

# Package ‘rSAFE’

May 9, 2026

**Title** Surrogate-Assisted Feature Extraction

**Version** 0.1.4

**Description**

Provides a model agnostic tool for white-box model trained on features extracted from a black-box model. For more information see: Gosiewska et al. (2020) <[doi:10.1016/j.dss.2021.113556](https://doi.org/10.1016/j.dss.2021.113556)>.

**Depends** R (>= 3.5.0)

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.1

**Imports** DALEX, dendextend, ggplot2, ggpubr, grDevices, ingredients, sets, stats

**Suggests** gbm, knitr, pander, randomForest, rmarkdown, spelling, testthat, vdiff

**VignetteBuilder** knitr

**URL** <https://github.com/ModelOriented/rSAFE>

**BugReports** <https://github.com/ModelOriented/rSAFE/issues>

**Language** en-US

**NeedsCompilation** no

**Author** Alicja Gosiewska [aut, cre],  
Anna Gierlak [aut],  
Przemyslaw Biecek [aut, ths],  
Michal Burdukiewicz [ctb] (ORCID:  
<<https://orcid.org/0000-0001-8926-582X>>)

**Maintainer** Alicja Gosiewska <[alicjagosiewska@gmail.com](mailto:alicjagosiewska@gmail.com)>

**Repository** CRAN

**Date/Publication** 2022-08-13 13:20:02 UTC

## Contents

|  |    |
|--|----|
| apartments . . . . .                   | 2  |
| HR_data . . . . .                      | 3  |
| plot.safe_extractor . . . . .          | 3  |
| print.safe_extractor . . . . .         | 4  |
| safely_detect_changepoints . . . . .   | 5  |
| safely_detect_interactions . . . . .   | 6  |
| safely_select_variables . . . . .      | 7  |
| safely_transform_categorical . . . . . | 8  |
| safely_transform_continuous . . . . .  | 9  |
| safely_transform_data . . . . .        | 10 |
| safe_extraction . . . . .              | 11 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>14</b> |
|--------------|-----------|

---

|            |                        |
|------------|------------------------|
| apartments | <i>Apartments data</i> |
|------------|------------------------|

---

### Description

Datasets apartments and apartmentsTest are artificial, generated from the same model. Structure of the dataset is copied from real dataset from PBImisc package, but they were generated in a way to mimic effect of Anscombe quartet for complex black box models.

### Usage

```
data(apartments)
```

### Format

a data frame with 1000 rows and 6 columns

### Details

- m2.price - price per square meter
- surface - apartment area in square meters
- no.rooms - number of rooms (correlated with surface)
- district - district in which apartment is located, factor with 10 levels (Bemowo, Bielany, Mokotow, Ochota, Praga, Srodmiescie, Ursus, Ursynow, Wola, Zoliborz)
- floor - floor
- construction.year - construction year

---

|         |   |
|---------|---|
| HR_data | <i>Why are our best and most experienced employees leaving prematurely?</i> |
|---------|---|

---

**Description**

A dataset from Kaggle competition Human Resources Analytics. <https://www.kaggle.com/>

**Format**

A data frame with 14999 rows and 10 variables

**Details**

- satisfaction\_level Level of satisfaction (0-1)
- last\_evaluation Time since last performance evaluation (in Years)
- number\_project Number of projects completed while at work
- average\_monthly\_hours Average monthly hours at workplace
- time\_spend\_company Number of years spent in the company
- work\_accident Whether the employee had a workplace accident
- left Whether the employee left the workplace or not (1 or 0) Factor
- promotion\_last\_5years Whether the employee was promoted in the last five years
- sales Department in which they work for
- salary Relative level of salary (high)

**Source**

Dataset HR-analytics from <https://www.kaggle.com>

---

|                     |  |
|---------------------|--|
| plot.safe_extractor | <i>Plotting Transformations of the SAFE Extractor Object</i> |
|---------------------|--|

---

**Description**

Plotting Transformations of the SAFE Extractor Object

**Usage**

```
## S3 method for class 'safe_extractor'  
plot(x, ..., variable = NULL)
```

**Arguments**

|          |  |
|----------|--|
| x        | safe_extractor object containing information about variables transformations created with safe_extraction() function |
| ...      | other parameters   |
| variable | character, name of the variable to be plotted  |

