

Package ‘shapr’

January 28, 2021

Version 0.2.0

Title Prediction Explanation with Dependence-Aware Shapley Values

Description Complex machine learning models are often hard to interpret. However, in many situations it is crucial to understand and explain why a model made a specific prediction. Shapley values is the only method for such prediction explanation framework with a solid theoretical foundation. Previously known methods for estimating the Shapley values do, however, assume feature independence. This package implements the method described in Aas, Jullum and Løland (2019) <arXiv:1903.10464>, which accounts for any feature dependence, and thereby produces more accurate estimates of the true Shapley values.

URL <https://norskregnesentral.github.io/shapr/>,
<https://github.com/NorskRegnesentral/shapr>

BugReports <https://github.com/NorskRegnesentral/shapr/issues>

License MIT + file LICENSE

Encoding UTF-8

LazyData true

ByteCompile true

Language en-US

RoxygenNote 7.1.1

Depends R (>= 3.5.0)

Imports stats, data.table, Rcpp (>= 0.12.15), condMVNorm, mvnfast,
Matrix

Suggests ranger, xgboost, mgcv, testthat, knitr, rmarkdown, roxygen2,
MASS, ggplot2, caret, gbm, party, partykit

LinkingTo RcppArmadillo, Rcpp

VignetteBuilder knitr

NeedsCompilation yes

Author Nikolai Sellereite [aut] (<<https://orcid.org/0000-0002-4671-0337>>),
Martin Jullum [cre, aut] (<<https://orcid.org/0000-0003-3908-5155>>),
Annabelle Redelmeier [aut],
Anders Løland [ctb],

Jens Christian Wahl [ctb],
 Camilla Lingjærde [ctb],
 Norsk Regnesentral [cph, fnd]

Maintainer Martin Jullum <Martin.Jullum@nr.no>

Repository CRAN

Date/Publication 2021-01-28 15:30:03 UTC

R topics documented:

explain	2
feature_combinations	6
make_dummies	8
plot.shapr	9
shapr	10

Index	13
--------------	-----------

explain	<i>Explain the output of machine learning models with more accurately estimated Shapley values</i>
---------	--

Description

Explain the output of machine learning models with more accurately estimated Shapley values

Usage

```
explain(x, explainer, approach, prediction_zero, ...)
```

```
## S3 method for class 'empirical'
explain(
  x,
  explainer,
  approach,
  prediction_zero,
  type = "fixed_sigma",
  fixed_sigma_vec = 0.1,
  n_samples_aicc = 1000,
  eval_max_aicc = 20,
  start_aicc = 0.1,
  w_threshold = 0.95,
  ...
)
```

```
## S3 method for class 'gaussian'
explain(
```

```

    x,
    explainer,
    approach,
    prediction_zero,
    mu = NULL,
    cov_mat = NULL,
    ...
)

## S3 method for class 'copula'
explain(x, explainer, approach, prediction_zero, ...)

## S3 method for class 'ctree'
explain(
  x,
  explainer,
  approach,
  prediction_zero,
  mincriterion = 0.95,
  minsplit = 20,
  minbucket = 7,
  sample = TRUE,
  ...
)

## S3 method for class 'combined'
explain(
  x,
  explainer,
  approach,
  prediction_zero,
  mu = NULL,
  cov_mat = NULL,
  ...
)

## S3 method for class 'ctree_comb_mincrit'
explain(x, explainer, approach, prediction_zero, mincriterion, ...)

```

Arguments

<code>x</code>	A matrix or data.frame. Contains the the features, whose predictions ought to be explained (test data).
<code>explainer</code>	An explainer object to use for explaining the observations. See shapr .
<code>approach</code>	Character vector of length 1 or <code>n_features</code> . <code>n_features</code> equals the total number of features in the model. All elements should either be "gaussian", "copula", "empirical", or "ctree". See details for more information.

