Package 'stranger'

January 25, 2018

Type Package

Title Simple Toolkit in R for ANomalies Get, Explain and Report

Version 0.3.2

Maintainer Eric Lecoutre <eric.lecoutre@welovedatascience.com>

Description Framework for unsupervised anomalies detection that simplifies the user experience because the one does not need to be concerned with the many packages and functions that are required. Package 'stranger' acts as a wrapper around existing packages (``a la 'caret' for modeling'') and provides a clean and uniform toolkit for evaluation/explain/reporting purposes.

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

URL https://welovedatascience.github.io/stranger,
https://github.com/welovedatascience/stranger

Suggests knitr, rmarkdown, prettydoc, dtplyr, testthat, FNN, dbscan, autoencoder, rpart, ranger, abodOutlier, mvoutlier, randomForest

VignetteBuilder knitr

Imports assertthat (>= 0.2) , ggplot2, tidyr, grid, methods

Depends R (>= 2.10), data.table, dplyr (>= 0.7.3)

NeedsCompilation no

Author Eric Lecoutre [aut, cre], Sven Wauters [aut]

Repository CRAN

Date/Publication 2018-01-25 14:31:13 UTC

R topics documented:

`add_id` *Enrich source data with an ID*

Description

`add_id` just ensures data is a data.table and adds a columns named .id

Usage

```
add_id(data)
```

Arguments

`data`	Source data (data.frame or data.table)

Details

`add_id` in called behing the scence by `crazyfy` if needed to ensure a valid id is present in source data. Still, we recommand to start from a dataset already having an id, be it generated by a call to `add_id` or already pre-existing to ensure correct source data enrichment.

Value

data.table object with a column named .id

Examples

```
library(stranger)
data(iris)
(iris.id <- add_id(iris))
```

`as.anomalies` *Create a class anomalies from a (pre-filterd) stranger/singular*

Description

One can use either dplyr filter verb or subseting records with base "[" to manually define anomalies Then `as.anomalies` can be used to create anomalies object to be used for instance for visualisation purpose.

Usage

```
as.anomalies(x)
```

Arguments

x	stranger/singular (pre-filered) object

Details

Note that it is expected from the user to filter the object. If not done, all records are defined as anomalies. This manual approach can be useful with some metrics and litterature recommandation, for instance using a cutpoint of 2 for 'abod' method.

Value

object of class anomalies – see also `get_anomalies`.

`BudgetUK` *Budget Shares of British Households*

Description

Budget Shares of British Households

Usage

```
data(BudgetUK)
```

Format

A dataframe containing 1.519 rows and 10 columns.

- wfood. budget share for food expenditure
- wfuel. budget share for fuel expenditure
- wcloth. budget share for clothing expenditure
- walc. budget share for alcohol expenditure
- wtrans. budget share for transport expenditure
- wother. budget share for other good expenditure
- totexp. total household expenditure (rounded to the nearest 10 UK pounds sterling)
- income. total net household income (rounded to the nearest 10 UK pounds sterling)
- age. age of household head
- children. number of children

Source

Blundell, Richard, Alan Duncan and Krishna Pendakur (1998) "Semiparametric estimation and consumer demand", Journal of Applied Econometrics, 13(5), 435–462.

References

Journal of Applied Econometrics data archive : http://qed.econ.queensu.ca/jae/

---

cor,stranger-method *Correlation for stranger objects*

Description

describeIn cor Correlation method for stranger objects.

Usage

```
## S4 method for signature 'stranger'
cor(x, method = c("pearson", "kendall", "spearman"))
```

Arguments

x	stranger object
method	see help from base cor function

crazyfy *Data preparation before detection of strangers*

Description

crazyfy preprocess data for anomalies detection computational routines with strange : missing values treatement, variables standardisation, eventual recoding in log, treatment of character/factor variables.

