Package: pre (via r-universe)

June 9, 2026

Title Prediction Rule Ensembles

Version 1.0.9

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 described in Fokkema (2020; <DOI:10.18637/jss.v092.i12>) and Fokkema & Strobl (2020; <DOI:10.1037/met0000256>). The main function pre() derives prediction rule ensembles consisting of rules and/or linear terms for continuous, binary, count, multinomial, survival 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 (>= 4.1.0)

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, shape, randomForest

License GPL-2 | GPL-3

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Repository https://cran.r-universe.dev

Date/Publication 2026-06-09 09:40:02 UTC

RemoteUrl https://github.com/cran/pre

RemoteRef HEAD

RemoteSha 74a5498c555bc6507bc9b1731b0df1d6608edb49

Contents

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",
  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 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).
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., ≥ 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

Fokkema, M. (2020). Fitting prediction rule ensembles with R package pre. *Journal of Statistical Software, 92*(12), 1-30. doi:10.18637/jss.v092.i12

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954, doi:10.1214/07AOAS148.

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"))
```

---

caret_pre_model *Model set up for train function of package caret*

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
```

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

Beck, A.T., Steer, R.A. & Carbin, M.G. (1988). Psychometric properties of the Beck Depression Inventory: Twenty-five years of evaluation. *Clinical Psychology Review, 8*(1), 77-100.

Carrillo, J. M., Rojo, N., Sanchez-Bernardos, M. L., & Avia, M. D. (2001). Openness to experience and depression. *European Journal of Psychological Assessment, 17*(2), 130.

Costa, P.T. & McCrae, R.R. (1985). *The NEO Personality Inventory.* Psychological Assessment Resources, Odessa, FL.

Examples

```
data("carrillo")
summary(carrillo)
```

coef.gpe *Coefficients for a General Prediction Ensemble (gpe)*

Description

coef function for gpe

Usage

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

Arguments

object	object of class pre
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).
...	Further arguments to be passed to coef.cv.glmnet.

See Also

coef.pre, gpe

coef.pre *Coefficients for the final prediction rule ensemble*

Description

coef.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	object of class pre
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).
...	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 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)
```

---

corplot *Plot correlations between baselearners in a prediction rule ensemble (pre)*

Description

corplot plots correlations between baselearners in a prediction rule ensemble

Usage

```
corplot(
  object,
  penalty.par.val = "lambda.1se",
  colors = NULL,
  fig.plot = c(0, 0.85, 0, 1),
  fig.legend = c(0.8, 0.95, 0, 1),
  legend.breaks = seq(-1, 1, by = 0.1)
)
```

Arguments

object	object of class pre
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).
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

```
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

```
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 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)`.
`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 (`$cvpreds`). Invisibly, a list of three objects is returned: `accuracy` (containing accuracy estimates), `cvpreds` (containing cross-validated predictions) and `fold_indicators` (a vector indicating the cross validation fold each observation was part of). For (multivariate) continuous outcomes, accuracy is a list with elements `$MSE` (mean squared error on test observations) and `$MAE` (mean absolute error on test observations). For (binary and multiclass) classification, accuracy is a list with elements `$SEL` (mean squared error on predicted probabilities), `$AEL` (mean absolute error on predicted probabilities), `$MCR` (average misclassification error rate) and `$table` (proportion table with (mis)classification rates).

See Also

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

Examples

```
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,
  newdata,
  penalty.par.val = "lambda.1se",
  response = 1L,
  plot = TRUE,
  intercept = FALSE,
  center.linear = FALSE,
  plot.max.nobs = 4,
  plot.dim = c(2, 2),
  plot.obs.names = TRUE,
  pred.type = "response",
  digits = 3L,
  cex = 0.8,
```

