Package 'rsubgroup'

April 22, 2020

Type Package

Title Subgroup Discovery and Analytics

Version 1.0

Date 2020-04-20

Author Martin Atzmueller

Maintainer Martin Atzmueller <martin@atzmueller.net>

Description A collection of efficient and effective tools and algorithms for subgroup discovery and analytics. The package integrates an R interface to the org.vikamine.kernel library of the VIKAMINE system <http://www.vikamine.org> implementing subgroup discovery, pattern mining and analytics in Java.

Classification/ACM G.4, H.2.8, I.5.1

License GPL (>= 3)

Depends R (>= 2.6.0), methods, rJava (>= 0.6-3), foreign (>= 0.8-40)

SystemRequirements Java (>= 8)

Collate 'AAAonLoad.R' 'randomSeed.R' 'classes.R' 'subgroup.R'

URL http://www.rsubgroup.org

NeedsCompilation no

Repository CRAN

Date/Publication 2020-04-22 14:20:02 UTC

R topics documented:

`as.target` *Constructs a target variable (for subgroup discovery)*

Description

Constructs a target variable, i.e., an object suitable to be passed to DiscoverSubgroups or CreateSDTask.

Usage

```
as.target(attribute, value=NULL)
```

Arguments

`attribute`	The attribute of the target variable.
`value`	For binary targets, the respective attribute value; the value is NULL for numeric targets.

See Also

`DiscoverSubgroups`.

Examples

```
# creating a target variable
# binary:
as.target("class", "true")

#numeric:
as.target("numeric_class")
```

CreateSDTask *Creates a Subgroup Discovery Task*

Description

Performs subgroup discovery according to the given task.

Usage

```
CreateSDTask(source, target, config = SDTaskConfig())
```

Arguments

source	a data.frame or the a character string giving the filename of an ARFF file to use. Providing a file name directly provides the data to the subgroup discovery algorithms on the Java side, which is more memory efficient than converting the data frame to the Java representation.
target	the target variable (constructed by as.target) to consider for subgroup discovery.
config	an instance of SDTaskConfig providing various parameters for subgroup discovery.

See Also

DiscoverSubgroups. DiscoverSubgroupsByTask SDTaskConfig

Examples

```
# creating a task
data(credit.data)

# task with binary target
task <- CreateSDTask(credit.data, as.target("class", "good"))

# task with numeric target
taskNum <- CreateSDTask(credit.data, as.target("credit_amount"))
```

credit.data *Statlog (German Credit Data) Data Set*

Description

This dataset classifies people described by a set of attributes as good or bad credit risks.

Usage

```
data(credit.data)
```

Format

A vector containing 1000 observations.

Source

UCI Repository, https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data).

DiscoverSubgroups *Performs Subgroup Discovery*

Description

Performs subgroup discovery according to the given target and the configuration on the data.

Usage

```
DiscoverSubgroups(source, target, config= SDTaskConfig(), as.df=FALSE)
```

Arguments

source	a data.frame or the a character string giving the filename of an ARFF file to use. Providing a file name directly provides the data to the subgroup discovery algorithms on the Java side, which is more memory efficient than converting the data frame to the Java representation.
target	the target variable (constructed by as.target) to consider for subgroup discovery.
config	an instance of SDTaskConfig providing various parameters for subgroup discovery.
as.df	TRUE, if the result patterns should be returned as a data.frame using ToDataFrame

See Also

DiscoverSubgroupsByTask. as.target CreateSDTask SDTaskConfig

Examples

```
# subgroup discovery on a data.frame, for binary target
data(credit.data)
result1 <- DiscoverSubgroups(
    credit.data, as.target("class", "good"), new("SDTaskConfig",
    attributes=c("checking_status", "credit_amount", "employment", "purpose")))
result2 <- DiscoverSubgroups(
    credit.data, as.target("class", "good"), new("SDTaskConfig",
    attributes=c("checking_status", "employment")))

ToDataFrame(result1)
ToDataFrame(result2)
```

```
# subgroup discovery for numeric target variable
result3 <- DiscoverSubgroups(
    credit.data, as.target("credit_amount"), new("SDTaskConfig",
    attributes=c("checking_status", "employment")))

ToDataFrame(result3)
```

DiscoverSubgroupsByTask
Performs Subgroup Discovery for a given Task

Description

Performs subgroup discovery according to the given task.

