values of the two predictor variables specified. Therefore, using nvals = NULL will often result in long computation times, and / or memory allocation errors. Also, pre ensembles derived from training datasets that are very wide or long may result in long computation times and / or memory allocation errors. In such cases, reducing the values supplied to nvals will reduce computation time and / or memory allocation errors. When the nvals argument is supplied, values for the minimum, maximum, and nvals - 2 intermediate values of the predictor variable will be plotted. Furthermore, if none of the variables specified appears in the final prediction rule ensemble, an error will occur.

See also section 8.1 of Friedman & Popescu (2008).

**References**


**See Also**

pre, singleplot

**Examples**

```r
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airquality[complete.cases(airquality),])
pairplot(airq.ens, c("Temp", "Wind"))
```
plot.pre   

Plot method for class pre

Description

plot.pre creates one or more plots depicting the rules in the final ensemble as simple decision trees.

Usage

## S3 method for class 'pre'
plot(
  x, 
  penalty.par.val = "lambda.1se", 
  gamma = NULL, 
  linear.terms = TRUE, 
  nterms = NULL, 
  fill = "white", 
  ask = FALSE, 
  exit.label = "0", 
  standardize = FALSE, 
  plot.dim = c(3, 3), 
  ...
)

Arguments

x       an object of class pre.
penalty.par.val  character or numeric. Value of the penalty parameter \(\lambda\) to be employed for selecting the final ensemble. The default "lambda.min" employs the \(\lambda\) value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the \(\lambda\) value with minimum cross-validated error, or a numeric value > 0 may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect pre_object$glmnet.fit and plot(pre_object$glmnet.fit).
gamma  Mixing parameter for relaxed fits. See coef.cv.glmnet.
linear.terms  logical. Should linear terms be included in the plot?
nterms  numeric. The total number of terms (or rules, if linear.terms = FALSE) being plotted. Default is NULL, resulting in all terms of the final ensemble to be plotted.
fill  character of length 1 or 2. Background color(s) for terminal panels. If one color is specified, all terminal panels will have the specified background color. If two colors are specified (the default, the first color will be used as the background color for rules with a positively valued coefficient; the second color for rules with a negatively valued coefficient.
Functions 

`pre` 

Derive a prediction rule ensemble 

Description

Function `pre` derives a sparse ensemble of rules and/or linear functions for prediction of a continuous, binary, count, multinomial, multivariate continuous or survival response. 

Usage

`pre(formula, data, family = gaussian, use.grad = TRUE, weights, type = "both", sampfrac = 0.5, maxdepth = 3L, learnrate = 0.01, mtry = Inf, ntrees = 500, confirmatory = NULL, singleconditions = FALSE, winsfrac = 0.025, normalize = TRUE, standardize = FALSE, ask = logical. Should user be prompted before starting a new page of plots? exit.label = character string. Label to be printed in nodes to which the rule does not apply ("exit nodes")? standardize = logical. Should printed importances be standardized? See `importance.pre`. plot.dim = integer vector of length two. Specifies the number of rows and columns in the plot. The default yields a plot with three rows and three columns, depicting nine baselearners per plotting page.

... Arguments to be passed to `gpar`. 

See Also

`pre, print.pre` 

Examples

```r
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airquality[complete.cases(airquality),])
plot(airq.ens)
```
```r
ordinal = TRUE,
nfolds = 10L,
tree.control,
tree.unbiased = TRUE,
removeduplicates = TRUE,
verbose = FALSE,
par.init = FALSE,
par.final = FALSE,
sparse = FALSE,
...
```

**Arguments**

`formula` a symbolic description of the model to be fit of the form `y ~ x1 + x2 + ... + xn`. Response (left-hand side of the formula) should be of class numeric (for family = "gaussian" or "mgaussian"), integer (for family = "poisson"), factor (for family = "binomial" or "multinomial"). See Examples below. Note that the minus sign (-) may not be used in the formula to omit the intercept or variables in `data`, and neither should `+ 0` be used to omit the intercept. To omit the intercept from the final ensemble, add `intercept = FALSE` to the call (although omitting the intercept from the final ensemble will only very rarely be appropriate). To omit variables from the final ensemble, make sure they are excluded from `data`.