```
  ylab = "Contribution to linear predictor",
  bar.col = c("#E495A5", "#39BEB1"),
  rule.col = "darkgrey",
  ...
)
```

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 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).
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 forpres 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 depcting 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 horizonantal (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.

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 obeservation'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 temr, minus the mean value of the linear term in the training data) times the estimated coefficient for the linear term.

References

Fokkema, M. & Strobl, C. (2020). Fitting prediction rule ensembles to psychological research data: An introduction and tutorial. *Psychological Methods* *25*(5), 636-652. doi:10.1037/met0000256, https://arxiv.org/abs/1907.05302

See Also

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

Examples

```
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")$contribution
## After mean centering, contributions of Month and Day have switched
##   sign (for these two observations):
explain(airq.ens2, newdata = airq[-train,][1:2,],
        penalty.par.val = "lambda.min", center.linear = TRUE)$contribution
```

---

gpe *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 customs 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

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954. Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics, 19*(1), 1-67.

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)
```

gpe_cv.glmnet *Default penalized trainer for gpe*

Description

Default "penalizer function" generator gpe which uses cv.glmnet.

Usage

```
gpe_cv.glmnet(...)
```

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

gpe_rules_pre *Get rule learner for gpe which mimics behavior of pre*

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	a list with control parameters to be passed to the tree fitting function, generated using ctree_control, mob_control (if use.grad = FALSE), rpart.control (if tree.unbiased = FALSE). Or a list containing (n)one or more arguments that can be passed to function randomForest (if randomForest = TRUE.
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)
```

Arguments

sampfrac	Fraction of n to use for sampling. It is the η/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

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954.

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 ν 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.

References

Hothorn, T., & Zeileis, A. (2015). partykit: A modular toolkit for recursive partytioning in R. *Journal of Machine Learning Research, 16*, 3905-3909.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals Statistics, 19*(1), 1-67.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Applied Statistics, 29*(5), 1189-1232.

Friedman, J. H. (1993). Fast MARS. Dept. of Statistics Technical Report No. 110, Stanford University.

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954.

Chen T., & Guestrin C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.

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 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).
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 & Popecus (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

Fokkema, M. (2020). Fitting prediction rule ensembles with R package pre. *Journal of Statistical Software, 92*(12), 1-30. doi:10.18637/jss.v092.i12

Fokkema, M. & Strobl, C. (2020). Fitting prediction rule ensembles to psychological research data: An introduction and tutorial. *Psychological Methods 25*(5), 636-652. doi:10.1037/met0000256, https://arxiv.org/abs/1907.05302

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954 doi:10.1214/07AOAS148.

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

```
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-zeor coefficients will be calculated (which may take a long time).

`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)`.

`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

Fokkema, M. (2020). Fitting prediction rule ensembles with R package pre. *Journal of Statistical Software, 92*(12), 1-30. doi:10.18637/jss.v092.i12

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954, doi:10.1214/07AOAS148.

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 inducation.

`av.tree.depth`
: integer of length one. Specifies the average maximum tree depth in trees used for rule induction.

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 orignal 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 `ntrees`, which can be supplied to the `maxdepth` argument of function `pre` to specify varying tree depths.

References

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954.

See Also

`pre`

Examples

```
## 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(av.no.term.nodes = 16L)
set.seed(42)
func2(10)
mean(func2(1000))

## Max. tree depth of 3, on average:
func3 <- maxdepth_sampler(av.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(av.no.term.nodes = 2L)
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),],
                maxdepth = func1)
airq.ens
```

`mi_mean` *Compute the average dataset over imputed datasets.*

Description

`mi_mean` computes the averages dataset over a list of imputed datasets. Can be used to reduce computation time of functions `singleplot` and `pairplot`.

Usage

```
mi_mean(data)
```

Arguments

`data`	List of imputed datasets to compute the average dataset over.

Details

It is assumed every imputed dataset contains the same observations (but not the same values) in the same order.

Value

A dataset that is the average over the imputed datasets specified with `data`. For continuous predictors, the mean over imputed values is returned, for categorical predictors, the majority class ovder imputed values is returned. In case of a non-unique maximum, the value is sampled from the class with identical maximum counts.