Usage

```
DiscoverSubgroupsByTask(task, as.df=FALSE)
```

Arguments

task	a subgroup discovery task constructed by CreateSDTask.
as.df	TRUE, if the result patterns should be returned as a data.frame using ToDataFrame

See Also

DiscoverSubgroups. CreateSDTask

Examples

```
# creating a task
data(credit.data)
task <- CreateSDTask(
    credit.data, as.target("class", "bad"), SDTaskConfig(
    attributes=c("checking_status", "employment")))
taskNum <- CreateSDTask(
    credit.data, as.target("credit_amount"), SDTaskConfig(
    attributes=c("checking_status", "employment")))

# running the tasks
DiscoverSubgroupsByTask(task)
DiscoverSubgroupsByTask(taskNum)
```

is.pattern.matching *Tests whether a pattern and a data list (row of a data frame) match*

Description

Tests whether a pattern and a data list (row of a data frame) match, e.g., for implementing classification methods.

Usage

```
is.pattern.matching(pattern, data.list)
```

Arguments

pattern	An instance of class Pattern, e.g., returned by DiscoverSubgroups.
data.list	A list having the attributes as 'keys', and the values as respective values of the list. This corresponds, for example, to a row of a data frame.

See Also

Pattern-class.

Pattern-class *Class "Pattern" — A Simple Subgroup Description Container*

Description

A Simple Container holding the results (subgroups, description and parameters) for the Subgroup and Pattern Mining Algorithms

Objects from the Class

Objects are created by calls of the form `new("Pattern",...)`.

Slots

description: The subgroup description, as a character vector.

selectors: The subgroup description, given as a list of (simple) selection expressions, where the 'key' is the attribute and the 'value' is the value.

quality: The numeric value denoting the quality of the subgroup pattern as determined by the applied quality function.

size: The size of the subgroup.

parameters Additional quality parameters of the subgroup.

See Also

DiscoverSubgroups. DiscoverSubgroupsByTask CreateSDTask

rsubgroup *rsubgroup Package - Algorithms and Tools for Efficient Subgroup Discovery and Analytics*

Description

The rsubgroup package contains a set of efficient and effective tools and algorithms for subgroup discovery and analytics. The package integrates an R interface to the org.vikamine.kernel library of the VIKAMINE system (http://www.vikamine.org).

Note: rsubgroup uses rJava. To set the maximum available heap space for Java, the .jinit command of rJava needs to be called before loading rsubgroup, i.e.

library(rJava) .jinit(parameters="-Xmx2048M") # for two gigabytes heap space, for example library(rsubgroup)

Please note that this needs to happen before rJava is used in any way. After the JVM has been initialized (and started), setting the heap space has no effect any more. Therefore, it is recommended to execute the .jinit(...) command right after loading the rJava package.

Details

Package:	rsubgroup
Type:	Package
Version:	0.7
Date:	2015-07-xx
License:	GPL (>= 3)
LazyLoad:	yes

Author(s)

Martin Atzmueller

Maintainer: Martin Atzmueller <martin@atzmueller.net>

References

1. Martin Atzmueller and Frank Puppe. SD-Map - A Fast Algorithm for Exhaustive Subgroup Discovery. Knowledge Discovery in Databases: PKDD 2006, LNAI 4213, pp. 6-17, Springer Verlag, 2006.
2. Martin Atzmueller and Florian Lemmerich. Fast Subgroup Discovery for Continuous Target Concepts. In: Foundations of Intelligent Systems, LNCS 5722, pp. 35-44, Springer Verlag, 2009.
3. Florian Lemmerich and Mathias Rohlfs and Martin Atzmueller. Fast Discovery of Relevant Subgroup Patterns. In: Proc. 23rd FLAIRS Conference, AAAI Press, 2010.

