Package ‘vimp’

April 27, 2020

Type Package
Title Perform Inference on Algorithm-Agnostic Variable Importance
Version 2.0.2
Description Calculate point estimates of and valid confidence intervals for
nonparametric, algorithm-agnostic variable importance measures in high and low dimensions,
using flexible estimators of the underlying regression functions. For more information
about the methods, please see Williamson et al. (Biometrics, 2020) and Williamson et al. (arXiv, 2020+) <arXiv:2004.03683>.
Depends R (>= 3.1.0)
Imports SuperLearner, stats, dplyr, magrittr, ROCR, tibble, rlang
Suggests knitr, rmarkdown, MASS, gam, xgboost, glmnet, ranger,
quadprog, covr, testthat, ggplot2, cowplot, RCurl, forcats
License MIT + file LICENSE
URL https://github.com/bdwilliamson/vimp
BugReports https://github.com/bdwilliamson/vimp/issues
LazyData TRUE
RoxygenNote 7.0.2
VignetteBuilder knitr
NeedsCompilation no
Author Brian D. Williamson [aut, cre]
<https://orcid.org/0000-0002-7024-548X>,
Noah Simon [aut] <https://orcid.org/0000-0002-8985-2474>,
Marco Carone [aut] <https://orcid.org/0000-0003-2106-0953>
Maintainer Brian D. Williamson <brianw26@uw.edu>
Repository CRAN
Date/Publication 2020-04-27 14:10:02 UTC
R topics documented:

average_vim .................................................. 2
cv_predictiveness_point_est ................................ 4
cv_predictiveness_update ................................... 5
cv_vim .......................................................... 6
cv_vimp_point_est ............................................. 9
cv_vimp_update ............................................... 10
format_vim ..................................................... 11
measure_accuracy ............................................. 12
measure_auc .................................................... 12
measure_cross_entropy ....................................... 13
measure_deviance ............................................. 13
measure_mse ................................................... 14
measure_r_squared .......................................... 14
merge_vim ...................................................... 15
predictiveness_ci ............................................. 16
predictiveness_point_est .................................... 17
predictiveness_se ............................................. 18
predictiveness_update ....................................... 18
print_vim ....................................................... 19
vimp ............................................................ 20
vimp_accuracy ................................................ 23
vimp_anova ..................................................... 26
vimp_auc ......................................................... 28
vimp_ci .......................................................... 31
vimp_deviance ................................................ 31
vimp_hypothesis_test ......................................... 34
vimp_point_est ................................................ 35
vimp_regression ............................................... 36
vimp_rsquared ................................................ 39
vimp_se ........................................................ 41
vimp_update .................................................... 42

Index 44

average_vim  Average multiple independent importance estimates

Description

Average the output from multiple calls to \texttt{vimp\_regression}, for different independent groups, into a single estimate with a corresponding standard error and confidence interval.

Usage

\begin{verbatim}
average_vim(..., weights = rep(1/length(list(...)), length(list(...)))
\end{verbatim}
average_vim

Arguments

...  an arbitrary number of vim objects.
weights  how to average the vims together, and must sum to 1; defaults to 1/(number of vims) for each vim, corresponding to the arithmetic mean

Value

an object of class vim containing the (weighted) average of the individual importance estimates, as well as the appropriate standard error and confidence interval. This results in a list containing:

- call - the call to average_vim()
- s - a list of the column(s) to calculate variable importance for
- SL.library - a list of the libraries of learners passed to SuperLearner
- full_fit - a list of the fitted values of the chosen method fit to the full data
- red_fit - a list of the fitted values of the chosen method fit to the reduced data
- est - a vector with the corrected estimates
- naive - a vector with the naive estimates
- update - a list with the influence curve-based updates
- mat - a matrix with the estimated variable importance, the standard error, and the \((1 - \alpha) \times 100\%\) confidence interval
- full_mod - a list of the objects returned by the estimation procedure for the full data regression (if applicable)
- red_mod - a list of the objects returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - the level, for confidence interval calculation
- y - a list of the outcomes

Examples

library(SuperLearner)
library(ranger)

## generate the data
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -5, 5)))

## apply the function to the x's
smooth <- (x[,1]/5)^2*(x[,1]+7)/5 + (x[,2]/3)^2

## generate Y ~ Normal (smooth, 1)
y <- smooth + stats::rnorm(n, 0, 1)

## set up a library for SuperLearner
learners <- "SL.ranger"

## get estimates on independent splits of the data
samp <- sample(1:n, n/2, replace = FALSE)

## using Super Learner (with a small number of folds, for illustration only)
est_2 <- vimp_regression(Y = y[samp], X = x[samp, ], indx = 2, V = 2, run_regression = TRUE, alpha = 0.05,
SL.library = learners, cvControl = list(V = 2))
est_1 <- vimp_regression(Y = y[-samp], X = x[-samp, ], indx = 2, V = 2, run_regression = TRUE, alpha = 0.05,
SL.library = learners, cvControl = list(V = 2))
est <- average_vim(est_1, est_2, weights = c(1/2, 1/2))

---

**cv_predictiveness_point_est**

Estimate a nonparametric predictiveness functional using cross-validation

---

**Description**

Compute nonparametric estimates of the chosen measure of predictiveness.

**Usage**

```r
cv_predictiveness_point_est(
  fitted_values,  # fitted values from a regression function; a list of length V, where each object is a set of predictions on the validation data.
y,  # the outcome.
weights = rep(1, length(y)),  # weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings)
folds,  # the cross-validation folds
type = "r_squared",  # which parameter are you estimating (defaults to anova, for ANOVA-based variable importance)?
na.rm = FALSE  # logical; should NA’s be removed in computation? (defaults to FALSE)
)```

**Arguments**

- `fitted_values`
- `y`
- `weights`
- `folds`
- `type`
- `na.rm`
cv_predictiveness_update

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

Value

The estimated measure of predictiveness.

cv_predictiveness_update

Estimate the influence function for an estimator of predictiveness

Description

Estimate the influence function for the given measure of predictiveness.

Usage

```r
cv_predictiveness_update(
  fitted_values, 
  y, 
  folds, 
  weights = rep(1, length(y)), 
  type = "r_squared", 
  na.rm = FALSE 
)
```

Arguments

- `fitted_values` fitted values from a regression function; a list of length V, where each object is a set of predictions on the validation data.
- `y` the outcome.
- `folds` the cross-validation folds.
- `weights` weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings).
- `type` which risk parameter are you estimating (defaults to `r_squared`, for the $R^2$)?
- `na.rm` logical; should NAs be removed in computation? (defaults to FALSE)

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

Value

The estimated influence function values for the given measure of predictiveness.
Nonparametric Variable Importance Estimates using Cross-validation

**Description**

Compute estimates and confidence intervals for the nonparametric variable importance parameter of interest, using cross-validation. This essentially involves splitting the data into V train/test splits; train the learners on the training data, evaluate importance on the test data; and average over these splits.

**Usage**

