Package 'pre'

August 31, 2017

Title Prediction Rule Ensembles

Version 0.4

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 several adjustments and improvements. The main function pre() derives prediction rule ensembles consisting of rules and/or linear terms for binary, count and 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 glmnet, MatrixModels, Formula, methods, graphics, partykit, earth, stringr

Suggests akima, grid, foreach, testthat, mlbench, datasets

License GPL-2 | GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-08-31 18:37:11 UTC

R topics documented:

`bsnullinteract` *Compute bootstrapped null interaction models*

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. Which value of the penalty parameter criterion should be used? The value yielding minimum cv error (`"lambda.min"`) or penalty parameter yielding error within 1 standard error of minimum cv error (`"lambda.1se"`)? Alternatively, a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet, for which estimated cv error can be inspected by running `object$glmnet.fit` and `plot(object$glmnet.fit)`.
`verbose`	logical. should progress be printed to the command line?

Details

Computationally intensive. Progress info is printed to command line.

Value

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

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"))
```

---

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 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 'gpe'
coef(object, penalty.par.val = "lambda.1se", ...)

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

Arguments

`object`	object of class `pre`
`penalty.par.val`	character. Penalty parameter criterion to be used for selecting final model: lambda giving minimum cv error (`"lambda.min"`) or lambda giving cv error that is within 1 standard error of minimum cv error (`"lambda.1se"`). Alternatively, a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet, for which estimated cv error can be inspected by running `object$glmnet.fit` and `plot(object$glmnet.fit)`.
`...`	additional arguments to be passed to `coef.glmnet`.

Details

In rare cases, duplucated variable names may appear in the model. For example, when the first variable is named 'V1' and is a factor, and there is a variable called 'V10' and/or 'V11' and/or 'V12' (etc), which is/are numeric. For the binary factor V1, dummy contrast variables were created to fit the model, called 'V10', 'V11', 'V12' (etc). As should be clear from this example, this yields replicated variable names, which may yield errors or incorrect results. Users should avoid this situation by renaming the variables prior to the 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 *Plotting baselearner correlations*

Description

corplot plots correlations between baselearners

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. Value of the penalty parameter value λ to be used for selecting the final ensemble. The ensemble with penalty parameter criterion yielding minimum cv error ("lambda.min") is taken, by default. Alternatively, the penalty parameter yielding error within 1 standard error of minimum cv error ("lambda.1se"), or a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet, for which estimated cv error can be inspected by running x$glmnet.fit and plot(x$glmnet.fit).

colors	vector of contiguous colors to be used for plotting. If colors = NULL (default), colorRampPalette(c("#053061", "#2166AC","#4393C3", "#92C5DE", "#D1E5F0", "#FFFFFF", " is used. A different set of plotting colors can be specified, for example: colors = cm.colors(100), or colorRampPalette(c("blue", "white", "red"))(150). See cm.colors or colorRampPalette.
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 breakspoints and colors to be depicted in the plot's legend.

Examples

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

---

cvpre *Full k-fold cross validation of a pre*

Description

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

Usage

```
cvpre(object, k = 10, verbose = FALSE, pclass = 0.5,
  penalty.par.val = "lambda.1se", parallel = FALSE)
```

Arguments

object	An object of class pre.
k	integer. The number of cross validation folds to be used.
verbose	logical. Should progress of the cross validation be printed to the command line?
pclass	numeric. Only used for classification. Cut-off value for the predicted probabilities that should be used to classify observations to the second class.
penalty.par.val	character. Calculate cross-validated error for ensembles with penalty parameter criterion giving minimum cv error ("lambda.min") or giving cv error that is within 1 standard error of minimum cv error ("lambda.1se")? Alternatively, a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet, for which estimated cv error can be inspected by running object$glmnet.fit and plot(object$glmnet.fit).
parallel	logical. Should parallel foreach be used? Must register parallel beforehand, such as doMC or others.

Value

A list with three objects: `$cvpreds` (a vector with cross-validated predicted y values), `$ss` (a vector indicating the cross-validation subsample each training observation was assigned to) and `$accuracy`. For continuous outputs, accuracy is a list with elements `$MSE` (mean squared error on test observations), `$MAE` (mean absolute error on test observations). For classification, accuracy is a list with elements `$SEL` (mean squared error on predicted probabilities), `$AEL` (mean absolute error on predicted probabilities), `$MCR` (average misclassification error rate) and `$table` (table with proportions of (in)correctly classified observations per class).

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)
```

---

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`. A similar fit to `pre` can be estimated with the following call:

```
gpe(formula = y ~ x1 + x2 + x3, data = data, base_learners = list(gpe_linear(), gpe_trees()))
```

Products of hinge functions using MARS can be added to the ensemble above with the following call:

```
gpe(formula = y ~ x1 + x2 + x3, data = data, base_learners = list(gpe_linear(), gpe_trees(), gpe_earth
```

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 The Annals of Applied Statistics, 2*(3), 916-954.