`data` data frame containing the variables in the model. Response must be of class factor for classification, numeric for (count) regression, `Surv` for survival regression. Input variables must be of class numeric, factor or ordered factor. Otherwise, `pre` will attempt to recode.

`family` specifies a glm family object. Can be a character string (i.e., "gaussian", "binomial", "poisson", "multinomial", "cox" or "mgaussian"), or a corresponding family object (e.g., `gaussian`, `binomial` or `poisson`, see `family`). Specification of argument `family` is strongly advised but not required. If `family` is not specified, `pre` will try to make an informed guess, based on the class of the response variable specified in `formula`. als see Examples below.

`use.grad` logical. Should gradient boosting with regression trees be employed when `learnrate > 0`? If TRUE, use trees fitted by `cTree` or `rpart` as in Friedman (2001), but without the line search. If `use.grad = FALSE`, `glmtree` instead of `cTree` will be employed for rule induction, yielding longer computation times, higher complexity, but possibly higher predictive accuracy. See Details for supported combinations of `family`, `use.grad` and `learnrate`.

`weights` optional vector of observation weights to be used for deriving the ensemble.

`type` character. Specifies type of base learners to include in the ensemble. Defaults to "both" (initial ensemble will include both rules and linear functions). Other option are "rules" (prediction rules only) or "linear" (linear functions only).

`sampfrac` numeric value > 0 and ≤ 1. Specifies the fraction of randomly selected training observations used to produce each tree. Values < 1 will result in sampling
without replacement (i.e., subsampling), a value of 1 will result in sampling with replacement (i.e., bootstrap sampling). Alternatively, a sampling function may be supplied, which should take arguments \( n \) (sample size) and weights.

- **maxdepth**: positive integer. Maximum number of conditions in rules. If \( \text{length}(\text{maxdepth}) == 1 \), it specifies the maximum depth of of each tree grown. If \( \text{length}(\text{maxdepth}) == \text{ntrees} \), it specifies the maximum depth of every consecutive tree grown. Alternatively, a random sampling function may be supplied, which takes argument \( \text{ntrees} \) and returns integer values. See also maxdepth_sampler.

- **learnrate**: numeric value > 0. Learning rate or boosting parameter.

- **mtry**: positive integer. Number of randomly selected predictor variables for creating each split in each tree. Ignored when \( \text{tree.unbiased} = \text{FALSE} \).

- **ntrees**: positive integer value. Number of trees to generate for the initial ensemble.

- **confirmatory**: character vector. Specifies one or more confirmatory terms to be included in the final ensemble. Linear terms can be specified as the name of a predictor variable included in data, rules can be specified as, for example, "x1 > 6 & x2 <= 8", where x1 and x2 should be names of variables in data. Terms thus specified will be included in the final ensemble, as their coefficients will not be penalized in the estimation.

- **singleconditions**: TRUE, FALSE or "only". Should rules with multiple conditions be disaggregated? Node membership for all tree except the root are coded as multi-condition rules. The conditions of these rules can be disaggregated to avoid selection of multi-condition rules. If FALSE (the default), all non-root nodes will be included as multi-condition rules in the initial ensemble. If TRUE, all nodes will additionally be included as single-condition rules. If "only", all nodes will be included as single-condition rules only.

- **winsfrac**: numeric value > 0 and \( \leq 0.5 \). Quantiles of data distribution to be used for winsorizing linear terms. If set to 0, no winsorizing is performed. Note that ordinal variables are included as linear terms in estimating the regression model and will also be winsorized.

- **normalize**: logical. Normalize linear variables before estimating the regression model? Normalizing gives linear terms the same a priori influence as a typical rule, by dividing the (winsorized) linear term by 2.5 times its SD. normalize = FALSE will give more preference to linear terms for selection.