```r
cv_vim(
  Y,
  X,
  f1,
  f2,
  indx = 1,
  V = length(unique(folds)),
  folds = NULL,
  stratified = FALSE,
  weights = rep(1, length(Y)),
  type = "r_squared",
  run_regression = TRUE,
  SL.library = c("SL.glmnet", "SL.xgboost", "SL.mean"),
  alpha = 0.05,
  delta = 0,
  scale = "identity",
  na.rm = FALSE,
  ...
)
```

**Arguments**

- **Y** the outcome.
- **X** the covariates.
- **f1** the predicted values on validation data from a flexible estimation technique regressing Y on X in the training data; a list of length V, where each object is a set of predictions on the validation data.
- **f2** the predicted values on validation data from a flexible estimation technique regressing the fitted values in f1 on X withholding the columns in indx; a list of length V, where each object is a set of predictions on the validation data.
- **indx** the indices of the covariate(s) to calculate variable importance for; defaults to 1.
- **V** the number of folds for cross-validation, defaults to 10.
folds

the folds to use, if f1 and f2 are supplied. A list of length two; the first element
provides the outer folds (for hypothesis testing), while the second element is a
list providing the inner folds (for cross-validation).

stratified

if run_regression = TRUE, then should the generated folds be stratified based on
the outcome (helps to ensure class balance across cross-validation folds)

weights

weights for the computed influence curve (e.g., inverse probability weights for
crossension-at-random settings)

type

the type of parameter (e.g., ANOVA-based is "anova").

run_regression

if outcome Y and covariates X are passed to cv_vim, and run_regression is
TRUE, then Super Learner will be used; otherwise, variable importance will be
computed using the inputted fitted values.

SL.library

a character vector of learners to pass to SuperLearner, if f1 and f2 are Y and
X, respectively. Defaults to SL.glmnet, SL.xgboost, and SL.mean.

alpha

the level to compute the confidence interval at. Defaults to 0.05, corresponding
to a 95% confidence interval.

delta

the value of the δ-null (i.e., testing if importance < δ); defaults to 0.

scale

should CIs be computed on original ("identity") or logit ("logit") scale?

na.rm

should we remove NA's in the outcome and fitted values in computation? (de-
defaults to FALSE)

... other arguments to the estimation tool, see "See also".

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics
behind this function, and the validity of the confidence intervals. In the interest of transparency, we
return most of the calculations within the vim object. This results in a list containing:

• call - the call to cv_vim
• s - the column(s) to calculate variable importance for
• SL.library - the library of learners passed to SuperLearner
• full_fit - the fitted values of the chosen method fit to the full data (a list, for train and test data)
• red_fit - the fitted values of the chosen method fit to the reduced data (a list, for train and test
data)
• est - the estimated variable importance
• naive - the naive estimator of variable importance
• naives - the naive estimator on each fold
• updates - the influence curve-based update for each fold
• se - the standard error for the estimated variable importance
• ci - the $(1 - \alpha) \times 100\%$ confidence interval for the variable importance estimate
• full_mod - the object returned by the estimation procedure for the full data regression (if
applicable)
• red_mod - the object returned by the estimation procedure for the reduced data regression (if
applicable)
• alpha - the level, for confidence interval calculation
• folds - the folds used for hypothesis testing and cross-validation
• y - the outcome
• weights - the weights
• mat - a tibble with the estimate, SE, CI, hypothesis testing decision, and p-value

Value

An object of class `vim`. See Details for more information.

See Also

`SuperLearner` for specific usage of the `SuperLearner` function and package.

Examples

```r
library(Super Learner)
library(ranger)
n <- 100
p <- 2
## generate the data
x <- data.frame(replicate(p, stats::runif(n, -5, 5)))

## apply the function to the x's
smooth <- (x[,1]/5)^2*(x[,1]+7)/5 + (x[,2]/3)^2

## generate Y ~ Normal (smooth, 1)
y <- as.matrix(smooth + stats::rnorm(n, 0, 1))

## set up a library for Super Learner
learners <- c("SL.mean", "SL.ranger")

## using Super Learner (with a small number of folds, for illustration only)
## -----------------------------------------
set.seed(4747)
est <- cv_vim(Y = y, X = x, indx = 2, V = 2,
type = "r_squared", run_regression = TRUE,
SL.library = learners, cvControl = list(V = 2), alpha = 0.05)

## doing things by hand, and plugging them in (with a small number of folds, for illustration only)
## ------------------------------------------
## set up the folds
indx <- 2
V <- 2
set.seed(4747)
outer_folds <- sample(rep(seq_len(2), length = n))
inner_folds_1 <- sample(rep(seq_len(V), length = sum(outer_folds == 1)))
inner_folds_2 <- sample(rep(seq_len(V), length = sum(outer_folds == 2)))
y_1 <- y[outer_folds == 1, , drop = FALSE]
```
cv_vimp_point_est

Estimate variable importance using cross-validation

Description

Compute nonparametric estimates of the chosen variable importance parameter, with a correction for using data-adaptive techniques to estimate the conditional means only if necessary.

Usage

```r
cv_vimp_point_est(
  full, reduced,
  y, folds,
  weights = rep(1, length(y)),
  type = "r_squared",
  na.rm = FALSE
)
```
cv_vimp_update

Arguments

full

fitted values from a regression of the outcome on the full set of covariates; a list of length \( V \), where each object is a set of predictions on the validation data.

reduced

fitted values from a regression of the fitted values from the full regression on the reduced set of covariates; a list of length \( V \), where each object is a set of predictions on the validation data.

y

the outcome.

folds

a list of outer and inner folds (outer for hypothesis testing, inner for cross-validation)

weights

weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings)

type

which parameter are you estimating (defaults to anova, for ANOVA-based variable importance)?

na.rm

logical; should NA's be removed in computation? (defaults to FALSE)

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

Value

The estimated variable importance for the given group of left-out covariates.

cv_vimp_update

Estimate the influence function for variable importance parameters

Description

Compute the value of the influence function for the given group of left-out covariates.

Usage

cv_vimp_update(
  full,
  reduced,
  y,
  folds,
  weights = rep(1, length(y)),
  type = "r_squared",
  na.rm = FALSE
)
Arguments

- **full**: fitted values from a regression of the outcome on the full set of covariates; a list of length V, where each object is a set of predictions on the validation data.
- **reduced**: fitted values from a regression of the fitted values from the full regression on the reduced set of covariates; a list of length V, where each object is a set of predictions on the validation data.
- **y**: the outcome.
- **folds**: a list of outer and inner folds (outer for hypothesis testing, inner for cross-validation)
- **weights**: weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings)
- **type**: which parameter are you estimating (defaults to **anova**, for ANOVA-based variable importance)?
- **na.rm**: logical; should NAs be removed in computation? (defaults to **FALSE**)

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

Value

The influence function values for the given group of left-out covariates.

---

format.vim  

*Format a vim object*

Description

Nicely formats the output from a `vim` object for printing.

Usage

```r
## S3 method for class 'vim'
format(x, ...)  
```

Arguments

- **x**: the `vim` object of interest.
- **...**: other options, see the generic `format` function.
measure_accuracy

*Estimate the classification accuracy*

**Description**

Compute nonparametric estimate of classification accuracy.

**Usage**

