Package 'vip'

October 1, 2018

Type Package

Title Variable Importance Plots

Version 0.1.2

Description A general framework for constructing variable importance plots from various types machine learning models in R. Aside from some standard model-based variable importance measures, this package also provides a novel approach based on partial dependence plots (PDPs) and individual conditional expectation (ICE) curves as described in Greenwell et al. (2018) <arXiv:1805.04755>.

License GPL (>= 2)

URL https://koalaverse.github.io/vip/index.html, https://github.com/koalaverse/vip/

BugReports https://github.com/koalaverse/vip/issues

Encoding UTF-8

LazyData true

Imports ggplot2 (>= 0.9.0), gridExtra, magrittr, ModelMetrics, pdp, plyr, stats, tibble, utils

Suggests C50, caret, Ckmeans.1d.dp, covr, doParallel, dplyr, earth, gbm, glmnet, h2o, keras, knitr, lattice, mlbench, NeuralNetTools, nnet, party, partykit, randomForest, ranger, rmarkdown, rpart, sparklyr, testthat, xgboost

RoxygenNote 6.1.0

NeedsCompilation no

Author Brandon Greenwell [aut, cre] (<https://orcid.org/0000-0002-8120-0084>), Brad Boehmke [aut] (<https://orcid.org/0000-0002-3611-8516>), Bernie Gray [aut]

Maintainer Brandon Greenwell <greenwell.brandon@gmail.com>

Repository CRAN

Date/Publication 2018-09-30 22:00:03 UTC

R topics documented:

vi 2
vint 3
vip 4
vi_ice 5
vi_model 6
vi_pdp 8
vi_permute 9

Index **12**

`vi` *Variable Importance*

Description

Compute variable importance scores for the predictors in a model.

Usage

```
vi(object, method = c("model", "pdp", "ice", "permute"), feature_names,
  FUN = NULL, abbreviate_feature_names = NULL, sort = TRUE,
  decreasing = TRUE, scale = FALSE, ...)
```

Arguments

`object`	A fitted model object (e.g., a `"randomForest"` object).
`method`	Character string specifying the type of variable importance (VI) to compute. Current options are `"model"` (for model-based VI scores), `"pdp"` (for PDP-based VI scores), `"ice"` (for ICE-based VI scores), and `"permute"` (for permutation-based VI scores). The default is `"model"`. For details on the PDP/ICE-based method, see the reference below.
`feature_names`	Character string giving the names of the predictor variables (i.e., features) of interest.
`FUN`	List with two components, `"cat"` and `"con"`, containing the functions to use for categorical and continuous features, respectively. If `NULL`, the standard deviation is used for continuous features. For categorical features, the range statistic is used (i.e., (max - min) / 4).
`abbreviate_feature_names`	Integer specifying the length at which to abbreviate feature names. Default is `NULL` which results in no abbreviation (i.e., the full name of each feature will be printed).
`sort`	Logical indicating whether or not to order the sort the variable importance scores. Default is `TRUE`.

decreasing	Logical indicating whether or not the variable importance scores should be sorted in descending (TRUE) or ascending (FALSE) order of importance. Default is TRUE.
scale	Logical indicating whether or not to scale the variable importance scores so that the largest is 100. Default is FALSE.
...	Additional optional arguments.

Value

A tidy data frame (i.e., a "tibble" object) with two columns: Variable and Importance. For "glm"-like object, an additional column, called Sign, is also included which includes the sign (i.e., POS/NEG) of the original coefficient.

References

Greenwell, B. M., Boehmke, B. C., and McCarthy, A. J. A Simple and Effective Model-Based Variable Importance Measure. arXiv preprint arXiv:1805.04755 (2018).