- **standardize**: logical. Should rules and linear terms be standardized to have SD equal to 1 before estimating the regression model? This will also standardize the dummyfied factors, users are advised to use the default standardize = FALSE.

- **ordinal**: logical. Should ordinal variables (i.e., ordered factors) be treated as continuous for generating rules? TRUE (the default) generally yields simpler rules, shorter computation times and better generalizability of the final ensemble.

- **nfolds**: positive integer. Number of cross-validation folds to be used for selecting the optimal value of the penalty parameter \( \lambda \) in selecting the final ensemble.

- **tree.control**: list with control parameters to be passed to the tree fitting function, generated using ctree.control, mob.control (if use.grad = FALSE), or rpart.control (if tree.unbiased = FALSE).
tree.unbiased logical. Should an unbiased tree generation algorithm be employed for rule generation? Defaults to TRUE, if set to FALSE, rules will be generated employing the CART algorithm (which suffers from biased variable selection) as implemented in rpart. See details below for possible combinations with family, use.grad and learnrate.

removecomplements logical. Remove rules from the ensemble which are identical to (1 - an earlier rule)?

removeduplicates logical. Remove rules from the ensemble which are identical to an earlier rule?

verbose logical. Should progress be printed to the command line?

par.init logical. Should parallel foreach be used to generate initial ensemble? Only used when learnrate == 0. Note: Must register parallel beforehand, such as doMC or others. Furthermore, setting par.init = TRUE will likely only increase computation time for smaller datasets.

par.final logical. Should parallel foreach be used to perform cross validation for selecting the final ensemble? Must register parallel beforehand, such as doMC or others.

sparse logical. Should sparse design matrices be used? May improve computation times for large datasets.

... Further arguments to be passed to cv.glmnet.

Details

Note: obervations with missing values will be removed prior to analysis (and a warning printed).

In some cases, duplicated variable names may appear in the model. For example, the first variable is a factor named 'V1' and there are also variables named 'V10' and/or 'V11' and/or 'V12' (etc). Then for for the binary factor V1, dummy contrast variables will be created, named 'V10', 'V11', 'V12' (etc). As should be clear from this example, this yields duplicated variable names, which may yield problems, for example in the calculation of predictions and importances, later on. This can be prevented by renaming factor variables with numbers in their name, prior to analysis.

The table below provides an overview of combinations of response variable types, use.grad, tree.unbiased and learnrate settings that are supported, and the tree induction algorithm that will be employed as a result:

<table>
<thead>
<tr>
<th>use.grad</th>
<th>tree.unbiased</th>
<th>learnrate</th>
<th>family</th>
<th>tree alg.</th>
<th>Response variable format</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>0</td>
<td>gaussian</td>
<td>ctree</td>
<td>Single, numeric (non-integer)</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>0</td>
<td>mgaussian</td>
<td>ctree</td>
<td>Multiple, numeric (non-integer)</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>0</td>
<td>binomial</td>
<td>ctree</td>
<td>Single, factor with 2 levels</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>0</td>
<td>multinomial</td>
<td>ctree</td>
<td>Single, factor with &gt;2 levels</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>0</td>
<td>poisson</td>
<td>ctree</td>
<td>Single, integer</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>0</td>
<td>cox</td>
<td>ctree</td>
<td>Object of class ‘Surv’</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>&gt;0</td>
<td>gaussian</td>
<td>ctree</td>
<td>Single, numeric (non-integer)</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>&gt;0</td>
<td>mgaussian</td>
<td>ctree</td>
<td>Multiple, numeric (non-integer)</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>&gt;0</td>
<td>binomial</td>
<td>ctree</td>
<td>Single, factor with 2 levels</td>
</tr>
</tbody>
</table>
If an error along the lines of 'factor ... has new levels ...' is encountered, consult \texttt{?rare_level_sampler} for explanation and solutions.