SDTaskConfig *Creates a Subgroup Discovery Task Configuration*

Description

Creates a subgroup discovery task configuration, that is, an instance of SDTaskConfig.

SDTaskConfig-class *Class "SDTaskConfig" — A Set of Configuration Settings*

Description

A Set of Configuration Settings for the Subgroup and Pattern Mining Algorithms

Objects from the Class

Objects are created by calls of the form `SDTaskConfig(...)`.

Slots

`attributes`: The list of attributes to consider for mining. Either a vector of attribute names, or NULL (the default), which includes all attributes.

`method`: A mining method; one of Beam-Search `beam`, BSD `bsd`, SD-Map `sdmap`, SD-Map enabling internal disjunctions `sdmap-dis`. The default is `method = "sdmap"`.

`qf`: A quality function; one of: Adjusted Residuals `ares`, Binomial Test `bin`, Chi-Square Test `chi2`, Gain `gain`, Lift `lift`, Piatetsky-Shapiro `ps`, Relative Gain `relgain`, Weighted Relative Accuracy `wracc`. The default is `qf = "ps"`.

`k`: The maximum number (top-k) of patterns to discover, i.e., the best k rules according to the selected quality function. The default is `k = 20`

`minqual`: The minimal quality (default `minqual = 0`).

`minsize`: The minimal size of a subgroup (as an integer) (minimal coverage of database records, default `minsize = 0`).

`mintp`: The minimal true positive (tp) threshold, an integer (minimal (absolute) number of true positives in a subgroup, relevant for binary target concepts only), defaults to `mintp = 0`.

`maxlen`: The maximal length of a description of a pattern, i.e., the maximal number of conjunctions. This impacts both understandability and efficiency. Simpler rules are easier to understand, and a small `maxlen` will restrict the search space (default `maxlen = 7`).

`nodefaults`: Ignore default values, i.e., do not include the respective first value (with index 0) of each attribute (default `nodefaults=FALSE`, i.e., include all values).

`relfilter`: Controls, whether irrelevant patterns are filtered during pattern mining; negatively impacts performance (default `relfilter = FALSE`)).

`postfilter`: Controls, whether a post-processing filter is applied; one (or a vector) of: Minimum Improvement (Global) `min-improve-global`, checks the patterns against all possible generalizations, Minimum Improvement (Pattern Set) `min-improve-set`, checks the patterns against all their generalizations in the result set, Relevancy Filter `relevancy`, removes patterns that are strictly irrelevant, Significant Improvement (Global) `sig-improve-global`, removes patterns that do not significantly improve (default 0.01 level) w.r.t. all their possible generalizations, Significant Improvement (Set) `sig-improve-set`, removes patterns that do not significantly improve (default 0.01 level) w.r.t. all generalizations in the result set, Weighted Covering `weighted-covering`, performs weighted covering on the data in order to select a covering set of subgroups while reducing the overlap on the data. By default no postfilter is set, i.e., `postfilter = ""`.

`parfilter`: Provides the minimal improvement value for the postfilter (for min-improve-* filters), or the significance level (P) for sig-improve-* filters.

See Also

`DiscoverSubgroups`. `DiscoverSubgroupsByTask` `CreateSDTask`

`ToDataFrame` *Transforms patterns into a data frame*

Description

Transforms a list/vector of patterns into a data frame for inspection and analysis.

Usage

```
ToDataFrame(patterns, ndigits = 2)
```

Arguments

`patterns`	List/vector of patterns.
`ndigits`	Number of significant digits when printing floats (optional).

See Also

`DiscoverSubgroups`.

Index