See Also

`mi_pre`, `singleplot`, `pairplot`

mi_pre *Fit a prediction rule ensemble to multiply-imputed data (experimental)*

Description

Function mi_pre derives a sparse ensemble of rules and/or linear rules based on imputed data. The function is still experimental, so use at own risk.

Usage

```
mi_pre(
  formula,
  data,
  weights = NULL,
  obs_ids = NULL,
  compl_frac = NULL,
  nfolds = 10L,
  sampfrac = 0.5,
  ...
)
```

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	A list of imputed datasets. The datasets must have identically-named columns, but need not have the same number of rows (this can happen, for example. if a bootstrap sampling approach had been employed for multiple imputation).
weights	A list of observation weights for each observation in each imputed dataset. The list must have the same length as data, and each element must be a numeric vector of length identical to the number of rows of the corresponding imputed dataset in data. The default is NULL, yielding constant observation weights w_i = 1/M, where M is the number of imputed datasets (i.e., length(data)).
obs_ids	A list of observation ids, corresponding to the id in the original data, of each observation in each imputed dataset. Defaults to NULL, which assumes that the imputed datasets contain the observations in identical order. If specified, the list must have the same length as data, and each element must be a numeric or character vector of length identical to the number of rows of the corresponding imputed dataset in data. At least some of the observations ids must be repeated at least some times, within or between imputed datasets.

`compl_frac` An optional list specifying the fraction of observed values for each observation. This will be used to compute observation weights as a function of the fraction of complete data per observations, as per Wan et al. (2015), but note that this is only recommended for users who know the risks (i.e., an analysis more like complete-case analysis). If specified, the list must have the same length as `data`, and each element must be a numeric vector of length identical to the number of rows of the corresponding imputed dataset in `data`.

`nfolds` positive integer. Number of cross-validation folds to be used for selecting the optimal value of the penalty parameter λ in selecting the final ensemble.

`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`.

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

Details

Experimental function to fit a prediction rule ensemble to multiply imputed data. Essentially, it is a wrapper function around function `pre()`, the main differences relate to sampling for the tree induction and fold assignment for estimation of the coefficients for the final ensemble.

Function `mi_pre` implements a so-called stacking approach to the analysis of imputed data (see also Wood et al., 2008), where imputed datasets are combined into one large dataset. In addition to adjustments of the sampling procedures, adjustments to observation weight are made to counter the artificial inflation of sample size.

Observations which occur repeatedly across the imputed datasets will be completely in- or excluded from each sample or fold, to avoid overfitting. Thus, complete observations instead of individual imputed observations are sampled, for tree and rule induction, as well as the cross-validation for selecting the penalty parameter values for the final ensemble.

It is assumed that data have already been imputed (using e.g., R package mice or missForest), and therefore function `mi_pre` takes a `list` of imputed datasets as input data.

Although the option to use the fraction of complete data for computing observation weight is provided through argument `compl_frac`, users are not advised to use it. See e.g., Du et al. (2022): "An alternative weight specification, proposed in Wan et al. (2015), is o_i = f_i/D, where f_i is the number of observed predictors out of the total number of predictors for subject i [...] upweighting subjects with less missingness and downweighting subjects with more missingness can, in some sense, be viewed as making the optimization more like complete-case analysis, which might be problematic for Missing at Random (MAR) and Missing not at Random (MNAR) scenarios."

Value

An object of class `pre`.

References

Du, J., Boss, J., Han, P., Beesley, L. J., Kleinsasser, M., Goutman, S.A., ... & Mukherjee, B. (2022). Variable selection with multiply-imputed datasets: choosing between stacked and grouped

methods. Journal of Computational and Graphical Statistics, 31(4), 1063-1075. doi:10.1080/10618600.2022.2035739.