\textbf{Value}

An object of class \texttt{pre}. It contains the initial ensemble of rules and/or linear terms and a range of possible final ensembles. By default, the final ensemble employed by all other methods and functions in package \texttt{pre} is selected using the 'minimum cross validated error plus 1 standard error' criterion. All functions and methods for objects of class \texttt{pre} take a \texttt{penalty.parameter.val} argument, which can be used to select a different criterion.

\textbf{Note}

Parts of the code for deriving rules from the nodes of trees was copied with permission from an internal function of the \texttt{partykit} package, written by Achim Zeileis and Torsten Hothorn.

\textbf{References}


See Also

print.pre, plot.pre, coef.pre, importance.pre, predict.pre, interact, cvpre

Examples

```r
## Fit pre to a continuous response:
airq <- airquality[complete.cases(airquality), ]
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airq)
airq.ens

## Fit pre to a binary response:
airq2 <- airquality[complete.cases(airquality), ]
airq2$Ozone <- factor(airq2$Ozone > median(airq2$Ozone))
set.seed(42)
airq.ens2 <- pre(Ozone ~ ., data = airq2, family = "binomial")
airq.ens2

## Fit pre to a multivariate continuous response:
airq3 <- airquality[complete.cases(airquality), ]
set.seed(42)
airq.ens3 <- pre(Ozone + Wind ~ ., data = airq3, family = "mgaussian")
airq.ens3

## Fit pre to a multinomial response:
set.seed(42)
iris.ens <- pre(Species ~ ., data = iris, family = "multinomial")
iris.ens

## Fit pre to a survival response:
library("survival")
lung <- lung[complete.cases(lung), ]
set.seed(42)
lung.ens <- pre(Surv(time, status) ~ ., data = lung, family = "cox")
lung.ens

## Fit pre to a count response:
## Generate random data (partly based on Dobson (1990) Page 93: Randomized
## Controlled Trial):
counts <- rep(as.integer(c(18, 17, 15, 20, 10, 20, 25, 13, 12)), times = 10)
outcome <- rep(gl(3, 1, 9), times = 10)
treatment <- rep(gl(3, 3), times = 10)
noise1 <- 1:90
set.seed(1)
noise2 <- rnorm(90)
countdata <- data.frame(treatment, outcome, counts, noise1, noise2)
```
```r
set.seed(42)
count.ens <- pre(counts ~ ., data = countdata, family = "poisson")
count.ens
```

### predict.gpe

**Predicted values based on gpe ensemble**

#### Description

Predict function for `gpe`

#### Usage

```r
## S3 method for class 'gpe'
predict(
  object,
  newdata = NULL,
  type = "link",
  penalty.par.val = "lambda.1se",
  ...)
```

#### Arguments

- `object` of class `gpe`
- `newdata` optional new data to compute predictions for
- `type` argument passed to `predict.cv.glmnet`
- `penalty.par.val` argument passed to `s` argument of `predict.cv.glmnet`
- `...` Unused

#### Details

The initial training data is used if `newdata = NULL`.

#### See Also

- `gpe`
predict.pre

Predicted values based on final prediction rule ensemble

Description

predict.pre generates predictions based on the final prediction rule ensemble, for training or new (test) observations.

Usage

## S3 method for class 'pre'
predict(
  object,
  newdata = NULL,
  type = "link",
  penalty.par.val = "lambda.1se",
  ...
)

Arguments

object          object of class pre.
newdata         optional data.frame of new (test) observations, including all predictor variables used for deriving the prediction rule ensemble.
type            character string. The type of prediction required; the default type = "link" is on the scale of the linear predictors. Alternatively, for count and factor outputs, type = "response" may be specified to obtain the fitted mean and fitted probabilities, respectively; type = "class" returns the predicted class membership.
penalty.par.val character or numeric. Value of the penalty parameter λ to be employed for selecting the final ensemble. The default "lambda.min" employs the λ value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the λ value with minimum cross-validated error, or a numeric value > 0 may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect pre_object$glmnet.fit and plot(pre_object$glmnet.fit).
...

