Package 'pre'

April 1, 2019

Title Prediction Rule Ensembles

Version 0.7.0

Author Marjolein Fokkema [aut, cre],
Benjamin Christoffersen [aut]

Maintainer Marjolein Fokkema <m.fokkema@fsw.leidenuniv.nl>

Description Derives prediction rule ensembles (PREs). Largely follows the procedure for deriving PREs as described in Friedman & Popescu (2008; <DOI:10.1214/07-AOAS148>), with adjustments and improvements. The main function pre() derives prediction rule ensembles consisting of rules and/or linear terms for continuous, binary, count, multinomial, and multivariate continuous responses. Function gpe() derives generalized prediction ensembles, consisting of rules, hinge and linear functions of the predictor variables.

URL https://github.com/marjoleinF/pre

BugReports https://github.com/marjoleinF/pre/issues

Depends R (>= 3.1.0)

Imports earth, Formula, glmnet, graphics, methods, partykit (>= 1.2-0), rpart, stringr, survival, Matrix, MatrixModels

Suggests akima, datasets, doParallel, foreach, glmertree, grid, mlbench, testthat, mboost

License GPL-2 | GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-04-01 12:20:03 UTC

R topics documented:

`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?

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. (2018). Fitting prediction rule ensembles with R package pre. https://arxiv.org/abs/1707.07149.

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

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 provides a model setup for the train function of package caret

Usage

```
caret_pre_model
```

Format

An object of class list of length 17.

Details

This is still somewhat experimental. Function pre will become availabel as a method in package caret in future versions, after additional testing and finetuning; caret_pre_model will then become depracated.

Examples

```
## Not run:

library("caret")

## Prepare data:
airq <- airquality[complete.cases(airquality),]
y <- airq$Ozone
x <- airq[,-1]

## Apply caret with only pre's default settings (trControl and ntrees argument
## are employed here only to reduce computation time):
set.seed(42)
prefit1 <- train(x = x, y = y, method = caret_pre_model,
                 trControl = trainControl(number = 1),
                 ntrees = 5L)
prefit1

## Create custom tuneGrid:
set.seed(42)
tuneGrid <- caret_pre_model$grid(x = x, y = y,
                                 use.grad = c(TRUE, FALSE),
                                 maxdepth = 3L:5L,
                                 learnrate = c(.01, .1),
                                 penalty.par.val = c("lambda.1se", "lambda.min"))
tuneGrid
## Apply caret (again, ntrees and trControl set only to reduce computation time):
prefit2 <- train(x = x, y = y, method = caret_pre_model,
```

```
                trControl = trainControl(number = 1),
                tuneGrid = tuneGrid)
prefit2

## Get best tuning parameter values:
prefit2$bestTune
## Get predictions from model with best tuning parameters:
predict(prefit2, newdata = x[1:10,])
## Predictors included in model with best tuning parameter values:
predictors(prefit2)
varImp(prefit2)
plot(prefit2)

## Obtain tuning grid through random search over the tuning parameter space:
set.seed(42)
tuneGrid2 <- caret_pre_model$grid(x = x, y = y, search = "random", len = 10)
tuneGrid2
set.seed(42)
prefit3 <- train(x = x, y = y, method = caret_pre_model,
                trControl = trainControl(number = 1, verboseIter = TRUE),
                tuneGrid = tuneGrid2, ntrees = 5L)
prefit3

## Count response:
set.seed(42)
prefit4 <- train(x = x, y = y, method = caret_pre_model,
                trControl = trainControl(number = 1),
                ntrees = 5L, family = "poisson")
prefit4

## Binary factor response:
y_bin <- factor(airq$Ozone > mean(airq$Ozone))
set.seed(42)
prefit5 <- train(x = x, y = y_bin, method = caret_pre_model,
                trControl = trainControl(number = 1),
                ntrees = 5L, family = "binomial")
prefit5

## Factor response with > 2 levels:
x_multin <- airq[,-5]
y_multin <- factor(airq$Month)
set.seed(42)
prefit6 <- train(x = x_multin, y = y_multin, method = caret_pre_model,
                trControl = trainControl(number = 1),
                ntrees = 5L, family = "multinomial")
prefit6

## End(Not run)
```

---

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).
...	additional arguments to be passed to coef.glmnet.

See Also

coef.pre

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)`.
`...`	additional arguments to be passed to `coef.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`, `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), colorspace::rainbow_hcl)(100) 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.

See Also

See rainbow_hcl and colorRampPalette.

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?

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, 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", 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).
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.
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 provided in the plot(s).
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.

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.

See Also

pre, plot.pre, coef.pre, importance, 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

## Fit PRE with three trees and lower penalty parameter value,
## to allow linear terms to appear in final ensemble:
set.seed(42)
airq.ens2 <- pre(Ozone ~ ., data = airq[train,], ntrees = 3L)
## Compared to intercept, Month has negative and
## Day has positive contribution:
explain(airq.ens2, newdata = airq[-train,][1:2,],
        penalty.par.val = "lambda.min")$contribution
## Compared with training data means, Month has positive
## and Day has negative contribution:
explain(airq.ens2, newdata = airq[-train,][1:2,],
        penalty.par.val = "lambda.min", center.linear = TRUE)$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 \code{\link{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	list with control parameters to be passed to the tree fitting function, generated using ctree_control, mob_control (if use.grad = FALSE), or rpart.control (if tree.unbiased = FALSE).
use.grad	logical. Should gradient boosting with regression trees be employed when learnrate > 0? If TRUE, use trees fitted by ctree or rpart as in Friedman (2001), but without the line search. If use.grad = FALSE, glmtree instead of ctree will be employed for rule induction, yielding longer computation times, higher complexity, but possibly higher predictive accuracy. See Details for supported combinations of family, use.grad and learnrate.
removeduplicates	logical. Remove rules from the ensemble which are identical to an earlier rule?
removecomplements	logical. Remove rules from the ensemble which are identical to (1 - an earlier rule)?
tree.unbiased	logical. Should an unbiased tree generation algorithm be employed for rule generation? Defaults to TRUE, if set to FALSE, rules will be generated employing the CART algorithm (which suffers from biased variable selection) as implemented in rpart. See details below for possible combinations with family, use.grad and learnrate.