```r
measure_accuracy(fitted_values, y, weights = rep(1, length(y)), na.rm = FALSE)
```

**Arguments**

- `fitted_values`: fitted values from a regression function.
- `y`: the outcome.
- `weights`: weights (IPW, etc.).
- `na.rm`: logical; should NA's be removed in computation? (defaults to FALSE)

**Value**

A named list of: (1) the estimated classification accuracy of the fitted regression function, and (2) the estimated influence function.

measure_auc

*Estimate area under the receiver operating characteristic curve (AUC)*

**Description**

Compute nonparametric estimate of AUC.

**Usage**

```r
measure_auc(fitted_values, y, weights = rep(1, length(y)), na.rm = FALSE)
```

**Arguments**

- `fitted_values`: fitted values from a regression function.
- `y`: the outcome.
- `weights`: weights (IPW, etc.).
- `na.rm`: logical; should NA's be removed in computation? (defaults to FALSE)

**Value**

A named list of: (1) the estimated AUC of the fitted regression function, and (2) the estimated influence function.
measure_cross_entropy  Estimate the cross-entropy

Description
Compute nonparametric estimate of cross-entropy.

Usage
measure_cross_entropy(
  fitted_values,
  y,
  weights = rep(1, length(y)),
  na.rm = FALSE
)

Arguments

fitted_values  fitted values from a regression function.
y  the outcome.
weights  weights (IPW, etc.).
na.rm  logical; should NA's be removed in computation? (defaults to FALSE)

Value
A named list of: (1) the estimated cross-entropy of the fitted regression function, and (2) the estimated influence function.

measure_deviance  Estimate the deviance

Description
Compute nonparametric estimate of deviance.

Usage
measure_deviance(fitted_values, y, weights = rep(1, length(y)), na.rm = FALSE)

Arguments

fitted_values  fitted values from a regression function.
y  the outcome.
weights  weights (IPW, etc.).
na.rm  logical; should NA's be removed in computation? (defaults to FALSE)
**measure_r_squared**

Estimate R-squared

**Description**

Estimate R-squared Compute nonparametric estimate of R-squared.

**Usage**

measure_r_squared(fitted_values, y, weights = rep(1, length(y)), na.rm = FALSE)

**Arguments**

fitted_values  fitted values from a regression function.

y  the outcome.

weights  weights (IPW, etc.).

na.rm  logical; should NA’s be removed in computation? (defaults to FALSE)

---

**measure_mse**

Estimate mean squared error

**Description**

Compute nonparametric estimate of mean squared error.

**Usage**

measure_mse(fitted_values, y, weights = rep(1, length(y)), na.rm = FALSE)

**Arguments**

fitted_values  fitted values from a regression function.

y  the outcome.

weights  weights (IPW, etc.).

na.rm  logical; should NA’s be removed in computation? (defaults to FALSE)

---

**Value**

A named list of: (1) the estimated deviance of the fitted regression function, and (2) the estimated influence function.

---

**Value**

A named list of: (1) the estimated mean squared error of the fitted regression function, and (2) the estimated influence function.
Value

A named list of: (1) the estimated R-squared of the fitted regression function, and (2) the estimated influence function.

merge_vim

Merge multiple vim objects into one

Description

Take the output from multiple different calls to vimp_regression and merge into a single vim object; mostly used for plotting results.

Usage

merge_vim(...)

Arguments

... an arbitrary number of vim objects, separated by commas.

Value

an object of class vim containing all of the output from the individual vim objects. This results in a list containing:

- call - the call to merge_vim()
- s - a list of the column(s) to calculate variable importance for
- SL.library - a list of the libraries of learners passed to SuperLearner
- full_fit - a list of the fitted values of the chosen method fit to the full data
- red_fit - a list of the fitted values of the chosen method fit to the reduced data
- est - a vector with the corrected estimates
- naive - a vector with the naive estimates
- update - a list with the influence curve-based updates
- se - a vector with the standard errors
- ci - a matrix with the CIs
- mat - a tibble with the estimated variable importance, the standard errors, and the $(1 - \alpha) \times 100\%$ confidence intervals
- full_mod - a list of the objects returned by the estimation procedure for the full data regression (if applicable)
- red_mod - a list of the objects returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - a list of the levels, for confidence interval calculation
Examples

```r
library(SuperLearner)
library(ranger)
## generate the data
## generate X
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -5, 5)))

## apply the function to the x's
smooth <- (x[,1]/5)^2*(x[,1]+7)/5 + (x[,2]/3)^2

## generate Y ~ Normal (smooth, 1)
y <- smooth + stats::rnorm(n, 0, 1)

## set up a library for SuperLearner
learners <- "SL.ranger"

## using Super Learner (with a small number of folds, for illustration only)
est_2 <- vimp_regression(Y = y, X = x, indx = 2, V = 2,
                         run_regression = TRUE, alpha = 0.05,
                         SL.library = learners, cvControl = list(V = 2))
est_1 <- vimp_regression(Y = y, X = x, indx = 1, V = 2,
                         run_regression = TRUE, alpha = 0.05,
                         SL.library = learners, cvControl = list(V = 2))
ests <- merge_vim(est_1, est_2)
```

predictiveness_ci

**Confidence intervals for measures of predictiveness**

**Description**

Compute confidence intervals for the true measure of predictiveness.

**Usage**

`predictiveness_ci(est, se, level = 0.95, one_sided = FALSE)`

**Arguments**

- `est` estimate of predictiveness, e.g., from a call to `predictiveness_point_est`
- `se` estimate of the standard error of `est`, e.g., from a call to `vimp_se`
- `level` confidence interval type (defaults to 0.95)
- `one_sided` should one-sided intervals be returned? (defaults to FALSE)
predictiveness_point_est

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

Value

The Wald-based confidence interval for the true predictiveness of the given group of covariates.

predictiveness_point_est

Estimate a nonparametric predictiveness functional

Description

Compute nonparametric estimates of the chosen measure of predictiveness.

Usage

predictiveness_point_est(
  fitted_values,  
  y,  
  weights = rep(1, length(y)),  
  type = "r_squared",  
  na.rm = FALSE  
)

Arguments

fitted_values  fitted values from a regression function.
y  the outcome.
weights  weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings)
type  which parameter are you estimating (defaults to anova, for ANOVA-based variable importance)?
na.rm  logical; should NA's be removed in computation? (defaults to FALSE)

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

Value

The estimated measure of predictiveness.
predictiveness_se

Estimate standard errors for measures of predictiveness.

Usage

predictiveness_se(est, update, denom = NULL, n = length(update), na.rm = FALSE)

Arguments

est the estimate of variable importance.
update the influence curve-based update.
denom a list of point estimate and influence curve for the denominator (if any) to make
the measure of predictiveness interpretable.
n the sample size.
na.rm logical; should NA's be removed in computation? (defaults to FALSE).

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics
behind this function and the definition of the parameter of interest.

Value

The standard error for the estimated measure of predictiveness for the given group of covariates.

predictiveness_update

Estimate the influence function for an estimator of predictiveness

Usage

predictiveness_update(  
  fitted_values,  
y,  
weights = rep(1, length(y)),  
type = "r_squared",  
na.rm = FALSE  
)
Arguments

fitted_values  fitted values from a regression function.
y  the outcome.
weights  weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings)
type  which risk parameter are you estimating (defaults to r_squared, for the $R^2$)?
na.rm  logical; should NAs be removed in computation? (defaults to FALSE)

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

Value

The estimated influence function values for the given measure of predictiveness.

---

print.vim  

Print a vim object

Description

Prints out the table of estimates, confidence intervals, and standard errors for a vim object.

Usage

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

Arguments

- `x`  the vim object of interest.
- `...`  other options, see the generic print function.
**Description**

Compute estimates of and confidence intervals for nonparametric risk-based variable importance.

**Usage**