Details

If newdata is not provided, predictions for training data will be returned.

See Also

pre, plot.pre, coef.pre, importance.pre, cvpre, interact, print.pre, predict.cv.glmnet
**print.gpe**

**Print a General Prediction Ensemble (gpe)**

**Description**

Print a General Prediction Ensemble (gpe)

**Usage**

```r
## S3 method for class 'gpe'
print(x, penalty.par.val = "lambda.1se", digits = getOption("digits"), ...)
```

**Arguments**

- `x` An object of class `gpe`.
- `penalty.par.val` character or numeric. Value of the penalty parameter $\lambda$ to be employed for selecting the final ensemble. The default "lambda.min" employs the $\lambda$ value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the $\lambda$ value with minimum cross-validated error, or a numeric value > 0 may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect `pre_object$glmnet.fit$glmnet.fit` and `plot(pre_object$glmnet.fit)`.  
- `digits` Number of decimal places to print
- `...` Further arguments to be passed to `coef.cv.glmnet`.

**See Also**

gpe print.pre
print.pre

Description
print.pre prints information about the generated prediction rule ensemble to the command line.

Usage
## S3 method for class 'pre'
print(x, penalty.par.val = "lambda.1se", digits = getOption("digits"), ...)

Arguments
x
An object of class pre.

penalty.par.val
character or numeric. Value of the penalty parameter $\lambda$ to be employed for selecting the final ensemble. The default "lambda.min" employs the $\lambda$ value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the $\lambda$ value with minimum cross-validated error, or a numeric value $> 0$ may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect pre_object$glmnet.fit and plot(pre_object$glmnet.fit).

digits
Number of decimal places to print.

...
Further arguments to be passed to coef.cv.glmnet.

Details
Note that the CV error is estimated with data that was also used for learning rules and may be too optimistic. Use function cvpre to obtain a more realistic estimate of future prediction error.

Value
Prints information about the fitted prediction rule ensemble.

See Also
pre, summary.pre, plot.pre, coef.pre, importance.pre, predict.pre, interact, cvpre

Examples
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airquality[complete.cases(airquality),])
print(airq.ens)
Dealing with rare factor levels in fitting prediction rule ensembles.

**Description**

Provides a sampling function to be supplied to the sampfrac argument of function pre, making sure that each level of specified factor(s) are present in each sample.

**Usage**

```r
rare_level_sampler(factors, data, sampfrac = 0.5, warning = FALSE)
```

**Arguments**

- **factors**: Character vector with name(s) of factors with rare levels.
- **data**: data.frame containing the variables in the model. Response must be of class factor for classification, numeric for (count) regression, Surv for survival regression. Input variables must be of class numeric, factor or ordered factor. Otherwise, pre will attempt to recode.
- **sampfrac**: numeric value > 0 and ≤ 1. Specifies the fraction of randomly selected training observations used to produce each tree. Values < 1 will result in sampling without replacement (i.e., subsampling), a value of 1 will result in sampling with replacement (i.e., bootstrap sampling). Alternatively, a sampling function may be supplied, which should take arguments n (sample size) and weights.
- **warning**: logical. Whether a warning should be printed if observations with rare factor levels are added to the training sample of the current iteration.

**Details**

Categorical predictor variables (factors) with rare levels may be problematic in boosting algorithms employing sampling (which is employed by default in function pre).

If a sample in a given boosting iteration does not have any observations with a given (rare) level of a factor, while this level is present in the full training dataset, and the factor is selected for splitting in the tree, then no prediction for that level of the factor can be generated, resulting in an error.

Note that boosting methods other than pre that also employ sampling (e.g., gbm or xgboost) may not generate an error in such cases, but also do not document how intermediate predictions are generated in such a case. It is likely that these methods use one-hot-encoding of factors, which from a perspective of model interpretation introduces new problems, especially when the aim is to obtain a sparse set of rules as in ‘pre’.