Usage

```
crazyfy(data, do = c("factor", "log", "impute", "range"), id = NULL,
  skewness.cutpoint = 2, NA.method = "mean", NA.value = 0,
  verbose = FALSE)
```

Arguments

data	Source data (data.frame or data.table).
do	character vector - List of processing steps to apply – see details.
id	(optional) character - name of a preexisting variable to be used as ID.
skewness.cutpoint	numeric - value that is used to determine whether log recoding should be applied.
NA.method	character - method to be used for missing values imputation; one of "mean" or "value" (then using following parameter NA.value).
NA.value	numeric Value to be used to impute missing values when NA.method if "value".
verbose	logical - should function display some details about processing.

Details

See here this list of possible pre-treatment operations. Factors/characters are transformed into numeric by using term frequency–inverse document frequency approach (td-idf). Note that we use the smooth weighting IDF weight, ie. we take the log of 1+N/nt where N is the number of observations and nt the frequency for the specific term t.

Value

Pre-processed data of classes data.table overloaded by crazy.data.table.

Examples

```
library(stranger)
data(iris)
crazy <- crazyfy(iris[,1:4])
```

explore *Data Exploration plots*

Description

This is a simple function that produced plots that may help you to understand the shape of your dataset. Today, 3 plots are implemented;(1) a small histogram for each features, (2) a correlation matrix and (3) a 2-Dimension Hexbin Frequency.

Usage

```
explore(data, type = "histogram", keep = NULL, drop = NULL, ...)
```

Arguments

data	is the data frame containing the observations. Each row represents an observation and each variable is stored in one column.
type	one of "histogram", "correlation"
keep	character vector: names of columns to keep (filter)
drop	character vector: names of columns to drop (filter)
...	is additional arguments to be passed to internal functions. Currently only col.fill and bins for histograms.

Examples

```
## Not run:
explore(BudgetUK, type ="hist")
explore(BudgetUK, type ="cor",drop="children")

## End(Not run)
```

filter.stranger *dplyr methods*

Description

Methods to use with dplyr filter function

Usage

```
## S3 method for class 'stranger'
filter(.data, ...)

## S3 method for class 'singular'
filter(.data, ...)
```

Arguments

.data	object (class: sranger or singular)
...	filter definition

`fortify.anomalies` *Merge stranger/singular objects to a dataset*

Description

`fortify` method for `anomalies`, `stranger` and `singular` objects to enrich data. Usually, this is invoked on source data. For obvious precautions, this is done on an id column as managed by the typical workflow (`crazyfy`). In case you enrich data without any specific id, it is just assumed to have those data in the same order than data used in anomaly detection computations – this behavior may really needs to unwanted results.

Usage

```
## S3 method for class 'anomalies'
fortify(x, data = NULL, id = NULL,
  colname = "flag_anomaly", ...)

## S3 method for class 'stranger'
fortify(x, data = NULL, id = NULL, all.x = TRUE,
  all.y = FALSE, ...)

## S3 method for class 'singular'
fortify(x, data = NULL, id = NULL, all.x = TRUE,
  all.y = FALSE, ...)
```

Arguments

x	anomalies objects as generated by a call to `get_anomalies`.
data	data to enrich; if NULL (default), we use the data used during computation
id	character - name of the id column. No need to specify this if you follow the recommended process using either first `add_id` or `crazyfy`.
colname	character - name of the column to be created for the flag (default: flag_anomaly).
...	fortify generic parameter – not used for stranger objects.
all.x	merge parameter
all.y	merge parameter

get_anomalies *Retrieve anomalies*

Description

Based on a summary normalized/stacked metric, retrieve top anomalies.

Usage

```
get_anomalies(x, rank.prop = 0.05, nmin = 10, nmax = 300,
  stack.use = "avg", method.use = "norm", verbose = TRUE, ...)
```

Arguments

x	stranger object (before of after singularize)
rank.prop	proportion of records to be considered as anomalies
nmin	constraint - minimum number of anomalies
nmax	constrait - maximum number of anomalies
stack.use	One of c("max","avg","min","damavg", "pruavg")) - must have been requested-when invoking 'singularize' (done by default).
method.use	One of c("norm","rank") - must have been requestedwhen invoking 'singularize' (done by default).
verbose	logical: provide some information.
...	additional parameters to pass to singularize (if called on a non-singularized object)
	Anomalies selection is performed using one summary metric. This summary metrics is assumed to stacked some base metrics - may be only one!. Stacking is performed after standardisation, being possible with two approaches: normalisation (method.use = "norm") or ranking (method.use = "rank"). See singularize function.
	Three parameters are used together to define anomalies: rank.prop is firsu used to filter on top x% anomalies then one applies on top of this criteria conditions on a minimal (nmin) and maximal (nmax) number of anomalies to be provided.