```r
vim(
  Y,
  X,
  f1 = NULL,
  f2 = NULL,
  indx = 1,
  weights = rep(1, length(Y)),
  type = "r_squared",
  run_regression = TRUE,
  SL.library = c("SL.glmnet", "SL.xgboost", "SL.mean"),
  alpha = 0.05,
  delta = 0,
  scale = "identity",
  na.rm = FALSE,
  folds = NULL,
  stratified = FALSE,
  ...
)
```

**Arguments**

- **Y**: the outcome.
- **X**: the covariates.
- **f1**: the fitted values from a flexible estimation technique regressing \( Y \) on \( X \).
- **f2**: the fitted values from a flexible estimation technique regressing \( Y \) on \( X \) withholding the columns in \( \text{indx} \).
- **indx**: the indices of the covariate(s) to calculate variable importance for; defaults to 1.
- **weights**: weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings).
- **type**: the type of importance to compute; defaults to \( r\_\text{ squared} \), but other supported options are \( \text{auc} \), \( \text{accuracy} \), and \( \text{anova} \).
- **run_regression**: if outcome \( Y \) and covariates \( X \) are passed to \texttt{vimp\_accuracy}, and \texttt{run\_regression} is TRUE, then Super Learner will be used; otherwise, variable importance will be computed using the inputted fitted values.
- **SL.library**: a character vector of learners to pass to \texttt{SuperLearner}, if \texttt{f1} and \texttt{f2} are \( Y \) and \( X \), respectively. Defaults to \texttt{SL.glmnet}, \texttt{SL.xgboost}, and \texttt{SL.mean}.
alpha

the level to compute the confidence interval at. Defaults to 0.05, corresponding to a 95% confidence interval.

delta

the value of the δ-null (i.e., testing if importance < δ); defaults to 0.

scale

should CIs be computed on original ("identity") or logit ("logit") scale?

na.rm

should we remove NA's in the outcome and fitted values in computation? (defaults to FALSE)

folds

the folds used for f1 and f2; assumed to be 1 for the observations used in f1 and 2 for the observations used in f2. If there is only a single fold passed in, then hypothesis testing is not done.

stratified

if run_regression = TRUE, then should the generated folds be stratified based on the outcome (helps to ensure class balance across cross-validation folds)

... other arguments to the estimation tool, see "See also".

Details

In the interest of transparency, we return most of the calculations within the vim object. This results in a list containing:

- call - the call to vim
- s - the column(s) to calculate variable importance for
- SL.library - the library of learners passed to SuperLearner
- type - the type of risk-based variable importance measured
- full_fit - the fitted values of the chosen method fit to the full data
- red_fit - the fitted values of the chosen method fit to the reduced data
- est - the estimated variable importance
- naive - the naive estimator of variable importance
- update - the influence curve-based update
- se - the standard error for the estimated variable importance
- ci - the \((1 - \alpha) \times 100\%\) confidence interval for the variable importance estimate
- test - a decision to either reject (TRUE) or not reject (FALSE) the null hypothesis, based on a conservative test
- pval - a conservative p-value based on the same conservative test as test
- full_mod - the object returned by the estimation procedure for the full data regression (if applicable)
- red_mod - the object returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - the level, for confidence interval calculation
- folds - the folds used for hypothesis testing
- y - the outcome
- weights - the weights
- mat - a tibble with the estimate, SE, CI, hypothesis testing decision, and p-value
Value

An object of classes `vim` and the type of risk-based measure. See Details for more information.

See Also

`SuperLearner` for specific usage of the `SuperLearner` function and package.

Examples

```r
library(SuperLearner)
library(ranger)

# generate the data
# generate X
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -1, 1)))

# apply the function to the x's
f <- function(x) 0.5 + 0.3*x[1] + 0.2*x[2]
smooth <- apply(x, 1, function(z) f(z))

# generate Y ~ Normal (smooth, 1)
y <- matrix(rbinom(n, size = 1, prob = smooth))

# set up a library for SuperLearner
learners <- "SL.ranger"

# using Y and X; use class-balanced folds
folds_1 <- sample(rep(seq_len(2), length = sum(y == 1)))
folds_0 <- sample(rep(seq_len(2), length = sum(y == 0)))
folds <- vector("numeric", length(y))
folds[y == 1] <- folds_1
folds[y == 0] <- folds_0

est <- vim(y, x, indx = 2, type = "r_squared",
alpha = 0.05, run_regression = TRUE,
SL.library = learners, cvControl = list(V = 2),
folds = folds)

# using pre-computed fitted values
full <- SuperLearner(Y = y[folds == 1], X = x[folds == 1, ],
SL.library = learners, cvControl = list(V = 2))
full.fit <- predict(full)$pred
reduced <- SuperLearner(Y = y[folds == 2], X = x[folds == 2, -2, drop = FALSE],
SL.library = learners, cvControl = list(V = 2))
red.fit <- predict(reduced)$pred