**Value**

a plot object

---

`print.safe_extractor` *Printing Summary of the SAFE Extractor Object*

---

**Description**

Printing Summary of the SAFE Extractor Object

**Usage**

```
## S3 method for class 'safe_extractor'
print(x, ..., variable = NULL)
```

**Arguments**

|          |   |
|----------|---|
| x        | safe_extractor object containing information about variables transformations created with safe_extraction() function                |
| ...      | other parameters  |
| variable | character, name of the variable to be plotted. If this argument is not specified then transformations for all variables are printed |

**Value**

No return value, prints the structure of the object

---

`safely_detect_changepoints`*Identifying Changes in a Series Using PELT Algorithm*

---

## Description

The `safely_detect_changepoints()` function calculates the optimal positioning and number of change-points for given data and penalty. It uses a PELT algorithm with a nonparametric cost function based on the empirical distribution. The implementation is inspired by the code available on <https://github.com/rkillick/changepoint>.

## Usage

```
safely_detect_changepoints(data, penalty = "MBIC", nquantiles = 10)
```

## Arguments

|                         |   |
|-------------------------|---|
| <code>data</code>       | a vector within which you wish to find changepoints   |
| <code>penalty</code>    | penalty for introducing another changepoint, one of "AIC", "BIC", "SIC", "MBIC", "Hannan-Quinn" or numeric non-negative value |
| <code>nquantiles</code> | the number of quantiles used in integral approximation  |

## Value

a vector of optimal changepoint positions (last observations of each segment)

## See Also

[safely\\_transform\\_continuous](#)

## Examples

```
library(rSAFE)

data <- rep(c(2,7), each=4)
safely_detect_changepoints(data)

set.seed(123)
data <- c(rnorm(15, 0), rnorm(20, 2), rnorm(30, 8))
safely_detect_changepoints(data)
safely_detect_changepoints(data, penalty = 25)
```

---

`safely_detect_interactions`*Detecting Interactions via Permutation Approach*

---

## Description

The `safely_detect_interactions()` function detects second-order interactions based on predictions made by a surrogate model. For each pair of features it performs values permutation in order to evaluate their non-additive effect.

## Usage

```
safely_detect_interactions(  
  explainer,  
  inter_param = 0.5,  
  inter_threshold = 0.5,  
  verbose = TRUE  
)
```

## Arguments

|                              |   |
|------------------------------|---|
| <code>explainer</code>       | DALEX explainer created with <code>explain()</code> function  |
| <code>inter_param</code>     | numeric, a positive value indicating which of single observation non-additive effects are to be regarded as significant, the higher value the higher non-additive effect has to be taken into account   |
| <code>inter_threshold</code> | numeric, a value from $[0, 1]$ interval indicating which interactions should be returned as significant. It corresponds to the percentage of observations for which interaction measure is greater than <code>inter_param</code> - if this percentage is less than <code>inter_threshold</code> then interaction effect is ignored. |
| <code>verbose</code>         | logical, if progress bar is to be printed   |

## Value

dataframe object containing interactions effects greater than or equal to the specified `inter_threshold`

## See Also

[safe\\_extraction](#)

## Examples

```
library(DALEX)  
library(randomForest)  
library(rSAFE)  
  
data <- apartments[1:500,]
```

```

set.seed(111)
model_rf <- randomForest(m2.price ~ construction.year + surface + floor +
                        no.rooms + district, data = data)
explainer_rf <- explain(model_rf, data = data[,2:6], y = data[,1])
safely_detect_interactions(explainer_rf, inter_param = 0.25,
                          inter_threshold = 0.2, verbose = TRUE)

```

---

**safely\_select\_variables**

*Performing Feature Selection on the Dataset with Transformed Variables*

---

**Description**

The `safely_select_variables()` function selects variables from dataset returned by `safely_transform_data()` function. For each original variable exactly one variable is chosen

- either original one or transformed one. The choice is based on the AIC value for linear model (regression) or logistic regression (classification).

**Usage**

```

safely_select_variables(
  safe_extractor,
  data,
  y = NULL,
  which_y = NULL,
  class_pred = NULL,
  verbose = TRUE
)

```

**Arguments**

|                             |   |
|-----------------------------|---|
| <code>safe_extractor</code> | object containing information about variables transformations created with <code>safe_extraction()</code> function  |
| <code>data</code>           | data, original dataset or the one returned by <code>safely_transform_data()</code> function. If data do not contain transformed variables then transformation is done inside this function using <code>'safe_extractor'</code> argument. Data may contain response variable or not - if it does then <code>'which_y'</code> argument must be given, otherwise <code>'y'</code> argument should be provided. |
| <code>y</code>              | vector of responses, must be given if data does not contain it  |
| <code>which_y</code>        | numeric or character (optional), must be given if data contains response values   |
| <code>class_pred</code>     | numeric or character, used only in multi-classification problems. If response vector has more than two levels, then <code>'class_pred'</code> should indicate the class of interest which will denote failure - all other classes will stand for success.   |
| <code>verbose</code>        | logical, if progress bar is to be printed   |