Examples

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

gpe_sample *Sampling Function Generator for gpe*

Description

Provides a sample function for gpe.

Usage

```
gpe_sample(sampfrac = 0.5)
```

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

Description

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

Usage

```
importance(object, standardize = FALSE, global = TRUE,
  quantprobs = c(0.75, 1), penalty.par.val = "lambda.1se",
  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

`object`	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`.

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).

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)`.

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. (2018). Fitting prediction rule ensembles with R package pre. https://arxiv.org/abs/1707.07149.

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

```
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, nullmods = NULL,
  penalty.par.val = "lambda.1se", 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).

`nullmods` object with bootstrapped null interaction models, resulting from application of `bsnullinteract`.

`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)`.

`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?

`...` Additional 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. (2018). Fitting prediction rule ensembles with R package pre. https://arxiv.org/abs/1707.07149.

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

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 specifyinf 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
```

`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",
  penalty.par.val = "lambda.1se", nvals = c(20, 20),
  pred.type = "response", ...)
```

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.
`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 2. For how many values of x1 and x2 should partial dependence be plotted? If NULL, all observed values for the two predictor variables specified will be used (see details).
pred.type	character string. Type of prediction to be plotted on z-axis. pred.type = "response" gives fitted values for continuous outputs and fitted probabilities for nominal outputs. pred.type = "link" gives fitted values for continuous outputs and linear predictor values for nominal outputs.
...	Additional arguments to be passed to image, contour or persp (depending on whether type is specified to be "heatmap", "contour", "both" or "perspective").

Details

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

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

Note

Function pairplot uses package akima to construct interpolated surfaces and has an ACM license that restricts applications to non-commercial usage, see https://www.acm.org/publications/policies/software-copyright-notice Function pairplot prints a note referring to this ACM licence.

References

Fokkema, M. (2018). Fitting prediction rule ensembles with R package pre. https://arxiv.org/abs/1707.07149.

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

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

plot.pre *Plot method for class pre*

Description

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

Usage

```
## S3 method for class 'pre'
plot(x, penalty.par.val = "lambda.1se",
  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).
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.
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

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

Usage

```
pre(formula, data, family = gaussian, use.grad = TRUE, weights,
  type = "both", sampfrac = 0.5, maxdepth = 3L, learnrate = 0.01,
  mtry = Inf, ntrees = 500, confirmatory = NULL,
  removecomplements = TRUE, removeduplicates = TRUE,
  winsfrac = 0.025, normalize = TRUE, standardize = FALSE,
  ordinal = TRUE, nfolds = 10L, tree.control, tree.unbiased = 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. 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, Otherwise, the program will try to make an informed guess, based on the class of the response variable specified in `formula`. als see Examples below.

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

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

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

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

`maxdepth` positive integer. Maximum number of conditions in rules. If `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.

`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?

`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.

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	list with control parameters to be passed to the tree fitting function, generated using `ctree_control`, `mob_control` (if `use.grad = FALSE`), or `rpart.control` (if `tree.unbiased = FALSE`).
tree.unbiased	logical. Should an unbiased tree generation algorithm be employed for rule generation? Defaults to `TRUE`, if set to `FALSE`, rules will be generated employing the CART algorithm (which suffers from biased variable selection) as implemented in `rpart`. See details below for possible combinations with `family`, `use.grad` and `learnrate`.
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? Likely improves computation times for large datasets.
...	Additional arguments to be passed to `cv.glmnet`.

Details

Note that obervations with missing values will be removed prior to analysis.

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

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

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'

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.value` argument, which can be used to select a different criterion.

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. (2018). Fitting prediction rule ensembles with R package pre. https://arxiv.org/abs/1707.07149.

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.

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, 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

## 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.pre <- pre(Species ~ ., data = iris, family = "multinomial")
iris.pre

## Fit pre to a survival response:
library("survival")
lung <- lung[complete.cases(lung), ]
set.seed(42)
lung.ens <- pre(Surv(time, status) ~ . - sex, 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 dataframe 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, 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
`...`	Additional arguments, currently not used.

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
...	Additional arguments, currently not used.

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

Examples

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

---

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 = "predicted", ...)
```

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

Details

By default, a partial dependence plot will be created for each unique observed value of the specified predictor variable. When the number of unique observed values is large, this may take a long time

to compute. In that case, specifying the `nvals` argument can substantially reduce computing time. When the `nvals` argument is supplied, values for the minimum, maximum, and `(nvals - 2)` intermediate values of the predictor variable will be plotted. Note that `nvals` can be specified only for numeric and ordered input variables. If the plot is requested for a nominal input variable, the `nvals` argument will be ignored and a warning printed.

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

References

Fokkema, M. (2018). Fitting prediction rule ensembles with R package pre. https://arxiv.org/abs/1707.07149.

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

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

`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)`.

`...` Additional arguments, currently not used.

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", ...)
```

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)`.

`...` Additional arguments, currently not used.

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, predict.pre, interact, cvpre

Examples

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

Index