est <- vim(Y = y, f1 = full.fit, f2 = red.fit,
indx = 2, run_regression = FALSE, alpha = 0.05, folds = folds,
type = "accuracy")
```
vimp: Nonparametric variable importance assessment

Description

The vimp package provides one major function: vim. This function calculates an estimate of the variable importance parameter of interest developed by Williamson, Gilbert, Simon, and Carone. The parameter is defined as the additional variability in the outcome explained by including the covariates of interest in the estimating procedure.

vimp Functions

The function vimp_regression() computes the estimates, standard error estimates, and confidence intervals for the ANOVA-based variable importance measure. It is an object of class "vim", and has its own print method.

The function merge_vim() takes the output of multiple calls to vimp_regression(), and combines the results into a single vim object.

The function format() formats a vim object for printing; print() prints the results; and plot() plots the estimates and standard errors.

vimp_accuracy

Nonparametric Variable Importance Estimates: Classification accuracy

Description

Compute estimates of and confidence intervals for nonparametric difference in classification accuracy-based variable importance. This is a wrapper function for cv_vim, with type = "accuracy".

Usage

vimp_accuracy(  
  Y,  
  X,  
  f1 = NULL,  
  f2 = NULL,  
  indx = 1,  
  V = 10,  
  weights = rep(1, length(Y)),  
  run_regression = TRUE,  
  SL.library = c("SL.glmnet", "SL.xgboost", "SL.mean"),  
  alpha = 0.05,  
  delta = 0,  
  na.rm = FALSE,  
)
folds = NULL,
stratified = TRUE,
scale = "identity",
...
)

Arguments

\textbf{Y} the outcome.

\textbf{X} the covariates.

\textbf{f1} the predicted values on validation data from a flexible estimation technique regressing \( Y \) on \( X \) in the training data; a list of length \( V \), where each object is a set of predictions on the validation data.

\textbf{f2} the predicted values on validation data from a flexible estimation technique regressing the fitted values in \( f1 \) on \( X \) withholding the columns in \( \text{indx} \); a list of length \( V \), where each object is a set of predictions on the validation data.

\textbf{indx} the indices of the covariate(s) to calculate variable importance for; defaults to 1.

\textbf{V} the number of folds for cross-validation, defaults to 10.

\textbf{weights} weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings)

\textbf{run_regression} if outcome \( Y \) and covariates \( X \) are passed to \texttt{cv_vim}, and \textbf{run_regression} is \texttt{TRUE}, then Super Learner will be used; otherwise, variable importance will be computed using the inputted fitted values.

\textbf{SL.library} a character vector of learners to pass to \texttt{SuperLearner}, if \( f1 \) and \( f2 \) are \( Y \) and \( X \), respectively. Defaults to \texttt{SL.glmnet}, \texttt{SL.xgboost}, and \texttt{SL.mean}.

\textbf{alpha} the level to compute the confidence interval at. Defaults to 0.05, corresponding to a 95\% confidence interval.

\textbf{delta} the value of the \( \delta \)-null (i.e., testing if importance < \( \delta \)); defaults to 0.

\textbf{na.rm} should we remove NA’s in the outcome and fitted values in computation? (defaults to \texttt{FALSE})

\textbf{folds} the folds to use, if \( f1 \) and \( f2 \) are supplied.

\textbf{stratified} if \textbf{run_regression} = \texttt{TRUE}, then should the generated folds be stratified based on the outcome (helps to ensure class balance across cross-validation folds)

\textbf{scale} scale should CIs be computed on original ("identity") or logit ("logit") scale? (defaults to "identity")

\textbf{...} other arguments to the estimation tool, see "See also".

Details

In the interest of transparency, we return most of the calculations within the \texttt{vim} object. This results in a list containing:

- \textbf{call} - the call to \texttt{vim}
- \textbf{s} - the column(s) to calculate variable importance for
• SL.library - the library of learners passed to SuperLearner
• full_fit - the fitted values of the chosen method fit to the full data
• red_fit - the fitted values of the chosen method fit to the reduced data
• est - the estimated variable importance
• naive - the naive estimator of variable importance
• update - the influence curve-based update
• se - the standard error for the estimated variable importance
• ci - the \((1 - \alpha) \times 100\%\) confidence interval for the variable importance estimate
• full_mod - the object returned by the estimation procedure for the full data regression (if applicable)
• red_mod - the object returned by the estimation procedure for the reduced data regression (if applicable)
• alpha - the level, for confidence interval calculation
• y - the outcome

Value
An object of classes \texttt{vim} and \texttt{vim\_accuracy}. See Details for more information.

See Also
\texttt{SuperLearner} for specific usage of the \texttt{SuperLearner} function and package.

Examples

library(SuperLearner)
library(ranger)
## generate the data
## generate X
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -1, 1)))

## apply the function to the x's
f <- function(x) 0.5 + 0.3*x[1] + 0.2*x[2]
smooth <- apply(x, 1, function(z) f(z))

## generate Y ~ Normal (smooth, 1)
y <- matrix(rbinom(n, size = 1, prob = smooth))

## set up a library for SuperLearner
learners <- "SL.ranger"

## estimate (with a small number of folds, for illustration only)
est <- vimp\_accuracy(y, x, indx = 2,
    alpha = 0.05, run\_regression = TRUE,
    SL.library = learners, V = 2, cvControl = list(V = 2))
vimp_anova

Nonparametric Variable Importance Estimates: ANOVA

Description

Compute estimates of and confidence intervals for nonparametric difference in classification accuracy-based variable importance. This is a wrapper function for cv_vim, with type = "anova".

Usage

vimp_anova(
  Y,
  X,
  f1 = NULL,
  f2 = NULL,
  indx = 1,
  V = 10,
  weights = rep(1, length(Y)),
  run_regression = TRUE,
  SL.library = c("SL.glmnet", "SL.xgboost", "SL.mean"),
  alpha = 0.05,
  delta = 0,
  na.rm = FALSE,
  scale = "identity",
  folds,
  stratified = FALSE,
  ...
)

Arguments

Y  the outcome.
X  the covariates.
f1  the predicted values on validation data from a flexible estimation technique regressing Y on X in the training data; a list of length V, where each object is a set of predictions on the validation data.
f2  the predicted values on validation data from a flexible estimation technique regressing the fitted values in f1 on X withholding the columns in indx; a list of length V, where each object is a set of predictions on the validation data.
indx  the indices of the covariate(s) to calculate variable importance for; defaults to 1.
V  the number of folds for cross-validation, defaults to 10.
weights  weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings)
run_regression  if outcome Y and covariates X are passed to cv_vim, and run_regression is TRUE, then Super Learner will be used; otherwise, variable importance will be computed using the inputted fitted values.
SL.library a character vector of learners to pass to SuperLearner, if f1 and f2 are Y and X, respectively. Defaults to SL.glmnet, SL.xgboost, and SL.mean.

alpha the level to compute the confidence interval at. Defaults to 0.05, corresponding to a 95% confidence interval.

delta the value of the δ-null (i.e., testing if importance < δ); defaults to 0.

na.rm should we remove NA's in the outcome and fitted values in computation? (defaults to FALSE)

scale scale should CIs be computed on original ("identity") or logit ("logit") scale? (defaults to "identity")

folds the folds to use, if f1 and f2 are supplied.

stratified if run_regression = TRUE, then should the generated folds be stratified based on the outcome (helps to ensure class balance across cross-validation folds)

... other arguments to the estimation tool, see "See also".

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function, and the validity of the confidence intervals. In the interest of transparency, we return most of the calculations within the vim object. This results in a list containing:

- call - the call to vim
- s - the column(s) to calculate variable importance for
- SL.library - the library of learners passed to SuperLearner
- full_fit - the fitted values of the chosen method fit to the full data
- red_fit - the fitted values of the chosen method fit to the reduced data
- est - the estimated variable importance
- naive - the naive estimator of variable importance
- update - the influence curve-based update
- se - the standard error for the estimated variable importance
- ci - the $(1 - \alpha) \times 100\%$ confidence interval for the variable importance estimate
- full_mod - the object returned by the estimation procedure for the full data regression (if applicable)
- red_mod - the object returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - the level, for confidence interval calculation
- y - the outcome

Value

An object of classes vim and vim_regression. See Details for more information.

See Also

SuperLearner for specific usage of the SuperLearner function and package.
Examples

```r
library(SuperLearner)
library(ranger)
## generate the data
## generate X
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -5, 5)))

## apply the function to the x's
smooth <- (x[,1]/5)^2*(x[,1]+7)/5 + (x[,2]/3)^2

## generate Y ~ Normal (smooth, 1)
y <- smooth + stats::rnorm(n, 0, 1)

## set up a library for SuperLearner
learners <- "SL.ranger"

## estimate (with a small number of folds, for illustration only)
est <- vimp_anova(y, x, indx = 2,
                  alpha = 0.05, run_regression = TRUE,
                  SL.library = learners, V = 2, cvControl = list(V = 2))
```

Legend

vimp_auc  Nonparametric Variable Importance Estimates: AUC

Description

Compute estimates of and confidence intervals for nonparametric difference in $AUC$-based variable importance. This is a wrapper function for cv_vim, with type = "auc".

Usage