Examples

```
#
# A projection pursuit regression example
#

# Load the sample data
data(mtcars)

# Fit a projection pursuit regression model
mtcars.ppr <- ppr(mpg ~ ., data = mtcars, nterms = 1)

# Compute variable importance scores
vi(mtcars.ppr, method = "ice")

# Plot variable importance scores
vip(mtcars.ppr, method = "ice")
```

---

vint *Interaction Effects*

Description

Compute the strength of two-way interaction effects. For details, see the reference below.

Usage

```
vint(object, feature_names, progress = "none", parallel = FALSE,
  paropts = NULL, ...)
```

Arguments

object	A fitted model object (e.g., a "randomForest" object).
feature_names	Character string giving the names of the two features of interest.
progress	Character string giving the name of the progress bar to use while constructing the interaction statistics. See create_progress_bar for details. Default is "none".
parallel	Logical indicating whether or not to run partial in parallel using a backend provided by the foreach package. Default is FALSE.
paropts	List containing additional options to be passed onto foreach when parallel = TRUE.
...	Additional optional arguments to be passed onto partial.

Details

Coming soon!

References

Greenwell, B. M., Boehmke, B. C., and McCarthy, A. J.: A Simple and Effective Model-Based Variable Importance Measure. arXiv preprint arXiv:1805.04755 (2018).

---

vip *Variable Importance Plots*

Description

Plot variable importance scores for the predictors in a model.

Usage

```
vip(object, ...)

## Default S3 method:
vip(object, num_features = 10L, bar = TRUE,
  width = 0.75, horizontal = TRUE, alpha = 1, color = "grey35",
  fill = "grey35", size = 1, shape = 19, ...)
```

Arguments

object	A fitted model object (e.g., a "randomForest" object) or an object that inherits from class "vi".
...	Additional optional arguments to be passed onto vi.
num_features	Integer specifying the number of variable importance scores to plot. Default is 10.
bar	Logical indicating whether or not to produce a barplot. Default is TRUE. If bar = FALSE, then a dotchart is displayed instead.

width	Numeric value specifying the width of the bars when bar = TRUE. Default is 0.75.
horizontal	Logical indicating whether or not to plot the importance scores on the x-axis (TRUE). Default is TRUE.
alpha	Numeric value between 0 and 1 giving the transparency of the bars (bar = TRUE) or points (bar = FALSE).
color	Character string specifying the color to use for the borders of the bars. Could also be a function, such as heat.colors. Default is "grey35".
fill	Character string specifying the color to use to fill the bars. Could also be a function, such as heat.colors. Default is "grey35".
size	Numeric value indicating the size to use for the points whenever bar = FALSE. Default is 1.
shape	Numeric value indicating the shape to use for the points whenever bar = FALSE. Default is 1.

Examples

```
#
# A projection pursuit regression example
#

# Load the sample data
data(mtcars)

# Fit a projection pursuit regression model
model <- ppr(mpg ~ ., data = mtcars, nterms = 1)

# Construct variable importance plot
vip(model, method = "ice")

# Better yet, store the variable importance scores and then plot
vi_scores <- vi(model, method = "ice")
vip(vi_scores, bar = FALSE, size = 3, horiz = FALSE)

# The \code{\link[magrittr]{\%T>\%}} operator is imported for convenience
vi_scores <- model %>%
  vi(method = "ice") %T>%
  {print(vip(.))}
vi_scores
```

---

vi_ice *ICE-Based Variable Importance*

Description

Compute ICE-based variable importance scores for the predictors in a model. (This function is meant for internal use only.)

Usage

```
vi_ice(object, ...)

## Default S3 method:
vi_ice(object, feature_names, FUN = NULL, ...)
```

Arguments

object	A fitted model object (e.g., a "randomForest" object).
...	Additional optional arguments to be passed onto partial.
feature_names	Character string giving the names of the predictor variables (i.e., features) of interest.
FUN	List with two components, "cat" and "con", containing the functions to use for categorical and continuous features, respectively. If NULL, the standard deviation is used for continuous features. For categorical features, the range statistic is used (i.e., (max - min) / 4).