With function `pre()`, the rare-factor-level issue, if encountered, can be dealt with by the user in one of the following ways (in random order):

- Use a sampling function that guarantees inclusion of rare factor levels in each sample. E.g., use `rare_level_sampler`, yielding a sampling function which creates training samples guaranteed to include each level of specified factor(s). Advantage: No loss of information, easy to
implement, guaranteed to solve the issue. Disadvantage: May result in oversampling of observations with rare factor levels, potentially biasing results. The bias is likely small though, and will be larger for smaller sample sizes and sampling fractions, and for larger numbers of rare levels. The latter will also increase computational demands.

- **Specify learnrate = 0.** This results in a (su)bagging instead of boosting approach. Advantage: Eliminates the rare-factor-level issue completely, because intermediate predictions need not be computed. Disadvantage: Boosting with low learning rate often improves predictive accuracy.

- **Data pre-processing:** Before running function `pre()`, combine rare factor levels with other levels of the factors. Advantage: Limited loss of information. Disadvantage: Likely, but not guaranteed to solve the issue.

- **Data pre-processing:** Apply one-hot encoding to the predictor matrix before applying function ‘pre()’. This can easily be done through applying function `model.matrix`. Advantage: Guaranteed to solve the error, easy to implement. Disadvantage: One-hot-encoding increases the number of predictor variables which may reduce interpretability and, but probably to a lesser extent, accuracy.

- **Data pre-processing:** Remove observations with rare factor levels from the dataset before running function `pre()`. Advantage: Guaranteed to solve the error. Disadvantage: Removing outliers results in a loss of information, and may bias the results.

- **Increase the value of sampfrac argument of function `pre()`**. Advantage: Easy to implement. Disadvantage: Larger samples are more likely but not guaranteed to contain all possible factor levels, thus not guaranteed to solve the issue.

### Value

A sampling function, which generates sub- or bootstrap samples as usual in function `pre`, but checks if all levels of the specified factor(s) are present and adds observation with those levels if not. If `warning = TRUE`, a warning is issued).

### See Also

`pre`

### Examples

```r
## Create dataset with two factors containing rare levels
dat <- iris[iris$Species != "versicolor",]
dat <- rbind(dat, iris[iris$Species == "versicolor",][1:5,])
dat$factor2 <- factor(rep(1:21, times = 5))

## Set up sampling function
samp_func <- rare_level_sampler(c("Species", "factor2"), data = dat,
sampfrac = .51, warning = TRUE)

## Illustrate behavior of sampling function
N <- nrow(dat)
wts <- rep(1, times = nrow(dat))
set.seed(3)
dat[samp_func(n = N, weights = wts),] # single sample
```
for (i in 1:500) dat[samp_func(n = N, weights = wts), ]
warnings() # to illustrates warnings that may occur when fitting a full PRE

## Illustrate use of function generator with function pre:
## (Note: low ntrees value merely to reduce computation time for the example)
set.seed(42)
# iris.ens <- pre(Petal.Width ~ . , data = dat, ntrees = 20) # would yield error
iris.ens <- pre(Petal.Width ~ . , data = dat, ntrees = 20,
sampfrac = samp_func) # should work

---

### Description

Wrapper functions for terms in gpe.

### Usage

rTerm(x)

lTerm(x, lb = -Inf, ub = Inf, scale = 1/0.4)

eTerm(x, scale = 1/0.4)

### Arguments

- **x**: Input symbol.
- **lb**: Lower quantile when winsorizing. -Inf yields no winsorizing in the lower tail.
- **ub**: Lower quantile when winsorizing. Inf yields no winsorizing in the upper tail.
- **scale**: Inverse value to time \( x \) by. Usually the standard deviation is used. \( 0.4 / \text{scale} \) is used as the multiplier as suggested in Friedman & Popescu (2008) and gives each linear term the same a-priori influence as a typical rule.