prediction_zero	Numeric. The prediction value for unseen data, typically equal to the mean of the response.
...	Additional arguments passed to prepare_data
type	Character. Should be equal to either "independence", "fixed_sigma", "AICc_each_k" or "AICc_full".
fixed_sigma_vec	Numeric. Represents the kernel bandwidth. Note that this argument is only applicable when approach = "empirical", and type = "fixed_sigma"
n_samples_aicc	Positive integer. Number of samples to consider in AICc optimization. Note that this argument is only applicable when approach = "empirical", and type is either equal to "AICc_each_k" or "AICc_full"
eval_max_aicc	Positive integer. Maximum number of iterations when optimizing the AICc. Note that this argument is only applicable when approach = "empirical", and type is either equal to "AICc_each_k" or "AICc_full"
start_aicc	Numeric. Start value of sigma when optimizing the AICc. Note that this argument is only applicable when approach = "empirical", and type is either equal to "AICc_each_k" or "AICc_full"
w_threshold	Positive integer between 0 and 1.
mu	Numeric vector. (Optional) Containing the mean of the data generating distribution. If NULL the expected values are estimated from the data. Note that this is only used when approach = "gaussian".
cov_mat	Numeric matrix. (Optional) Containing the covariance matrix of the data generating distribution. NULL means it is estimated from the data if needed (in the Gaussian approach).
mincriterion	Numeric value or vector where length of vector is the number of features in model. Value is equal to 1 - alpha where alpha is the nominal level of the conditional independence tests. If it is a vector, this indicates which mincriterion to use when conditioning on various numbers of features.
minsplit	Numeric value. Equal to the value that the sum of the left and right daughter nodes need to exceed.
minbucket	Numeric value. Equal to the minimum sum of weights in a terminal node.
sample	Boolean. If TRUE, then the method always samples n_samples from the leaf (with replacement). If FALSE and the number of obs in the leaf is less than n_samples, the method will take all observations in the leaf. If FALSE and the number of obs in the leaf is more than n_samples, the method will sample n_samples (with replacement). This means that there will always be sampling in the leaf unless sample = FALSE AND the number of obs in the node is less than n_samples.

Details

The most important thing to notice is that shapr has implemented four different approaches for estimating the conditional distributions of the data, namely "empirical", "gaussian", "copula" and "ctree".

In addition, the user also has the option of combining the four approaches. E.g. if you're in a situation where you have trained a model the consists of 10 features, and you'd like to use the "gaussian" approach when you condition on a single feature, the "empirical" approach if you condition on 2-5 features, and "copula" version if you condition on more than 5 features this can be done by simply passing `approach = c("gaussian", rep("empirical", 4), rep("copula", 5))`. If `approach[i] = "gaussian"` it means that you'd like to use the "gaussian" approach when conditioning on i features.

Value

Object of class `c("shapr", "list")`. Contains the following items:

dt `data.table`

model Model object

p Numeric vector

x_test `data.table`

Note that the returned items `model`, `p` and `x_test` are mostly added due to the implementation of `plot.shapr`. If you only want to look at the numerical results it is sufficient to focus on `dt`. `dt` is a `data.table` where the number of rows equals the number of observations you'd like to explain, and the number of columns equals $m + 1$, where m equals the total number of features in your model.

If `dt[i, j + 1] > 0` it indicates that the j -th feature increased the prediction for the i -th observation. Likewise, if `dt[i, j + 1] < 0` it indicates that the j -th feature decreased the prediction for the i -th observation. The magnitude of the value is also important to notice. E.g. if `dt[i, k + 1]` and `dt[i, j + 1]` are greater than 0, where $j \neq k$, and `dt[i, k + 1] > dt[i, j + 1]` this indicates that feature j and k both increased the value of the prediction, but that the effect of the k -th feature was larger than the j -th feature.

The first column in `dt`, called 'none', is the prediction value not assigned to any of the features (ϕ_0). It's equal for all observations and set by the user through the argument `prediction_zero`. In theory this value should be the expected prediction without conditioning on any features. Typically we set this value equal to the mean of the response variable in our training data, but other choices such as the mean of the predictions in the training data are also reasonable.

Author(s)

Camilla Lingjaerde, Nikolai Sellereite, Martin Jullum, Annabelle Redelmeier

Examples

```
if (requireNamespace("MASS", quietly = TRUE)) {
  # Load example data
  data("Boston", package = "MASS")

  # Split data into test- and training data
  x_train <- head(Boston, -3)
  x_test <- tail(Boston, 3)

  # Fit a linear model
  model <- lm(medv ~ lstat + rm + dis + indus, data = x_train)
```

```
# Create an explainer object
explainer <- shapr(x_train, model)

# Explain predictions
p <- mean(x_train$medv)

# Empirical approach
explain1 <- explain(x_test, explainer,
  approach = "empirical",
  prediction_zero = p, n_samples = 1e2
)

# Gaussian approach
explain2 <- explain(x_test, explainer,
  approach = "gaussian",
  prediction_zero = p, n_samples = 1e2
)

# Gaussian copula approach
explain3 <- explain(x_test, explainer,
  approach = "copula",
  prediction_zero = p, n_samples = 1e2
)

# ctree approach
explain4 <- explain(x_test, explainer,
  approach = "ctree",
  prediction_zero = p
)

# Combined approach
approach <- c("gaussian", "gaussian", "empirical", "empirical")
explain5 <- explain(x_test, explainer,
  approach = approach,
  prediction_zero = p, n_samples = 1e2
)

# Print the Shapley values
print(explain1$dt)

# Plot the results
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(explain1)
}
}
```