Details

Coming soon!

Value

A tidy data frame (i.e., a "tibble" object) with two columns, Variable and Importance, containing the variable name and its associated importance score, respectively.

---

vi_model *Model-Based Variable Importance*

Description

Compute model-based variable importance scores for the predictors in a model. (This function is meant for internal use only.)

Usage

```
vi_model(object, ...)

## Default S3 method:
vi_model(object, ...)

## S3 method for class 'C5.0'
vi_model(object, ...)

## S3 method for class 'constparty'
vi_model(object, ...)
```

```
## S3 method for class 'earth'
vi_model(object, ...)

## S3 method for class 'gbm'
vi_model(object, ...)

## S3 method for class 'H2OBinomialModel'
vi_model(object, ...)

## S3 method for class 'H2OMultinomialModel'
vi_model(object, ...)

## S3 method for class 'H2ORegressionModel'
vi_model(object, ...)

## S3 method for class 'lm'
vi_model(object, ...)

## S3 method for class 'ml_model_decision_tree_regression'
vi_model(object, ...)

## S3 method for class 'ml_model_decision_tree_classification'
vi_model(object, ...)

## S3 method for class 'ml_model_gbt_regression'
vi_model(object, ...)

## S3 method for class 'ml_model_gbt_classification'
vi_model(object, ...)

## S3 method for class 'ml_model_generalized_linear_regression'
vi_model(object, ...)

## S3 method for class 'ml_model_linear_regression'
vi_model(object, ...)

## S3 method for class 'ml_model_random_forest_regression'
vi_model(object, ...)

## S3 method for class 'ml_model_random_forest_classification'
vi_model(object, ...)

## S3 method for class 'randomForest'
vi_model(object, ...)

## S3 method for class 'RandomForest'
vi_model(object, auc = FALSE, ...)
```

```
## S3 method for class 'ranger'
vi_model(object, ...)

## S3 method for class 'rpart'
vi_model(object, ...)

## S3 method for class 'train'
vi_model(object, ...)

## S3 method for class 'xgb.Booster'
vi_model(object, ...)
```

Arguments

object	A fitted model object (e.g., a "randomForest" object).
...	Additional optional arguments.
auc	Logical indicating whether or not to compute the AUC-based variable scores described in Janitza et al. (2012). Only available for cforest objects. See varimpAUC for details. Default is FALSE.

Details

Coming soon!

Value

A tidy data frame (i.e., a "tibble" object) with two columns: Variable and Importance. For "glm"-like object, an additional column, called Sign, is also included which includes the sign (i.e., POS/NEG) of the original coefficient.

---

vi_pdp *PDP-Based Variable Importance*

Description

Compute PDP-based variable importance scores for the predictors in a model. (This function is meant for internal use only.)

Usage

```
vi_pdp(object, ...)

## Default S3 method:
vi_pdp(object, feature_names, FUN = NULL, ...)
```

Arguments

`object`	A fitted model object (e.g., a `"randomForest"` object).
`...`	Additional optional arguments to be passed onto `partial`.
`feature_names`	Character string giving the names of the predictor variables (i.e., features) of interest.
`FUN`	List with two components, `"cat"` and `"con"`, containing the functions to use for categorical and continuous features, respectively. If `NULL`, the standard deviation is used for continuous features. For categorical features, the range statistic is used (i.e., (max - min) / 4).

Details

Coming soon!

Value

A tidy data frame (i.e., a `"tibble"` object) with two columns, `Variable` and `Importance`, containing the variable name and its associated importance score, respectively.

`vi_permute` *Permutation-Based Variable Importance*

Description

Compute permutation-based variable importance scores for the predictors in a model. (This function is meant for internal use only.)