### Details

The motivation to use wrappers is to ease getting the different terms as shown in the examples and to simplify the formula passed to `cv.glmnet` in gpe. `lTerm` potentially rescales and/or winsorizes \( x \) depending on the input. `eTerm` potentially rescale \( x \) depending on the input.

### Value

\( x \) potentially transformed with additional information provided in the attributes.

### References

See Also
gpe, gpe_trees gpe_linear gpe_earth

Examples

```r
t <- terms(~ rTerm(x1 < 0) + rTerm(x2 > 0) + lTerm(x3) + eTerm(x4),
  specials = c("rTerm", "lTerm", "eTerm"))
attr(t, "specials")
# $rTerm
# [1] 1 2
#
# $lTerm
# [1] 3
#
# $eTerm
# [1] 4
```

---

**singleplot**

Create partial dependence plot for a single variable in a prediction rule ensemble (pre)

**Description**

singleplot creates a partial dependence plot, which shows the effect of a predictor variable on the ensemble’s predictions. Note that plotting partial dependence is computationally intensive. Computation time will increase fast with increasing numbers of observations and variables. For large datasets, package ‘plotmo’ (Milborrow, 2019) provides more efficient functions for plotting partial dependence and also supports ‘pre’ models.

**Usage**

```r
singleplot(
  object,
  varname,
  penalty.par.val = "lambda.1se",
  nvals = NULL,
  type = "response",
  ylab = "predicted",
  gamma = NULL,
  ...
)
```
Arguments

object an object of class `pre`.

varname character vector of length one, specifying the variable for which the partial dependence plot should be created. Note that `varname` should correspond to the variable as described in the model formula used to generate the ensemble (i.e., including functions applied to the variable).

penalty.par.val character or numeric. Value of the penalty parameter $\lambda$ to be employed for selecting the final ensemble. The default "lambda.min" employs the $\lambda$ value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the $\lambda$ value with minimum cross-validated error, or a numeric value $> 0$ may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect `pre_object$glmnet.fit` and `plot(pre_object$glmnet.fit)`.

nvals optional numeric vector of length one. For how many values of x should the partial dependence plot be created?

type character string. Type of prediction to be plotted on y-axis. `type = "response"` gives fitted values for continuous outputs and fitted probabilities for nominal outputs. `type = "link"` gives fitted values for continuous outputs and linear predictor values for nominal outputs.

ylab character. Label to be printed on the y-axis.

gamma Mixing parameter for relaxed fits. See `coef.cv.glmnet`.

... Further arguments to be passed to `plot.default`.

Details

By default, a partial dependence plot will be created for each unique observed value of the specified predictor variable. When the number of unique observed values is large, this may take a long time to compute. In that case, specifying the `nvals` argument can substantially reduce computing time. When the `nvals` argument is supplied, values for the minimum, maximum, and $(nvals - 2)$ intermediate values of the predictor variable will be plotted. Note that `nvals` can be specified only for numeric and ordered input variables. If the plot is requested for a nominal input variable, the `nvals` argument will be ignored and a warning printed.

See also section 8.1 of Friedman & Popescu (2008).

References


See Also

`pre`, `pairplot`
summary.gpe

Summary method for a General Prediction Ensemble (gpe)

Description

summary.gpe prints information about the generated ensemble to the command line

Usage

## S3 method for class 'gpe'
summary(object, penalty.par.val = "lambda.1se", ...)

Arguments

object
An object of class gpe.

penalty.par.val
character or numeric. Value of the penalty parameter $\lambda$ to be employed for selecting the final ensemble. The default "lambda.min" employs the $\lambda$ value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the $\lambda$ value with minimum cross-validated error, or a numeric value > 0 may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect pre_object$glmnet.fit and plot(pre_object$glmnet.fit).

... Further arguments to be passed to coef.cv.glmnet.