```r
vimp_auc(
  Y,  # Predictive response
  X,  # Predictor(s)
  f1 = NULL,  # A function to be used for f1
  f2 = NULL,  # A function to be used for f2
  indx = 1,  # The index of the response
  V = 10,  # Number of folds
  weights = rep(1, length(Y)),  # Weights for each fold
  run_regression = TRUE,  # Whether to run a regression
  SL.library = c("SL.glmnet", "SL.xgboost", "SL.mean"),  # Library of learners
  alpha = 0.05,  # Significance level
  delta = 0,  # Minimum difference to consider as significant
  na.rm = FALSE,  # Whether to remove NA values
  folds = NULL,  # The folds to use
)
```
stratified = TRUE,
scale = "identity",
...
)

Arguments

Y
the outcome.

X
the covariates.

f1
the predicted values on validation data from a flexible estimation technique regressing Y on X in the training data; a list of length V, where each object is a set of predictions on the validation data.

f2
the predicted values on validation data from a flexible estimation technique regressing the fitted values in f1 on X withholding the columns in indx; a list of length V, where each object is a set of predictions on the validation data.

indx
the indices of the covariate(s) to calculate variable importance for; defaults to 1.

V
the number of folds for cross-validation, defaults to 10.

weights
weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings)

run_regression
if outcome Y and covariates X are passed to cv_vim, and run_regression is TRUE, then Super Learner will be used; otherwise, variable importance will be computed using the inputted fitted values.

SL.library
a character vector of learners to pass to SuperLearner, if f1 and f2 are Y and X, respectively. Defaults to SL.glmnet, SL.xgboost, and SL.mean.

alpha
the level to compute the confidence interval at. Defaults to 0.05, corresponding to a 95% confidence interval.

delta
the value of the δ-null (i.e., testing if importance < δ); defaults to 0.

na.rm
should we remove NA’s in the outcome and fitted values in computation? (defaults to FALSE)

folds
the folds to use, if f1 and f2 are supplied.

stratified
if run_regression = TRUE, then should the generated folds be stratified based on the outcome (helps to ensure class balance across cross-validation folds)

scale
scale should CIs be computed on original ("identity") or logit ("logit") scale? (defaults to "identity")

... other arguments to the estimation tool, see "See also".

Details

AUC for each regression (full and reduced) is computed using performance. In the interest of transparency, we return most of the calculations within the vim object. This results in a list containing:

• call - the call to vimp
• s - the column(s) to calculate variable importance for
vimp_auc

- SL.library - the library of learners passed to SuperLearner
- full_fit - the fitted values of the chosen method fit to the full data
- red_fit - the fitted values of the chosen method fit to the reduced data
- est - the estimated variable importance
- naive - the naive estimator of variable importance
- update - the influence curve-based update
- se - the standard error for the estimated variable importance
- ci - the \((1 - \alpha) \times 100\%\) confidence interval for the variable importance estimate
- full_mod - the object returned by the estimation procedure for the full data regression (if applicable)
- red_mod - the object returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - the level, for confidence interval calculation
- y - the outcome

Value

An object of classes \texttt{vim} and \texttt{vim_auc}. See Details for more information.

See Also

\texttt{SuperLearner} for specific usage of the \texttt{SuperLearner} function and package, and \texttt{performance} for specific usage of the \texttt{ROCR} package.

Examples

```
library(SuperLearner)
library(ranger)
## generate the data
## generate X
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -1, 1)))

## apply the function to the x's
f <- function(x) 0.5 + 0.3*x[1] + 0.2*x[2]
smooth <- apply(x, 1, function(z) f(z))

## generate Y ~ Normal (smooth, 1)
y <- matrix(rbinom(n, size = 1, prob = smooth))

## set up a library for SuperLearner
learners <- "SL.ranger"

## estimate (with a small number of folds, for illustration only)
est <- vimp_auc(y, x, indx = 2,
                 alpha = 0.05, run_regression = TRUE,
                 SL.library = learners, V = 2, cvControl = list(V = 2))
```
vimp_ci

Confidence intervals for variable importance

Description

Compute confidence intervals for the true variable importance parameter.

Usage

vimp_ci(est, se, scale = "identity", level = 0.95)

Arguments

est estimate of variable importance, e.g., from a call to vimp_point_est.
se estimate of the standard error of est, e.g., from a call to vimp_se.
scale scale to compute interval estimate on (defaults to "identity": compute SE and CI on log scale and back-transform).
level confidence interval type (defaults to 0.95).

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

Value

The Wald-based confidence interval for the true importance of the given group of left-out covariates.

vimp_deviance

Nonparametric Variable Importance Estimates: Deviance

Description

Compute estimates of and confidence intervals for nonparametric deviance-based variable importance. This is a wrapper function for cv_vim, with type = "deviance".

Usage

vimp_deviance(
  Y,
  X,
  f1 = NULL,
  f2 = NULL,
  indx = 1,
  V = 10,
)
weights = rep(1, length(Y)),
run_regression = TRUE,
SL.library = c("SL.glmnet", "SL.xgboost", "SL.mean"),
alpha = 0.05,
delta = 0,
na.rm = FALSE,
folds = NULL,
stratified = TRUE,
scale = "identity",
...)

Arguments

Y the outcome.
X the covariates.
f1 the predicted values on validation data from a flexible estimation technique regressing Y on X in the training data; a list of length V, where each object is a set of predictions on the validation data.
f2 the predicted values on validation data from a flexible estimation technique regressing the fitted values in f1 on X withholding the columns in indx; a list of length V, where each object is a set of predictions on the validation data.
indx the indices of the covariate(s) to calculate variable importance for; defaults to 1.
V the number of folds for cross-validation, defaults to 10.
weights weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings)
run_regression if outcome Y and covariates X are passed to cv_vim, and run_regression is TRUE, then Super Learner will be used; otherwise, variable importance will be computed using the inputted fitted values.
SL.library a character vector of learners to pass to SuperLearner, if f1 and f2 are Y and X, respectively. Defaults to SL.glmnet, SL.xgboost, and SL.mean.
alpha the level to compute the confidence interval at. Defaults to 0.05, corresponding to a 95% confidence interval.
delta the value of the δ-null (i.e., testing if importance < δ); defaults to 0.
na.rm should we remove NA's in the outcome and fitted values in computation? (defaults to FALSE)
folds the folds to use, if f1 and f2 are supplied.
stratified if run_regression = TRUE, then should the generated folds be stratified based on the outcome (helps to ensure class balance across cross-validation folds)
scale scale should CIs be computed on original ("identity") or logit ("logit") scale? (defaults to "identity")
... other arguments to the estimation tool, see "See also".
Details

In the interest of transparency, we return most of the calculations within the \texttt{vim} object. This results in a list containing:

- call - the call to \texttt{vim}
- s - the column(s) to calculate variable importance for
- SL.library - the library of learners passed to \texttt{SuperLearner}
- full_fit - the fitted values of the chosen method fit to the full data
- red_fit - the fitted values of the chosen method fit to the reduced data
- est - the estimated variable importance
- naive - the naive estimator of variable importance
- update - the influence curve-based update
- se - the standard error for the estimated variable importance
- ci - the \((1 - \alpha) \times 100\%\) confidence interval for the variable importance estimate
- full_mod - the object returned by the estimation procedure for the full data regression (if applicable)
- red_mod - the object returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - the level, for confidence interval calculation
- y - the outcome

Value

An object of classes \texttt{vim} and \texttt{vim_deviance}. See Details for more information.

See Also

\texttt{SuperLearner} for specific usage of the \texttt{SuperLearner} function and package.

Examples