Wood, A. M., White, I. R., & Royston, P. (2008). How should variable selection be performed with multiply imputed data? Statistics in medicine, 27(17), 3227-3246. doi:10.1002/sim.3177

See Also

pre mi_mean

Examples

```
library("mice")
set.seed(42)

## Shoot extra holes in airquality data
airq <- sapply(airquality, function(x) {
  x[sample(1:nrow(airquality), size = 25)] <- NA
  return(x)
})

## impute the data
imp <- mice(airq, m = 5)
imp <- as.list(complete(imp, action = "all"))

## fit a rule ensemble to the imputed data
set.seed(42)
airq.ens.mi <- mi_pre(Ozone ~ . , data = imp)
```

---

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",
```

```
  response = NULL,
  nvals = c(20L, 20L),
  pred.type = "response",
  newdata = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  rug = TRUE,
  ...
)
```

Arguments

object	an object of class pre
varnames	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 "factor", an error will be printed.
type	character string. Type of plot to be generated. type = "heatmap" yields a heatmap plot, type = "contour" yields a contour plot, type = "both" yields a heatmap plot with added contours, type = "perspective" yields a three dimensional plot.
gamma	Mixing parameter for relaxed fits. See coef.cv.glmnet.
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).
response	numeric vector of length 1. Only relevant for multivariate gaussian and multinomial responses. If NULL (default), PDPs for all response variables or categories will be produced. A single integer can be specified, indicating for which response variable or category PDPs should be produced.
nvals	optional numeric vector of length 2. For how many values of x1 and x2 should partial dependence be plotted? If NULL, a grid of all possible combinations of the observed values of 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.
newdata	Optional data.frame in which to look for variables with which to predict. If NULL, the data.frame used to fit the original ensemble will be used.
xlab	character. Label to be printed on the x-axis. If NULL, the first elements of the supplied varnames will be printed on the x-axis.

ylab	character. Label to be printed on the y-axis. If NULL, the second element of the supplied varnames will be printed on the y-axis.
main	character vector. Title(s) for the plot. If NULL, the name of the response will be printed.
rug	logical. Ignored if type = "perspective". Should a rug be plotted on the x- and y-axes, representing the location of observed datapoint? Note that the rugs will only show where values have been observed, not their frequency/density.
...	Further arguments to be passed to image, contour or persp (depending on whether type is specified to be "heatmap", "contour", "both" or "perspective").

Details

Partial dependence functions are described in section 8.1 of Friedman & Popescu (2008).

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 specified predictor variables. If NA instead of a numeric value is specified for one of the predictor variables, all observed values for that variables will be used. Specifying nvals = NULL and nvals = c(NA, NA) will yield the exact same result.

High values, NA or NULL for nvals result in long computation times and possibly memory problems. 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 numeric value(s) are specified for nvals, values for the minimum, maximum, and nvals - 2 intermediate values of the predictor variable will be plotted.

Alternatively, newdata can be specified to provide a different (smaller) set of observations to compute partial dependence over. If mi_pre was used to derive the original rule ensemble, newdata = "mean.mi" can be specified. This will result in an average dataset being computed over the imputed datasets, which are then used to compute partial dependence functions. This greatly reduces the number of observations and thereby computation time.

If none of the variables specified with argument varnames was selected for the final prediction rule ensemble, an error will be returned.

References

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954.

Milborrow, S. (2019). plotmo: Plot a model's residuals, response, and partial dependence plots. https://CRAN.R-project.org/package=plotmo

See Also

pre, singleplot

Examples

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

## For multinomial and mgaussian families, one PDP is created per category or outcome
set.seed(42)
airq.ens3 <- pre(Ozone + Wind ~ ., data = airq, family = "mgaussian")
pairplot(airq.ens3, varnames = c("Day", "Month"))

set.seed(42)
iris.ens <- pre(Species ~ ., data = iris, family = "multinomial")
pairplot(iris.ens, varname = c("Petal.Width", "Petal.Length"))
```

---

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 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).
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.
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