feature_combinations *Define feature combinations, and fetch additional information about each unique combination*

Description

Define feature combinations, and fetch additional information about each unique combination

Usage

```
feature_combinations(
  m,
  exact = TRUE,
  n_combinations = 200,
  weight_zero_m = 10^6
)
```

Arguments

<code>m</code>	Positive integer. Total number of features.
<code>exact</code>	Logical. If TRUE all 2^m combinations are generated, otherwise a subsample of the combinations is used.
<code>n_combinations</code>	Positive integer. Note that if <code>exact = TRUE</code> , <code>n_combinations</code> is ignored. However, if $m > 12$ you'll need to add a positive integer value for <code>n_combinations</code> .
<code>weight_zero_m</code>	Numeric. The value to use as a replacement for infinite combination weights when doing numerical operations.

Value

A data.table that contains the following columns:

id_combination Positive integer. Represents a unique key for each combination. Note that the table is sorted by `id_combination`, so that is always equal to `x[["id_combination"]] = 1:nrow(x)`.

features List. Each item of the list is an integer vector where `features[[i]]` represents the indices of the features included in combination `i`. Note that all the items are sorted such that `features[[i]] == sort(features[[i]])` is always true.

n_features Vector of positive integers. `n_features[i]` equals the number of features in combination `i`, i.e. `n_features[i] = length(features[[i]])`.

N Positive integer. The number of unique ways to sample `n_features[i]` features from `m` different features, without replacement.

Author(s)

Nikolai Sellereite, Martin Jullum

Examples

```
# All combinations
x <- feature_combinations(m = 3)
nrow(x) # Equals 2^3 = 8

# Subsample of combinations
x <- feature_combinations(exact = FALSE, m = 10, n_combinations = 1e2)
```

`make_dummies`*Initiate the making of dummy variables*

Description

Initiate the making of dummy variables

Usage

```
make_dummies(traindata, testdata)
```

Arguments

<code>traindata</code>	data.table or data.frame.
<code>testdata</code>	data.table or data.frame. New data that has the same feature names, types, and levels as <code>traindata</code> .

Value

A list that contains the following entries:

feature_list List. Output from `check_features`

train_dummies A data.frame containing all of the factors in `traindata` as one-hot encoded variables.

test_dummies A data.frame containing all of the factors in `testdata` as one-hot encoded variables.

traindata_new Original `traindata` with correct column ordering and factor levels. To be passed to [shapr](#).

testdata_new Original `testdata` with correct column ordering and factor levels. To be passed to [explain](#).

Author(s)

Annabelle Redelmeier, Martin Jullum

Examples

```
if (requireNamespace("MASS", quietly = TRUE)) {
  data("Boston", package = "MASS")
  x_var <- c("lstat", "rm", "dis", "indus")
  y_var <- "medv"
  x_train <- as.data.frame(Boston[401:411, x_var])
  y_train <- Boston[401:408, y_var]
  x_test <- as.data.frame(Boston[1:4, x_var])

  # convert to factors for illustational purpose
  x_train$rm <- factor(round(x_train$rm))
  x_test$rm <- factor(round(x_test$rm), levels = levels(x_train$rm))
}
```

```
dummylist <- make_dummies(traindata = x_train, testdata = x_test)
}
```

plot.shapr

Plot of the Shapley value explanations

Description

Plots the individual prediction explanations.

Usage

```
## S3 method for class 'shapr'
plot(
  x,
  digits = 3,
  plot_phi0 = TRUE,
  index_x_test = NULL,
  top_k_features = NULL,
  ...
)
```

Arguments

x	An shapr object. See explain .
digits	Integer. Number of significant digits to use in the feature description
plot_phi0	Logical. Whether to include phi0 in the plot
index_x_test	Integer vector. Which of the test observations to plot. E.g. if you have explained 10 observations using explain , you can generate a plot for the first 5 observations by setting index_x_test = 1:5.
top_k_features	Integer. How many features to include in the plot. E.g. if you have 15 features in your model you can plot the 5 most important features, for each explanation, by setting top_k_features = 1:5.
...	Currently not used.