Examples

```
data <- crazyfy(iris[,1:4])
(anom <- get_anomalies(strange(data)))
## Not run:
library(dplyr)
ss <- iris %>% select(-Species) %>%
 crazyfy() %>%
 strange(weird="autoencode") %>%
 singularize(methods="norm",stacks="avg")
 anom2 <- ss %>% get_anomalies(nmin=2, nmax=4)
```

```
 ss %>% plot(type="n",score="N_anom_norm_avg",anomaly_id=anom2[1])

## End(Not run)
```

---

get_info *stranger object information*

Description

Retrieve some information on the content of an object built with stranger package.

Usage

```
get_info(x, ...)

## S3 method for class 'stranger'
get_info(x, simplify = TRUE, ...)
```

Arguments

x	One object from stranger package
...	Additional parameters - not used currently
simplify	boolean: simplify or keep output as list

Value

matrix - metrics attributes (corresponding weird info), one row per weird.

Examples

```
## Not run:
library(dplyr)
info <-  iris %>% select(-Species) %>% crazyfy() %>% stranger() %>% get_info()
info

## End(Not run)
```

get_info.singular *singularized metrics information*

Description

Retrieve some informaiton on the content of a singularize object. Note: always print (cat) information, store the content if you want to programmatically access it.

Usage

```
## S3 method for class 'singular'
get_info(x, ...)
```

Arguments

x	One object from stranger package
...	Additional parameters - not used currently

Value

(invisible) list with two componenents: metrics and standardizations. First slot consists in metrics attributes (corresponding weird info), second slot is a vector containing the names of aggregated standardized derived metrics.

Examples

```
## Not run:
library(dplyr)
info <-  iris %>% select(-Species) %>% crazyfy() %>% stranger() %>% singularize() %>% get_info()

## End(Not run)
```

lucky_odds *Quickly apply stanger full process flow to flag candidates anomalies applying one weird method*

Description

lucky_odds is basically a wrapper around the process: add_id –> crazyfy –> strange (weird method with set of parameters) –> singularize (default parameters: all methods) –> get_anomalies (flag top *n.anom* anomalies) –> fortify to enrich source data. By calling lucky_odds, analyst gets back source data with an additional column flagging some records. Though obviously simplifying the analysis process, not all options are available and intermediate objects are not available for further analysis.

Usage

```
lucky_odds(data, n.anom = 5, ..., analysis.drop = NULL,
  analysis.keep = NULL, weird = "knn", stack = "avg",
  stack.method = "norm")
```

Arguments

data	Source data (data.frame or data.table).
n.anom	Number of anomaly candidate records to flags.
...	Additional parameters to be passed to weird method (analysis.method).
analysis.drop	Character - set of variables to be removed from analysis (metrics computations by weird).
analysis.keep	Character - set of variables to be kept for analyis (metrics computations by weird).
weird	weird method to use for metric computation
stack	Stacking metric passed to get_anomalies.
stack.method	Stacking selection method passed to get_anomalies.