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

---

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,
  ad.alpha = NA,
  ad.penalty = "lambda.min",
  use.grad = TRUE,
  weights,
  type = "both",
  sampfrac = 0.5,
  maxdepth = 3L,
  learnrate = 0.01,
  mtry = Inf,
  ntrees = 500,
  confirmatory = NULL,
  singleconditions = FALSE,
  randomForest = FALSE,
  winsfrac = 0.025,
  normalize = TRUE,
  standardize = FALSE,
  ordinal = TRUE,
  nfolds = 10L,
  tree.control,
  tree.unbiased = TRUE,
  removecomplements = 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, the program will try to make an informed guess, based on the class of the response variable specified in formula. Also see Examples below.

ad.alpha — Alpha value to be used for computing the penalty weights for the adaptive lasso. Defaults to NA, yielding standard lasso estimation. To use adaptive lasso, specify a value between (and including) 0 and 1. A value of 0 will yield ridge-estimated penalty weights for computing the final (lasso) penalized model. See vignette("relaxed", "pre") or argument alpha of cv.glmnet.

ad.penalty — Penalty parameter value to be used for computing the penalty weights for the adaptive lasso. Defaults to "lambda.min". If OLS instead of elastic net regression should be used to compute weights, specify ad.penalty = 0. See also vignette("relaxed", "pre").

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 fitting 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 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.

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 tree.unbiased=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.

randomForest
: use function randomForest of package of the same name for rule induction? If set to TRUE, arguments weights, ntree, mtry and maxdepth will be passed to randomForest. Argument learnrate will be fixed to 0 and sampfrac will be ignored (preferences about sampling procedures should be passed to tree.control when using randomForest) which also allows to specify other arguments of function randomForest.

winsfrac
: numeric value > 0 and ≤ 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 dummified 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 λ in selecting the final ensemble.

tree.control
: a list with control parameters to be passed to the tree fitting function, generated using ctree_control, mob_control (if use.grad = FALSE), rpart.control (if tree.unbiased = FALSE). Or a list containing (n)one or more arguments that can be passed to function randomForest (if randomForest = TRUE.

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 that observations with missing values will be removed prior to analysis (and a warning printed). See vignette("missingness", "pre") an mi_pre for suggestions on how to better deal with missing values.

In some cases, duplicated variable names may occur in the final ensemble 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:

use.grad	tree.unbiased	learnrate	family	tree alg.	Response variable format
TRUE	TRUE	0	gaussian	ctree	Single, numeric (non-integer)
TRUE	TRUE	0	mgaussian	ctree	Multiple, numeric (non-integer)
TRUE	TRUE	0	binomial	ctree	Single, factor with 2 levels
TRUE	TRUE	0	multinomial	ctree	Single, factor with >2 levels
TRUE	TRUE	0	poisson	ctree	Single, integer
TRUE	TRUE	0	cox	ctree	Object of class 'Surv'
TRUE	TRUE	>0	gaussian	ctree	Single, numeric (non-integer)
TRUE	TRUE	>0	mgaussian	ctree	Multiple, numeric (non-integer)
TRUE	TRUE	>0	binomial	ctree	Single, factor with 2 levels
TRUE	TRUE	>0	multinomial	ctree	Single, factor with >2 levels
TRUE	TRUE	>0	poisson	ctree	Single, integer
TRUE	TRUE	>0	cox	ctree	Object of class 'Surv'
FALSE	TRUE	0	gaussian	glmtree	Single, numeric (non-integer)
FALSE	TRUE	0	binomial	glmtree	Single, factor with 2 levels
FALSE	TRUE	0	poisson	glmtree	Single, integer

FALSE	TRUE	>0	gaussian	glmtree	Single, numeric (non-integer)
FALSE	TRUE	>0	binomial	glmtree	Single, factor with 2 levels
FALSE	TRUE	>0	poisson	glmtree	Single, integer
TRUE	FALSE	0	gaussian	rpart	Single, numeric (non-integer)
TRUE	FALSE	0	binomial	rpart	Single, factor with 2 levels
TRUE	FALSE	0	multinomial	rpart	Single, factor with >2 levels
TRUE	FALSE	0	poisson	rpart	Single, integer
TRUE	FALSE	0	cox	rpart	Object of class 'Surv'
TRUE	FALSE	>0	gaussian	rpart	Single, numeric (non-integer)
TRUE	FALSE	>0	binomial	rpart	Single, factor with 2 levels
TRUE	FALSE	>0	poisson	rpart	Single, integer
TRUE	FALSE	>0	cox	rpart	Object of class 'Surv'