```r
library(SuperLearner)
library(ranger)
## generate the data
## generate X
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -1, 1)))

## apply the function to the x's
f <- function(x) 0.5 + 0.3*x[1] + 0.2*x[2]
smooth <- apply(x, 1, function(z) f(z))

## generate Y ~ Normal (smooth, 1)
y <- matrix(stats::rbinom(n, size = 1, prob = smooth))
```
## set up a library for SuperLearner
learners <- "SL.ranger"

## estimate (with a small number of folds, for illustration only)
est <- vimp_deviance(y, x, indx = 2, 
alpha = 0.05, run_regression = TRUE, 
SL.library = learners, V = 2, cvControl = list(V = 2))

---

### vimp_hypothesis_test

**Perform a hypothesis test against the null hypothesis of $\delta$ importance**

**Description**

Perform a hypothesis test against the null hypothesis of zero importance by: (i) for a user-specified level $\alpha$, compute a $(1 - \alpha) \times 100\%$ confidence interval around the predictiveness for both the full and reduced regression functions (these must be estimated on independent splits of the data); (ii) if the intervals do not overlap, reject the null hypothesis.

**Usage**

```r
vimp_hypothesis_test(
  full,
  reduced,
  y,
  folds, 
  delta = 0,
  weights = rep(1, length(y)),
  type = "r_squared",
  alpha = 0.05,
  cv = FALSE,
  scale = "identity",
  na.rm = FALSE
)
```

**Arguments**

- `full` either (i) fitted values from a regression of the outcome on the full set of covariates from a first independent split of the data (if `cv = FALSE`) or (ii) a list of predicted values from a cross-validated procedure (if `cv = TRUE`).
- `reduced` fitted values from a regression either (1) of the outcome on the reduced set of covariates, or (2) of the predicted values from the full regression on the reduced set of covariates; either (i) a single set of predictions (if `cv = FALSE`) fit on an independent split of the data from `full` or (ii) a list of predicted values from a cross-validated procedure (if `cv = TRUE`).
- `y` the outcome.
folds

the folds used for splitting. If cv = FALSE, assumed to be a vector with 1 for the full regression and 2 for the reduced regression (if V = 2). If cv = TRUE, assumed to be a list with first element the outer folds (for hypothesis testing) and second element a list with the inner cross-validation folds.

delta

the value of the $\delta$-null (i.e., testing if importance < $\delta$); defaults to 0.

weights

weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings)

type

which parameter are you estimating (defaults to r_squared, for difference in R-squared-based variable importance)?

alpha

the desired type I error rate (defaults to 0.05).

cv

was V-fold cross-validation used to estimate the predictiveness (TRUE) or was the sample split in two (FALSE); defaults to FALSE.

scale

scale to compute CI on ("identity" for identity scale, "logit" for logit scale and back-transform)

na.rm

logical; should NAs be removed in computation? (defaults to FALSE)

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

Value

TRUE if the null hypothesis is rejected (i.e., if the confidence intervals do not overlap); otherwise, FALSE.

Usage

vimp_point_est(
full,
reduced,
y,
folds,
weights = rep(1, length(y)),
type = "r_squared",
na.rm = FALSE
)
vimp_regression

Arguments

- **full**: fitted values from a regression of the outcome on the full set of covariates.
- **reduced**: fitted values from a regression of the fitted values from the full regression on the reduced set of covariates.
- **y**: the outcome.
- **folds**: the folds for hypothesis testing.
- **weights**: weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings).
- **type**: which parameter are you estimating (defaults to anova, for ANOVA-based variable importance)?
- **na.rm**: logical; should NA's be removed in computation? (defaults to FALSE)

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

Value

The estimated variable importance for the given group of left-out covariates.

---

vimp_regression  Nonparametric Variable Importance Estimates

Description

Compute estimates of and confidence intervals for nonparametric ANOVA-based variable importance. This is a wrapper function for cv_vim, with type = "anova". This function is deprecated in vimp version 2.0.0.

Usage

vimp_regression(
  Y,
  X,
  f1 = NULL,
  f2 = NULL,
  indx = 1,
  V = 10,
  weights = rep(1, length(Y)),
  run_regression = TRUE,
  SL.library = c("SL.glmnet", "SL.xgboost", "SL.mean"),
  alpha = 0.05,
  delta = 0,
  na.rm = FALSE,
)
vimp_regression

folds,
stratified = FALSE,
...
)

Arguments

Y the outcome.
X the covariates.
f1 the predicted values on validation data from a flexible estimation technique re-
gressing Y on X in the training data; a list of length V, where each object is a set
of predictions on the validation data.
f2 the predicted values on validation data from a flexible estimation technique re-
gressing the fitted values in f1 on X withholding the columns in indx; a list of
length V, where each object is a set of predictions on the validation data.
indx the indices of the covariate(s) to calculate variable importance for; defaults to 1.
V the number of folds for cross-validation, defaults to 10.
weights weights for the computed influence curve (e.g., inverse probability weights for
coarsened-at-random settings)
run_regression if outcome Y and covariates X are passed to cv_vim, and run_regression is
TRUE, then Super Learner will be used; otherwise, variable importance will be
computed using the inputted fitted values.
SL.library a character vector of learners to pass to SuperLearner, if f1 and f2 are Y and
X, respectively. Defaults to SL.glmnet, SL.xgboost, and SL.mean.
alpha the level to compute the confidence interval at. Defaults to 0.05, corresponding
to a 95% confidence interval.
delta the value of the δ-null (i.e., testing if importance < δ); defaults to 0.
na.rm should we remove NA’s in the outcome and fitted values in computation? (de-
faults to FALSE)
folds the folds to use, if f1 and f2 are supplied.
stratified if run_regression = TRUE, then should the generated folds be stratified based on
the outcome (helps to ensure class balance across cross-validation folds)
...
other arguments to the estimation tool, see "See also".

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics
behind this function, and the validity of the confidence intervals. In the interest of transparency, we
return most of the calculations within the vim object. This results in a list containing:

- call - the call to vim
- s - the column(s) to calculate variable importance for
- SL.library - the library of learners passed to SuperLearner
- full_fit - the fitted values of the chosen method fit to the full data
vimp_regression

- red_fit - the fitted values of the chosen method fit to the reduced data
- est - the estimated variable importance
- naive - the naive estimator of variable importance
- update - the influence curve-based update
- se - the standard error for the estimated variable importance
- ci - the $(1 - \alpha) \times 100\%$ confidence interval for the variable importance estimate
- full_mod - the object returned by the estimation procedure for the full data regression (if applicable)
- red_mod - the object returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - the level, for confidence interval calculation
- y - the outcome

Value

An object of classes `vim` and `vim_regression`. See Details for more information.

See Also

`SuperLearner` for specific usage of the `SuperLearner` function and package.

Examples