Selecting variables

If your source data contains variables you don't want to use in metrics computations - weird method: knnw, autoencode..., then you have to first select analysis variables. You can thos this use either analysis.keep OR analysis.drop. Those two parameters are mutually exclusive. library(stranger) data(iris) anomalies <- lucky_odds(iris[,1:4]) table(anomalies$flag_anomaly)

---

plot.stranger *Data visualizations of anomaly score locally around a specific data point*

Description

Data visualizations of anomaly score locally around a specific data point

Usage

```
## S3 method for class 'stranger'
plot(x, type = "cluster", id = ".id", score = NULL,
  anomaly_id = NULL, ...)

## S3 method for class 'fortifiedanomaly'
plot(x, type = "feature_importance", id = ".id",
  anomaly_id = NULL, score = NULL, ...)

## S3 method for class 'anomalies'
plot(x, type = "feature_importance", id = ".id",
```

```
  anomaly_id = NULL, ...)

## S3 method for class 'singular'
plot(x, type = "cluster", id = ".id", score = NULL,
  anomaly_id = NULL, ...)
```

Arguments

x	is either of class dataframe, stranger or anomaly. It contains the observations; each row represents an observation and each variable is stored in one column. It must have at least one column with IDs and one column with the anomaly score for each ID.
type	is the name of the visualization; (1) A hierarchical clustering, named "cluster", showing among the top n-anomaly which records belongs to the same cluster a specific record. Finding the commun pattern amoung the cluster may lead to the orign of of the specifi record score. (2) A dots plot, named "neighbours", showing the relationship between the anomly score and each feature for the k nearest neighbours of a specific record. (3) A bar chart, named "feature_importance", showing how sensitive is the anomaly score of a specific record to each of feature. This may help to identify the features behind the score. (4) A dots plot, names "score_decline", showing the decrease in anomaly score among the k nearest neighbours of a specific record. The shape indicates how extrem and how frequent is the anomaly score of a speicif record amoung its neighbours. (5) A Regression tree, named "regression_tree", showing the roots to high score around a specific record.
id	is the colname with records IDs
score	is the colname which contains the anomaly score
anomaly_id	is the record ID you want to investigate
...	Additional parameters to pass

Details

Function that produces visualizations to understand the anomaly score locally around a specific data point. We believe this should help people to trust scores a made by models even if they don't fully understand them. Today, 5 visualisazions are implemented; (1) A hierarchical clustering, named "cluster", showing among the top n-anomaly which records belongs to the same cluster a specific record. Finding the commun pattern amoung the cluster may lead to the orign of of the specifi record score. (2) A dots plot, named "neighbours", showing the relationship between the anomly score and each feature for the k nearest neighbours of a specific record. (3) A bar chart, named "feature_importance", showing how sensitive is the anomaly score of a specific record to each of feature. This may help to identify the features behind the score. (4) A dots plot, names "score_decline", showing the decrease in anomaly score among the k nearest neighbours of a specific record. The shape indicates how extrem and how frequent is the anomaly score of a speicif record amoung its neighbours. (5) A Regression tree, named "regression_tree", showing the roots to high score around a specific record.

Extra parameters that can be used in ...:

- check logical indicating if object data should be checked for validity. The default is TRUE, this check is not necessary when data is known to be valid such as when it is the direct result of stranger().
- keep character vector: names of columns to keep (filter)
- drop character vector: names of columns to drop (filter)
- n.cluster is the number of cluster groups to emphasis. This parameter must only be specified with type ="cluster".
- n.anom is the number of top anomalies to be considered. This parameter must only be specified with type ="cluster".
- k is the number of neighbours to be considered. This parameter must always be specified, except with type = "cluster".
- n_label specifies the number of data point to be labelled in the plot. This parameter must only be specified with type ="scores_decline".

Value

A plot

`singularize` *Normalize anomalies metrics and (eventually) stack them*

Description

`singularize` derives normalized/standardized versions of different weirds contained in a stranger object so that they are directly comparable and then propose various aggregated measures (stacking).

Usage

```
singularize(strangerObject, methods = c("norm", "rank"), stacks = c("max",
  "avg", "min", "damavg", "pruavg"), prefix = "N_anom", ...)
```

Arguments

`strangerObject`	An object build with `stranger` or `strangest`
`methods`	character vector of standardization methods: norm (normalize) and rank (ranking version). By default, both methods are used.
`stacks`	character vectors of stacked versions. Note that some methods require more than 2 weirds metrics to be used
`prefix`	default prefix used to name generated new metrics
`...`	Additional parameters - Currently not used.

strange *Computes anomaly metrics by invoking specific method(s) with associated sets of parameters*

Description

strange invokes a *weird* method – that is a wrapper around a pre-existing anomaly detection measure (distance, probability...). stranger allows to invoke several *weird* methods in a single call.