Alternatively, specifying `randomForest = TRUE` overrides all of these arguments and employs function `randomForest` from the package of the same name for rule induction. All arguments of function `randomForest` should be passed using the `tree.control` argument, except for arguments `formula`, `data`, `mtry` and `ntrees`, because the other arguments of function `randomForest` do not have identical counterparts in function `pre`.

If an error along the lines of 'factor ... has new levels ...' is encountered, consult `?rare_level_sampler` for explanation and solutions.

Value

An object of class `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 `pre` is selected using the 'minimum cross validated error plus 1 standard error' criterion. All functions and methods for objects of class `pre` take a `penalty.parameter.val` argument, which can be used to select a different criterion.

If only a set of rules needs to be generated, but the final regression model should not be fitted, specify the hidden argument `fit.final = FALSE`.

Note

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

References

Fokkema, M. (2020). Fitting prediction rule ensembles with R package pre. *Journal of Statistical Software, 92*(12), 1-30. doi:10.18637/jss.v092.i12

Fokkema, M. & Strobl, C. (2020). Fitting prediction rule ensembles to psychological research data: An introduction and tutorial. *Psychological Methods 25*(5), 636-652. doi:10.1037/met0000256, https://arxiv.org/abs/1907.05302

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Applied Statistics, 29*(5), 1189-1232.

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954, doi:10.1214/07AOAS148.

Hothorn, T., & Zeileis, A. (2015). partykit: A modular toolkit for recursive partytioning in R. *Journal of Machine Learning Research, 16*, 3905-3909.

See Also

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

Examples

```
## Fit pre to a continuous response:
airq <- airquality[complete.cases(airquality), ]
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airq)
airq.ens
## Use relaxed lasso to estimate final model
airq.ens.rel <- pre(Ozone ~ ., data = airq, relax = TRUE)
airq.ens.rel
## Use adaptive lasso to estimate final model
airq.ens.ad <- pre(Ozone ~ ., data = airq, ad.alpha = 0)
airq.ens.ad

## 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)
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

```
## 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).
...	further arguments to be passed to predict.cv.glmnet.

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

Examples

```
set.seed(1)
train <- sample(1:sum(complete.cases(airquality)), size = 100)
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airquality[complete.cases(airquality),][train,])
predict(airq.ens)
predict(airq.ens, newdata = airquality[complete.cases(airquality),][-train,])
```

`print.gpe` *Print a General Prediction Ensemble (gpe)*

Description

Print a General Prediction Ensemble (gpe)

Usage

```
## 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 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)`.
`digits`	Number of decimal places to print
`...`	Further arguments to be passed to `coef.cv.glmnet`.

See Also

`gpe` `print.pre`

print.pre *Print method for objects of class 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 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).
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)
```

prune_pre *Get the optimal lambda and gamma parameter values for an ensemble of given size*

Description

Function prune_pre returns the optimal values of lambda and gamma for the requested ensemble size.

Usage

```
prune_pre(object, nonzero, plusminus = 3)
```

Arguments

object	an object of class pre that was fit using the relaxed lasso. If an object of class pre is specified that was not fit using the relaxed lasso, an error will be printed.
nonzero	maximum number of terms to retain.
plusminus	number of terms above and below nonzero for which CV results will be printed.

Value

The lambda and gamma values that yield optimal predictive accuracy for the specified number of terms. These are invisibly returned, see Examples on how to use them. A sentence describing what the optimal values are is printed to the command line, with an overview of the performance (in terms of cross-validated accuracy and the number of terms retained) of lambda values near the optimum. If the specified number of terms to retain is lower than what would be obtained using the lambda.min or lambda.1se criterion, a warning will also be printed.