```r
library(SuperLearner)
library(ranger)
## generate the data
## generate X
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -5, 5)))

## apply the function to the x's
smooth <- (x[,1]/5)^2*(x[,1]+7)/5 + (x[,2]/3)^2

## generate Y ~ Normal (smooth, 1)
y <- smooth + stats::rnorm(n, 0, 1)

## set up a library for SuperLearner
learners <- "SL.ranger"

## estimate (with a small number of folds, for illustration only)
est <- vimp_regression(y, x, indx = 2,
    alpha = 0.05, run_regression = TRUE,
    SL.library = learners, V = 2, cvControl = list(V = 2))
```

vimp_rsquared

Nonparametric Variable Importance Estimates: $R^2$

Description

Compute estimates of and confidence intervals for nonparametric $R^2$-based variable importance. This is a wrapper function for cv_vim, with type = "r_squared".

Usage

vimp_rsquared(
  Y,
  X,
  f1 = NULL,
  f2 = NULL,
  indx = 1,
  V = 10,
  weights = rep(1, length(Y)),
  run_regression = TRUE,
  SL.library = c("SL.glmnet", "SL.xgboost", "SL.mean"),
  alpha = 0.05,
  delta = 0,
  na.rm = FALSE,
  folds = NULL,
  stratified = FALSE,
  ...
)

Arguments

- **Y**: the outcome.
- **X**: the covariates.
- **f1**: the predicted values on validation data from a flexible estimation technique regressing Y on X in the training data; a list of length V, where each object is a set of predictions on the validation data.
- **f2**: the predicted values on validation data from a flexible estimation technique regressing the fitted values in f1 on X withholding the columns in indx; a list of length V, where each object is a set of predictions on the validation data.
- **indx**: the indices of the covariate(s) to calculate variable importance for; defaults to 1.
- **V**: the number of folds for cross-validation, defaults to 10.
- **weights**: weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings)
- **run_regression**: if outcome Y and covariates X are passed to cv_vim, and run_regression is TRUE, then Super Learner will be used; otherwise, variable importance will be computed using the inputted fitted values.
SL.library a character vector of learners to pass to SuperLearner, if f1 and f2 are Y and X, respectively. Defaults to SL.glmnet, SL.xgboost, and SL.mean.

alpha the level to compute the confidence interval at. Defaults to 0.05, corresponding to a 95% confidence interval.

delta the value of the δ-null (i.e., testing if importance < δ); defaults to 0.

na.rm should we remove NA's in the outcome and fitted values in computation? (defaults to FALSE)

folds the folds to use, if f1 and f2 are supplied.

stratified if run_regression = TRUE, then should the generated folds be stratified based on the outcome (helps to ensure class balance across cross-validation folds)

... other arguments to the estimation tool, see "See also".

Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function, and the validity of the confidence intervals. In the interest of transparency, we return most of the calculations within the vim object. This results in a list containing:

- call - the call to vim
- s - the column(s) to calculate variable importance for
- SL.library - the library of learners passed to SuperLearner
- full_fit - the fitted values of the chosen method fit to the full data
- red_fit - the fitted values of the chosen method fit to the reduced data
- est - the estimated variable importance
- naive - the naive estimator of variable importance
- update - the influence curve-based update
- se - the standard error for the estimated variable importance
- ci - the $(1 - \alpha) \times 100\%$ confidence interval for the variable importance estimate
- full_mod - the object returned by the estimation procedure for the full data regression (if applicable)
- red_mod - the object returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - the level, for confidence interval calculation
- y - the outcome

Value

An object of classes vim and vim_rsquared. See Details for more information.

See Also

SuperLearner for specific usage of the SuperLearner function and package.
library(SuperLearner)
library(ranger)

## generate the data
## generate X
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -5, 5)))

## apply the function to the x's
smooth <- (x[,1]/5)^2*(x[,1]+7)/5 + (x[,2]/3)^2

## generate Y ~ Normal (smooth, 1)
y <- smooth + stats::rnorm(n, 0, 1)

## set up a library for SuperLearner
learners <- "SL.ranger"

## estimate (with a small number of folds, for illustration only)
est <- vimp_rsquared(y, x, indx = 2,
                      alpha = 0.05, run_regression = TRUE,
                      SL.library = learners, V = 2, cvControl = list(V = 2))

vimp_se

Estimate standard errors

Compute standard error estimates for estimates of variable importance.

Usage

vimp_se(
est,
        update,
        denom = NULL,
        n = length(update),
        scale = "log",
        na.rm = FALSE
)

Arguments

est the estimate of variable importance.
update the influence curve-based update.
denom a list of point estimate and influence curve for the denominator (if any) to make the measure of predictiveness interpretable.
vimp_update

Estimate the influence function for variable importance parameters

**Details**

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

**Value**

The standard error for the estimated variable importance for the given group of left-out covariates.

---

**Description**

Compute the value of the influence function for the given group of left-out covariates.

**Usage**

```r
vimp_update(
  full,
  reduced,
  y,
  folds = folds,
  weights = rep(1, length(y)),
  type = "r_squared",
  na.rm = FALSE
)
```

**Arguments**

- `full`: fitted values from a regression of the outcome on the full set of covariates.
- `reduced`: fitted values from a regression either (1) of the outcome on the reduced set of covariates, or (2) of the fitted values from the full regression on the reduced set of covariates.
- `y`: the outcome.
- `folds`: the folds for hypothesis testing.
- `weights`: weights for the computed influence curve (e.g., inverse probability weights for coarsened-at-random settings).
- `type`: which parameter are you estimating (defaults to anova, for ANOVA-based variable importance)?
- `na.rm`: logical; should NAs be removed in computation? (defaults to FALSE)
Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

Value

The influence function values for the given group of left-out covariates.
Index

average_vim, 2

\texttt{cv\_predictiveness\_point\_est}, 4
\texttt{cv\_predictiveness\_update}, 5
\texttt{cv\_vim}, 6
\texttt{cv\_vimp\_point\_est}, 9
\texttt{cv\_vimp\_update}, 10

\texttt{format\_vim}, 11

measure\_accuracy, 12
measure\_auc, 12
measure\_cross\_entropy, 13
measure\_deviance, 13
measure\_mse, 14
measure\_r\_squared, 14
merge\_vim, 15

performance, 29, 30
predictiveness\_ci, 16
predictiveness\_point\_est, 17
predictiveness\_se, 18
predictiveness\_update, 18
print\_vim, 19

SuperLearner, 8, 22, 25, 27, 30, 33, 38, 40

vim, 20
vimp, 23
vimp\_accuracy, 23
vimp\_anova, 26
vimp\_auc, 28
vimp\_ci, 31
vimp\_deviance, 31
vimp\_hypothesis\_test, 34
vimp\_point\_est, 35
vimp\_regression, 36
vimp\_rsquared, 39
vimp\_se, 41
vimp\_update, 42