Details

Note that the cv error is estimated with data that was also used for learning rules and may be too optimistic.

Value

Prints information about the fitted ensemble.

See Also

gpe, print.gpe, coef.gpe, predict.gpe
Summary method for objects of class pre

Description

summary.pre prints information about the generated prediction rule ensemble to the command line.

Usage

```r
## S3 method for class 'pre'
summary(
  object,
  penalty.par.val = "lambda.1se",
  digits =getOption("digits"),
  ...
)
```

Arguments

- `object`: An object of class `pre`.
- `penalty.par.val`: character or numeric. Value of the penalty parameter \( \lambda \) to be employed for selecting the final ensemble. The default "lambda.min" employs the \( \lambda \) value within 1 standard error of the minimum cross-validated error. Alternatively, "lambda.min" may be specified, to employ the \( \lambda \) value with minimum cross-validated error, or a numeric value \( > 0 \) may be specified, with higher values yielding a sparser ensemble. To evaluate the trade-off between accuracy and sparsity of the final ensemble, inspect `pre_object(glmnet.fit)` and `plot(pre_object$glmnet.fit)`.
- `digits`: Number of decimal places to print
- `...`: Further arguments to be passed to `coef.cv.glmnet`.

Details

Note that the cv error is estimated with data that was also used for learning rules and may be too optimistic. Use `cvpre` to obtain a more realistic estimate of future prediction error.

Value

Prints information about the fitted prediction rule ensemble.

See Also

`pre`, `print.pre`, `plot.pre`, `coef.pre`, `importance.pre`, `predict.pre`, `interact`, `cvpre`
Examples

```r
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airquality[complete.cases(airquality),])
summary(airq.ens)
```
Index

* datasets
  caret_pre_model, 4
carrillo, 4
bsnullinteract, 2, 24
caret_pre_model, 4
carrillo, 4
coef.cv.glmnet, 6, 20, 22, 26, 28, 37, 38, 43–45
coef.gpe, 5, 44
coef.pre, 6, 10, 12, 34, 36, 38, 45
contour, 27
corplot, 7
tree, 16, 18, 30
tree_control, 16, 18, 31
cv.glmnet, 13–15, 18, 32, 41
cvpred, 7, 8, 12, 34, 36, 38, 45
earth, 18
eTerm, 19
eTerm(rTerm), 41
explain, 10

family, 30
glm.tree, 16, 18, 30
gpar, 29
gpe, 5, 6, 13, 14–17, 19, 35, 37, 41, 42, 44
gpe_cv.glmnet, 13, 14, 14
gpe_earth, 13, 14, 42
gpe_earth (gpe_trees), 17
gpe_linear, 13, 14, 42
gpe_linear (gpe_trees), 17
gpe_rules_pre, 15
gpe_sample, 13, 14, 16
gpe_trees, 13, 14, 17, 42
image, 27
importance (importance.pre), 19

importance.pre, 7, 10, 12, 19, 29, 34, 36, 38, 45
interact, 2, 3, 7, 10, 12, 22, 34, 36, 38, 45
Iterm, 19
Iterm(rTerm), 41
maxdepth_sampler, 16, 24, 31
mob_control, 16, 31
model.matrix, 40

pairplot, 26, 43
par, 8
persp, 27
plot.default, 43
plot.pre, 7, 10, 12, 28, 34, 36, 38, 45
pre, 3, 6, 7, 9, 10, 12, 14, 15, 20–22, 24–29, 39, 36, 38, 40, 43, 45
predict.cv.glmnet, 11, 35, 36
predict.gpe, 35, 44
predict.pre, 3, 7, 9–11, 34, 36, 38, 45
print.gpe, 37, 44
print.pre, 7, 10, 12, 29, 34, 36, 37, 38, 45

rare_level_sampler, 39
rpart, 16, 30, 32
rpart.control, 16, 31
rTerm, 19, 41

singleplot, 27, 42
summary.gpe, 44
summary.pre, 38, 45