Usage

```
vi_permute(object, ...)

## Default S3 method:
vi_permute(object, train, target, metric = "auto",
  smaller_is_better = NULL, reference_class = NULL, pred_fun = NULL,
  verbose = FALSE, progress = "none", parallel = FALSE,
  paropts = NULL, ...)
```

Arguments

`object`	A fitted model object (e.g., a `"randomForest"` object).
`...`	Additional optional arguments. (Currently ignored.)
`train`	A matrix-like R object (e.g., a data frame or matrix) containing the training data.
`target`	Either a character string giving the name (or position) of the target column in `train` or, if `train` only contains feature columns, a vector containing the target values used to train `object`.

`metric` Either a function or character string specifying the performance metric to use in computing model performance (e.g., RMSE for regression or accuracy for binary classification). If `metric` is a function, then it requires two arguments, `actual` and `predicted`, and should return a single, numeric value. Ideally, this should be the same metric that was to train `object`.

`smaller_is_better` Logical indicating whether or not a smaller value of `metric` is better. Default is `NULL`. Must be supplied if `metric` is a user-supplied function.

`reference_class` Character string specifying which response category represents the "reference" class (i.e., the class for which the predicted class probabilities correspond to). Only needed for binary classification problems.

`pred_fun` Optional prediction function that requires two arguments, `object` and `newdata`. Default is `NULL`. Must be supplied whenever `metric` is a custom function.

`verbose` Logical indicating whether or not to print information during the construction of variable importance scores. Default is `FALSE`.

`progress` Character string giving the name of the progress bar to use. See `create_progress_bar` for details. Default is `"none"`.

`parallel` Logical indicating whether or not to run `vi_permute()` in parallel (using a backend provided by the `foreach` package). Default is `FALSE`. If `TRUE`, an appropriate backend must be provided by `foreach`.

`paropts` List containing additional options to be passed onto `foreach` when `parallel = TRUE`.

Details

Coming soon!

Value

A tidy data frame (i.e., a `"tibble"` object) with two columns: `Variable` and `Importance`. For `"glm"`-like object, an additional column, called `Sign`, is also included which gives the sign (i.e., POS/NEG) of the original coefficient.

Examples

```
## Not run:
# Load required packages
library(ggplot2)  # for ggtitle() function
library(mlbench)  # for ML benchmark data sets
library(nnet)     # for fitting neural networks

# Simulate training data
set.seed(101)  # for reproducibility
trn <- as.data.frame(mlbench.friedman1(500))  # ?mlbench.friedman1

# Inspect data
tibble::as.tibble(trn)
```

```
# Fit PPR and NN models (hyperparameters were chosen using the caret package
# with 5 repeats of 5-fold cross-validation)
pp <- ppr(y ~ ., data = trn, nterms = 11)
set.seed(0803) # for reproducibility
nn <- nnet(y ~ ., data = trn, size = 7, decay = 0.1, linout = TRUE,
           maxit = 500)

# Plot VI scores
set.seed(2021)  # for reproducibility
p1 <- vip(pp, method = "permute", target = "y", metric = "rsquared",
          pred_fun = predict) + ggtitle("PPR")
p2 <- vip(nn, method = "permute", target = "y", metric = "rsquared",
          pred_fun = predict) + ggtitle("NN")
grid.arrange(p1, p2, ncol = 2)

# Mean absolute error
mae <- function(actual, predicted) {
  mean(abs(actual - predicted))
}

# Permutation-based VIP with user-defined MAE metric
set.seed(1101)  # for reproducibility
vip(pp, method = "permute", target = "y", metric = mae,
    smaller_is_better = TRUE,
    pred_fun = function(object, newdata) predict(object, newdata)  # wrapper
) + ggtitle("PPR")

## End(Not run)
```

Index

cforest, *8*
create_progress_bar, *4*, *10*

foreach, *4*

heat.colors, *5*

partial, *4*, *6*, *9*

varimpAUC, *8*
vi, 2, *4*
vi_ice, 5
vi_model, 6
vi_pdp, 8
vi_permute, 9
vint, 3
vip, 4