Usage

```
strange(data, weird = "knn", tuneGrid = NULL, colname = NULL, ...)
```

Arguments

data	crazy data, ie outcome of a call to crazyfy.
weird	Weird method to be used - for the list of available methods, use weird_list.
tuneGrid	(optional) vector or data.frame of values for the parameters of the invoked method.
colname	(optional) character - name to be given to the resulting anomaly metric computation (distance/probability).
...	additional parameters to be passed to the invoked *weird* method.

Details

You will use strange to use one method and may be interested by stranger if you wants to apply different methods in a single call. When comparing stranger package with caret, strange function is the equivalent of train, where as stranger corresponds to caretEnsemble function and package.

Value

stranger object – that is a data.table with attributes and overloaded with class stranger

metadata

<TBD>

Examples

```
## Not run:
library(stranger)
data(iris)
crazydata <- crazyfy(iris[,1:4])
curious <- strange(crazydata, method="knn")

## End(Not run)
```

weird *Define a call to a weird function*

Description

weirds function are wrapper around other packages anomalies detection routines. `weird` function is intended to be used when invoking `stranger` so that several anomalies detection routines are used at one time.

Usage

```
weird(method, ...)
```

Arguments

`method`	- weird method to be used, check list of available methods by using `weirds_list`
`...`	- additional parameters to be passed to weird method

Value

list with class weirdSpecs to be used as value when using the parameter `tuneList` of `stranger` function.

For every `weird` function, possible parameters in ... should be checked by looking to the help of the underlying function. Some special common parameters are `info` to be set to TRUE to have information about the method, inclusing the name and package of the main underlying function. Another common parameter is `colname` that allow to overide default prefix name used for the outcome of the computation.

Methods specific parameters, additional parameters may be used (for instance `simplify` for _knn_ method or `type` for kmeans. See the vignette describing the list of available _weirds_ methods in base package.

weirds_list *Provides the list of available methods*

Description

Provides the list of available methods

Usage

```
weirds_list(onlynames = FALSE)
```

Arguments

`onlynames`	- logical: should be return some details or only the nickname of methods to be used in `strange`, `stranger`, `weird` or `lucky_odds`.

weird_knn *weirdness wrappers for available anomalies detection methods*

Description

Those wrapper are mainly used internally by other functions and the recommended way is to start your detection process by using one of: strange, stranger or lucky_odds functions.

Usage

```
weird_knn(data = NULL, info = FALSE, colname = NULL, simplify = "mean",
  ...)

weird_lof(data = NULL, info = FALSE, colname = NULL, ...)

weird_autoencode(data = NULL, info = FALSE, colname = NULL, ...)

weird_abod(data = NULL, info = FALSE, colname = NULL, ...)

weird_pcout(data = NULL, info = FALSE, colname = NULL, ...)

weird_isofor(data = NULL, info = FALSE, colname = NULL, ...)

weird_kmeans(data = NULL, info = FALSE, colname = NULL, type = "means",
  ...)

weird_mahalanobis(data = NULL, info = FALSE, colname = NULL, ...)

weird_randomforest(data = NULL, info = FALSE, colname = NULL, ...)
```

Arguments

data	data pre-processed with crazyfy function
info	logical: either run the function (default) or simply return some information about it
colname	character: optional prefix to be used to prepare the naming of the generated column. All weird methods do have a default prefix (applied when colname is NULL). Note that generated output column name will still have some information added to reflect values used for some key parameters.
simplify	character of function - used by knn to aggregates distances to nearest k points.
...	additional parameters to be passed to weird method, for instance k for knn
type	one of "means" or "euclidian". See kmeans from stats package.

Details

For each weird wrapper, please refer to the documentation of the underlying function in its package.

Index