**Value**

vector of variables names, selected based on AIC values

**See Also**

[safely\\_transform\\_data](#)

**Examples**

```
library(DALEX)
library(randomForest)
library(rSAFE)

data <- apartments[1:500,]
set.seed(111)
model_rf <- randomForest(m2.price ~ construction.year + surface + floor +
                          no.rooms + district, data = data)
explainer_rf <- explain(model_rf, data = data[,2:6], y = data[,1])
safe_extractor <- safe_extraction(explainer_rf, verbose = FALSE)
safely_select_variables(safe_extractor, data, which_y = "m2.price", verbose = FALSE)
```

---

safely\_transform\_categorical

*Calculating a Transformation of Categorical Feature Using Hierarchical Clustering*

---

**Description**

The `safely_transform_categorical()` function calculates a transformation function for the categorical variable using predictions obtained from black box model and hierarchical clustering. The gap statistic criterion is used to determine the optimal number of clusters.

**Usage**

```
safely_transform_categorical(
  explainer,
  variable,
  method = "complete",
  B = 500,
  collapse = "_"
)
```

**Arguments**

|           |   |
|-----------|---|
| explainer | DALEX explainer created with <code>explain()</code> function      |
| variable  | a feature for which the transformation function is to be computed |

|          |  |
|----------|--|
| method   | the agglomeration method to be used in hierarchical clustering, one of: "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", "centroid" |
| B        | number of reference datasets used to calculate gap statistics  |
| collapse | a character string to separate original levels while combining them to the new one   |

**Value**

list of information on the transformation of given variable

**See Also**

[safe\\_extraction](#)

**Examples**

```
library(DALEX)
library(randomForest)
library(rSAFE)

data <- apartments[1:500,]
set.seed(111)
model_rf <- randomForest(m2.price ~ construction.year + surface + floor +
  no.rooms + district, data = data)
explainer_rf <- explain(model_rf, data = data[,2:6], y = data[,1])
safely_transform_categorical(explainer_rf, "district")
```

---

safely\_transform\_continuous

*Calculating a Transformation of a Continuous Feature Using PDP/ALE Plot*

---

**Description**

The `safely_transform_continuous()` function calculates a transformation function for the continuous variable using a PD/ALE plot obtained from black box model.

**Usage**

```
safely_transform_continuous(
  explainer,
  variable,
  response_type = "ale",
  grid_points = 50,
  N = 200,
  penalty = "MBIC",
  nquantiles = 10,
  no_segments = 2
)
```

**Arguments**

|               |   |
|---------------|---|
| explainer     | DALEX explainer created with explain() function   |
| variable      | a feature for which the transformation function is to be computed   |
| response_type | character, type of response to be calculated, one of: "pdp", "ale". If features are uncorrelated, one can use "pdp" type - otherwise "ale" is strongly recommended. |
| grid_points   | number of points on x-axis used for creating the PD/ALE plot, default 50  |
| N             | number of observations from the dataset used for creating the PD/ALE plot, default 200  |
| penalty       | penalty for introducing another changepoint, one of "AIC", "BIC", "SIC", "MBIC", "Hannan-Quinn" or numeric non-negative value                                       |
| nquantiles    | the number of quantiles used in integral approximation  |
| no_segments   | numeric, a number of segments variable is to be divided into in case of founding no breakpoints   |

**Value**

list of information on the transformation of given variable

**See Also**

[safe\\_extraction](#), [safely\\_detect\\_changepoints](#)

**Examples**

```
library(DALEX)
library(randomForest)
library(rSAFE)

data <- apartments[1:500,]
set.seed(111)
model_rf <- randomForest(m2.price ~ construction.year + surface + floor +
  no.rooms + district, data = data)
explainer_rf <- explain(model_rf, data = data[,2:6], y = data[,1])
safely_transform_continuous(explainer_rf, "construction.year")
```

---

safely\_transform\_data *Performing Transformations on All Features in the Dataset*

---

**Description**

The safely\_transform\_data() function creates new variables in dataset using safe\_extractor object.