Details

See vignette("understanding_shapr", package = "shapr") for an example of how you should use the function.

Value

ggplot object with plots of the Shapley value explanations

Author(s)

Martin Jullum

Examples

```
if (requireNamespace("MASS", quietly = TRUE)) {  
  #' # Load example data  
  data("Boston", package = "MASS")  
  
  # Split data into test- and training data  
  x_train <- head(Boston, -3)  
  x_test <- tail(Boston, 3)  
  
  # Fit a linear model  
  model <- lm(medv ~ lstat + rm + dis + indus, data = x_train)  
  
  # Create an explainer object  
  explainer <- shapr(x_train, model)  
  
  # Explain predictions  
  p <- mean(x_train$medv)  
  
  # Empirical approach  
  explanation <- explain(x_test,  
    explainer,  
    approach = "empirical",  
    prediction_zero = p,  
    n_samples = 1e2  
  )  
  
  if (requireNamespace("ggplot2", quietly = TRUE)) {  
    # Plot the explanation (this function)  
    plot(explanation)  
  }  
}
```

shapr

Create an explainer object with Shapley weights for test data.

Description

Create an explainer object with Shapley weights for test data.

Usage

```
shapr(x, model, n_combinations = NULL)
```

Arguments

<code>x</code>	Numeric matrix or data.frame/data.table. Contains the data used to estimate the (conditional) distributions for the features needed to properly estimate the conditional expectations in the Shapley formula.
<code>model</code>	The model whose predictions we want to explain. Run <code>shapr::get_supported_models()</code> for a table of which models shapr supports natively.
<code>n_combinations</code>	Integer. The number of feature combinations to sample. If NULL, the exact method is used and all combinations are considered. The maximum number of combinations equals $2^{\text{ncol}(x)}$.

Value

Named list that contains the following items:

exact Boolean. Equals TRUE if `n_combinations = NULL` or `n_combinations < 2ncol(x)`, otherwise FALSE.

n_features Positive integer. The number of columns in `x`

S Binary matrix. The number of rows equals the number of unique combinations, and the number of columns equals the total number of features. I.e. let's say we have a case with three features. In that case we have $2^3 = 8$ unique combinations. If the `j`-th observation for the `i`-th row equals 1 it indicates that the `j`-th feature is present in the `i`-th combination. Otherwise it equals 0.

W Second item

X data.table. Returned object from `feature_combinations`

x_train data.table. Transformed `x` into a data.table.

feature_list List. The `updated_feature_list` output from `preprocess_data`

In addition to the items above, `model` and `n_combinations` are also present in the returned object.

Author(s)

Nikolai Sellereite

Examples

```
if (requireNamespace("MASS", quietly = TRUE)) {
  # Load example data
  data("Boston", package = "MASS")
  df <- Boston

  # Example using the exact method
  x_var <- c("lstat", "rm", "dis", "indus")
  y_var <- "medv"
  df1 <- df[, x_var]
  model <- lm(medv ~ lstat + rm + dis + indus, data = df)
  explainer <- shapr(df1, model)

  print(nrow(explainer$X))
}
```

```
# 16 (which equals 2^4)

# Example using approximation
y_var <- "medv"
x_var <- setdiff(colnames(df), y_var)
model <- lm(medv ~ ., data = df)
df2 <- df[, x_var]
explainer <- shapr(df2, model, n_combinations = 1e3)

print(nrow(explainer$X))

# Example using approximation where n_combinations > 2^m
x_var <- c("lstat", "rm", "dis", "indus")
y_var <- "medv"
df3 <- df[, x_var]
model <- lm(medv ~ lstat + rm + dis + indus, data = df)
explainer <- shapr(df1, model, n_combinations = 1e3)

print(nrow(explainer$X))
# 16 (which equals 2^4)
}
```

Index

`explain`, [2](#), [8](#), [9](#)

`feature_combinations`, [6](#), [11](#)

`make_dummies`, [8](#)

`plot.shapr`, [9](#)

`prepare_data`, [4](#)

`preprocess_data`, [11](#)

`shapr`, [3](#), [8](#), [10](#)

`shapr::get_supported_models()`, [11](#)