See Also

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

---

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_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 (rules and linear terms) and input variables*

Description

`importance` calculates importances for rules, linear terms and input variables in the ensemble, and provides 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",
  diag.xlab = TRUE, diag.xlab.hor = 0, diag.xlab.vert = 2, cex.axis = 1,
  ...)
```

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. Should model be selected with lambda yielding minimum cv error ("lambda.min"), or lambda giving cv error that is within 1 standard error of minimum cv error ("lambda.1se")? Alternatively, a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet.
`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`.
`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, 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.
...	further arguments to be passed to barplot (only used when plot = TRUE).

Value

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

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 user-specified variables*

Description

interact calculates test statistics for assessing the strength of interactions between the input variable(s) specified, 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("#8CC876", "#7FBFF5"), ylab = "Interaction strength",
  main = "Interaction test statistics", se.linewidth = 0.05,
  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. Which value of the penalty parameter criterion should be used? The value yielding minimum cv error ("lambda.min") or penalty parameter yielding error within 1 standard error of minimum cv error ("lambda.1se")? Alternatively, a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet, for which estimated cv error can be inspected by running object$glmnet.fit and plot(object$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 and Only the first element is 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).
parallel	logical. Should parallel foreach be used? Must register parallel beforehand, such as doMC or others.
k	integer. Calculating interaction test statistics is a 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 likely depend on the number of generated bootstrapped null interaction models as well as the complexity of the fitted ensembles. Users are therefore advised to test for the presence of interaction effects by setting the nsamp argument of the function bsnullinteract ≥ 100.

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"))
```

---

pairplot *Create partial dependence plot for a pair of predictor variables*

Description

pairplot creates a partial dependence plot to assess the effects of a pair of predictor variables on the predictions of the ensemble

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. Should model be selected with lambda giving minimum cv error ("lambda.min"), or lambda giving cv error that is within 1 standard error of minimum cv error ("lambda.1se")? Alternatively, a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet, for which estimated cv error can be inspected by running `object$glmnet.fit` and `plot(object$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.

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.

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, ask = FALSE, exit.label = "0", standardize = FALSE,
  plot.dim = c(3, 3), ...)
```

Arguments

x	an object of class pre.
penalty.par.val	character. Which value of the penalty parameter criterion should be used? The value yielding minimum cv error ("lambda.min") or penalty parameter yielding error within 1 standard error of minimum cv error ("lambda.1se")? Alternatively, a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet, for which estimated cv error can be inspected by running x$glmnet.fit and plot(x$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.
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 or binary outcome.

Usage