**Usage**

```
safely_transform_data(safe_extractor, data, verbose = TRUE)
```

**Arguments**

safe\_extractor object containing information about variables transformations created with safe\_extraction() function

data data for which features are to be transformed

verbose logical, if progress bar is to be printed

**Value**

data with extra columns containing newly created variables

**See Also**

[safe\\_extraction](#), [safely\\_select\\_variables](#)

**Examples**

```
library(DALEX)
library(randomForest)
library(rSAFE)

data <- apartments[1:500,]
set.seed(111)
model_rf <- randomForest(m2.price ~ construction.year + surface + floor +
                        no.rooms + district, data = data)
explainer_rf <- explain(model_rf, data = data[,2:6], y = data[,1])
safe_extractor <- safe_extraction(explainer_rf, verbose = FALSE)
safely_transform_data(safe_extractor, data, verbose = FALSE)
```

---

safe\_extraction      *Creating SAFE Extractor - an Object Used for Surrogate-Assisted Feature Extraction*

---

**Description**

The safe\_extraction() function creates a SAFE-extractor object which may be used later for surrogate feature extraction.

**Usage**

```
safe_extraction(
  explainer,
  response_type = "ale",
  grid_points = 50,
  N = 200,
  penalty = "MBIC",
  nquantiles = 10,
  no_segments = 2,
```

```

method = "complete",
B = 500,
collapse = "_",
interactions = FALSE,
inter_param = 0.25,
inter_threshold = 0.25,
verbose = TRUE
)

```

### Arguments

|                 |   |
|-----------------|---|
| explainer       | DALEX explainer created with explain() function   |
| response_type   | character, type of response to be calculated, one of: "pdp", "ale". If features are uncorrelated, one can use "pdp" type - otherwise "ale" is strongly recommended.   |
| grid_points     | number of points on x-axis used for creating the PD/ALE plot, default 50  |
| N               | number of observations from the dataset used for creating the PD/ALE plot, default 200  |
| penalty         | penalty for introducing another changepoint, one of "AIC", "BIC", "SIC", "MBIC", "Hannan-Quinn" or numeric non-negative value   |
| nquantiles      | the number of quantiles used in integral approximation  |
| no_segments     | numeric, a number of segments variable is to be divided into in case of founding no breakpoints   |
| method          | the agglomeration method to be used in hierarchical clustering, one of: "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", "centroid"  |
| B               | number of reference datasets used to calculate gap statistics   |
| collapse        | a character string to separate original levels while combining them to the new one  |
| interactions    | logical, if interactions between variables are to be taken into account   |
| inter_param     | numeric, a positive value indicating which of single observation non-additive effects are to be regarded as significant, the higher value the higher non-additive effect has to be taken into account   |
| inter_threshold | numeric, a value from $[0, 1]$ interval indicating which interactions should be returned as significant. It corresponds to the percentage of observations for which interaction measure is greater than inter_param - if this percentage is less than inter_threshold then interaction effect is ignored. |
| verbose         | logical, if progress bar is to be printed   |

### Value

safe\_extractor object containing information about variables transformation

### See Also

[safely\\_transform\\_categorical](#), [safely\\_transform\\_continuous](#), [safely\\_detect\\_interactions](#), [safely\\_transform\\_data](#)

**Examples**

```
library(DALEX)
library(randomForest)
library(rSAFE)

data <- apartments[1:500,]
set.seed(111)
model_rf <- randomForest(m2.price ~ construction.year + surface + floor +
                          no.rooms + district, data = data)
explainer_rf <- explain(model_rf, data = data[,2:6], y = data[,1], verbose = FALSE)
safe_extractor <- safe_extraction(explainer_rf, grid_points = 30, N = 100, verbose = FALSE)
print(safe_extractor)
plot(safe_extractor, variable = "construction.year")
```

# Index

\* **apartments**

apartments, [2](#)

\* **datasets**

HR\_data, [3](#)

apartments, [2](#)

apartmentsTest (apartments), [2](#)

HR\_data, [3](#)

plot.safe\_extractor, [3](#)

print.safe\_extractor, [4](#)

safe\_extraction, [6](#), [9–11](#), [11](#)

safely\_detect\_changepoints, [5](#), [10](#)

safely\_detect\_interactions, [6](#), [12](#)

safely\_select\_variables, [7](#), [11](#)

safely\_transform\_categorical, [8](#), [12](#)

safely\_transform\_continuous, [5](#), [9](#), [12](#)

safely\_transform\_data, [8](#), [10](#), [12](#)