See Also

pre

Examples

```
## Fit a rule ensemble to predict Ozone concentration
airq <- airquality[complete.cases(airquality), ]
set.seed(42)
airq.ens <- pre(Ozone ~ ., data = airq, relax = TRUE)

## Inspect the result (default lambda.1se criterion)
airq.ens

## Inspect the lambda path
## (lower x-axis gives lambda values, upper x-axis corresponding no. of non-zero terms)
## Not run: plot(airq.ens$glmnet.fit)

## Accuracy still quite good with only 5 terms, obtain corresponding parameter values
```

```
opt_pars <- prune_pre(airq.ens, nonzero = 5)
opt_pars

## Use the parameter values for interpretation and prediction, e.g.
predict(airq.ens, newdat = airq[c(22, 33), ], penalty = opt_pars$lambda, gamma = opt_pars$gamma)
summary(airq.ens, penalty = opt_pars$lambda, gamma = opt_pars$gamma)
print(airq.ens, penalty = opt_pars$lambda, gamma = opt_pars$gamma)
```

`rare_level_sampler` *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

```
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

```
## 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 what it does
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 illustrate warnings that may occur when fitting a full PRE

## Illustrate use 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
```

`rTerm` *Wrapper Functions for terms in gpe*

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 / 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

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954.

See Also

gpe, gpe_trees gpe_linear gpe_earth

Examples

```
mt <- terms(
~ rTerm(x1 < 0) + rTerm(x2 > 0) + lTerm(x3) + eTerm(x4),
specials = c("rTerm", "lTerm", "eTerm"))
attr(mt, "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

```
singleplot(
  object,
  varname,
  penalty.par.val = "lambda.1se",
  nvals = NULL,
  type = "response",
```

```
  ylab = NULL,
  response = NULL,
  gamma = NULL,
  newdata = NULL,
  xlab = NULL,
  rug = TRUE,
  ...
)
```

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 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).
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, defaults to the response variable name(s).
response	numeric vector of length 1. Only relevant for multivariate gaussian and multinomial responses. If NULL (default), PDPs for all response variables or categories will be produced. A single integer can be specified, indicating for which response variable or category PDPs should be produced.
gamma	Mixing parameter for relaxed fits. See coef.cv.glmnet.
newdata	Optional data.frame in which to look for variables with which to predict. If NULL (the default), the data.frame used to fit the original ensemble will be used. Smaller subsets of the original data can be specified to (substantially) reduce computation time. See Details.
xlab	character. Label to be printed on the x-axis. If NULL, the supplied varname will be printed on the x-axis.
rug	logical. Should a rug be plotted on the x-axis, representing the location of observed datapoint? Note that the rug will only show where values have been observed, not their frequency/density.
...	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. See also section 8.1 of Friedman & Popescu (2008).

When the number of unique observed values is large, partial dependence functions can take a very long time to compute. Specifying the `nvals` argument can substantially reduce computation 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.

Alternatively, `newdata` can be specified to provide a different (smaller) set of observations to compute partial dependence over. If `mi_pre` was used to derive the original rule ensemble, function `mean_mi` can be used for this.

Value

A 1D partial dependence plot will be plotted. Invisibly, partial dependence values are returned.

References

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954.

Milborrow, S. (2019). plotmo: Plot a model's residuals, response, and partial dependence plots. https://CRAN.R-project.org/package=plotmo

See Also

`pre`, `pairplot`

Examples

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

## For multinomial and mgaussian families, one PDP is created per category or outcome
set.seed(42)
airq.ens3 <- pre(Ozone + Wind ~ ., data = airq, family = "mgaussian")
singleplot(airq.ens3, varname = "Day")

set.seed(42)
iris.ens <- pre(Species ~ ., data = iris, family = "multinomial")
singleplot(iris.ens, varname = "Petal.Width")
```

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 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).
...	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.pre *Summary method for objects of class pre*

Description

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

Usage

```
## 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 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).
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

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

Index