```
pre(formula, data, family = gaussian, use.grad = TRUE, weights,
  type = "both", sampfrac = 0.5, maxdepth = 3L, learnrate = 0.01,
  mtry = Inf, ntrees = 500, removecomplements = TRUE,
  removeduplicates = TRUE, winsfrac = 0.025, normalize = TRUE,
  standardize = FALSE, nfolds = 10L, verbose = FALSE, par.init = FALSE,
  par.final = FALSE, tree.control, ...)
```

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 or of class factor (with two levels). If the response is a factor, an ensemble for binary classification is created. Otherwise, an ensemble for prediction of a numeric response is created. If the outcome is a non-negative count, this should additionally be specified by setting`family = "poisson"`. Note that input variables may not have 'rule' as (part of) their name, and the formula may not exclude the intercept (that is, `+ 0` or `- 1` may not be used in the right-hand side of the formula).
data	data.frame containing the variables in the model. Response must be a factor for binary classification, numeric for (count) regression. Input variables must be of class numeric, factor or ordered factor.
family	specification of a glm family. Can be a character string (i.e., `"gaussian"`, `"binomial"` or `"poisson"`) or a corresponding family object (see `family`. Specification is required only for non-negative count responses, e.g., `family = "poisson"`. Otherwise, `family = "gaussian"` is employed if response is numeric and `family = "binomial"` is employed if response is a binary factor.
use.grad	logical. Should binary outcomes use gradient boosting with regression trees when `learnrate > 0`? That is, use `ctree` as in Friedman (2001), without the line search. By default set to `TRUE`, as this yields shorter computation time. If set to `FALSE`, `glmtree` with intercept only models in the nodes will be employed. This will yield longer computation times (but may increase the likelihood of detecting interactions).
weights	an optional vector of observation weights to be used for deriving the ensemble.
type	character. Specifies type of base learners to be included 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. Takes values > 0 and ≤ 1, representing 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., bootstrapping).
maxdepth	positive integer. Maximum number of conditions in a rule.
learnrate	numeric. Learning rate or boosting parameter.
mtry	numeric. Number of randomly selected predictor variables for creating each split in each tree.
ntrees	numeric. Number of trees to generate for the initial ensemble.
removecomplements	logical. Remove rules from the ensemble which have the same support in the training data as the inverse of other rules?
removeduplicates	logical. Remove rules from the ensemble which have the exact same support in training data?
winsfrac	numeric. 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`.
nfolds	numeric. Number of cross-validation folds to be used for selecting the optimal value of the penalty parameter λ in selecting the final ensemble.
verbose	logical. Should information on the initial and final ensemble be printed to the command line?
par.init	logical. Should parallel foreach be used to generate initial ensemble? Only used when learnrate == 0 and `family != "poisson"`. Must register parallel beforehand, such as doMC or others.
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.
tree.control	list with control parameters to be passed to the tree fitting function, generated using `ctree_control`, or `mob_control` if `use.grad = FALSE`.
...	Additional arguments to be passed to `cv.glmnet`.

Details

Obervations with missing values will be removed prior to analysis.

In rare cases, duplicated variable names may appear in the model. For example, the first variable is a factor named 'V1' and there are also non-factor variables called 'V10' and/or 'V11' and/or 'V12' (etc). Then for the binary factor V1, dummy contrast variables will be created, called 'V10', 'V11', 'V12' (etc). As should be clear from this example, this yields duplicated variable names, which will

yield warnings, errors and incorrect results. Users should prevent this by renaming variables prior to analysis.

Inputs can be numeric, ordered or factor variables. Reponse can be a numeric, count or binary categorical variable.

Value

an object of class `pre`, which contains the initial ensemble of rules and/or linear terms and the final ensembles for a wide range of penalty parameter values. By default, the final ensemble employed by all of the other methods and functions in package `pre` is selected using the 'minimum cross validated error plus 1 standard error' criterion. All functions and methods also take a `penalty.parameter.value` argument, which can be used to select a more or less sparse final ensembles. The `penalty.parameter.value` argument takes values `"lambda.1se"` (the default), `"lambda.min"`, or a numeric value. Users can assess the trade of between sparsity and accuracy provided by every possible value of the penalty parameter (λ) by running `object$glmnet.fit` and `plot(object$glmnet.fit)`.

Note

The code for deriving rules from the nodes of trees was taken from an internal function of the `partykit` package of Achim Zeileis and Torsten Hothorn.

References

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

See Also

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

Examples

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

---

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 unbiased 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 nominal outputs, type = "response" gives the fitted probabilities and type = "class" gives the predicted class membership.
penalty.par.val	character. Penalty parameter criterion to be used for selecting final model: lambda giving minimum cv error ("lambda.min") or lambda giving cv error that is within 1 standard error of minimum cv error ("lambda.1se"). Alternatively, a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet, for which estimated cv error can be inspected by running object$glmnet.fit and plot(object$glmnet.fit).
...	currently not used.

Details

When newdata is not provided, training data included in the specified object is used.

See Also

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

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 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 'gpe'
print(x, penalty.par.val = "lambda.1se",
  digits = getOption("digits"), ...)

## 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. Information for which final prediction rule ensemble(s) should be printed? The ensemble with penalty parameter criterion yielding minimum cv error ("lambda.min") or penalty parameter yielding error within 1 standard error of minimum cv error ("lambda.1se")? Alternatively, a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet, for which estimated cv error can be inspected by running x$glmnet.fit and plot(x$glmnet.fit).
digits	Number of digits 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 an accurate estimate of future prediction error.

Value

Prints information about the fitted prediction rule ensemble.

See Also

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*

Description

singleplot creates a partial dependence plot, which shows the effect of a predictor variable on the ensemble's predictions

Usage

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

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. penalty.par.val character. Penalty parameter criterion to be used for selecting final model: lambda giving minimum cv error

("lambda.min") or lambda giving cv error that is within 1 standard error of minimum cv error ("lambda.1se"). Alternatively, a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet, for which estimated cv error can be inspected by running `object$glmnet.fit` and `plot(object$glmnet.fit)`.

`penalty.par.val`
: character. Penalty parameter criterion to be used for selecting final model: lambda giving minimum cv error (`"lambda.min"`) or lambda giving cv error that is within 1 standard error of minimum cv error (`"lambda.1se"`). Alternatively, a numeric value may be specified, corresponding to one of the values of lambda in the sequence used by glmnet, for which estimated cv error can be inspected by running `object$glmnet.fit` and `plot(object$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.

`...`
: 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 is printed.

See Also

`pre`, `pairplot`

Examples

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

Index