Package 'niarules'

March 9, 2024

Type Package

Title Numerical Association Rule Mining using Population-Based Nature-Inspired Algorithms

Version 0.1.0

Classification/ACM G.4, H.2.8

Description Framework is devoted to mining numerical association rules through the utilization of nature-inspired algorithms for optimization. Drawing inspiration from the 'NiaARM' 'Python' and the 'NiaARM' 'Julia' packages, this repository introduces the capability to perform numerical association rule mining in the R programming language.
Fister Jr., Iglesias, Galvez, Del Ser, Osaba and Fister (2018) <doi:10.1007/978-3-030-03493-1_9>.

URL https://github.com/firefly-cpp/niarules

BugReports https://github.com/firefly-cpp/niarules/issues

Depends R (>= 4.0.0)

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.0

Imports stats, utils

Suggests testthat

NeedsCompilation no

Author Iztok Jr. Fister [aut, cre, cph]
(<https://orcid.org/0000-0002-6418-1272>)

Maintainer Iztok Jr. Fister <iztok@iztok.space>

Repository CRAN

Date/Publication 2024-03-09 11:30:02 UTC

R topics documented:

`add_association_rule` *Add an association rule to the list of rules.*

Description

This function adds a new association rule to the existing list of rules.

Usage

```
add_association_rule(
  rules,
  antecedent,
  consequence,
  support,
  confidence,
  fitness
)
```

Arguments

rules	The current list of association rules.
antecedent	The antecedent part of the association rule.
consequence	The consequent part of the association rule.
support	The support of the association rule.
confidence	The confidence of the association rule.
fitness	The fitness of the association rule.

Value

The updated list of association rules.

---

add_attribute *Add an attribute to the "rule" list.*

Description

This function adds an attribute to the existing list.

Usage

```
add_attribute(rules, name, type, border1, border2, value)
```

Arguments

rules	The current rules list.
name	The name of the feature in the rule.
type	The type of the feature in the rule.
border1	The first border value in the rule.
border2	The second border value in the rule.
value	The value associated with the rule.

Value

The updated rules list.

Examples

```
rules <- list()
new_rules <- add_attribute(rules, "feature1", "numerical", 0.2, 0.8, "EMPTY")
```

`build_rule` *Build rules based on a candidate solution.*

Description

This function takes a candidate solution vector and a features list and builds rule.

Usage

```
build_rule(solution, features)
```

Arguments

`solution`	The solution vector.
`features`	The features list.

Value

A rule.

`calculate_border` *Calculate the border value based on feature information and a given value.*

Description

This function calculates the border value for a feature based on the feature information and a given value.

Usage

```
calculate_border(feature_info, value)
```

Arguments

`feature_info`	Information about the feature.
`value`	The value to calculate the border for.

Value

The calculated border value.

Examples

```
feature_info <- list(type = "numerical", lower_bound = 0, upper_bound = 1)
border_value <- calculate_border(feature_info, 0.5)
```

calculate_fitness *Calculate the fitness of an association rule.*

Description

This function calculates the fitness of an association rule using support and confidence.

Usage

```
calculate_fitness(supp, conf)
```

Arguments

supp	The support of the association rule.
conf	The confidence of the association rule.

Value

The fitness of the association rule.

calculate_selected_category *Calculate the selected category based on a value and the number of categories.*

Description

This function calculates the selected category based on a given value and the total number of categories.

Usage

```
calculate_selected_category(value, num_categories)
```

Arguments

value	The value to calculate the category for.
num_categories	The total number of categories.

Value

The calculated selected category.

Examples

```
selected_category <- calculate_selected_category(0.3, 5)
```

check_attribute *Check if the attribute conditions are satisfied for an instance.*

Description

This function checks if the attribute conditions specified in the association rule are satisfied for a given instance row.

Usage

```
check_attribute(attribute, instance_row)
```

Arguments

attribute	An attribute with type and name information.
instance_row	A row representing an instance in the dataset.

Value

TRUE if conditions are satisfied, FALSE otherwise.

cut_point *Calculate the cut point for an association rule.*

Description

This function calculates the cut point, denoting which part of the vector belongs to the antecedent and which to the consequence of the mined association rule.

Usage

```
cut_point(sol, num_attr)
```

Arguments

sol	The cut value from the solution vector.
num_attr	The number of attributes in the association rule.

Value

The cut point value.

`differential_evolution` *Implementation of Differential Evolution metaheuristic algorithm.*

Description

This function uses Differential Evolution, a stochastic population-based optimization algorithm, to find the optimal numerical association rule.

Usage

```
differential_evolution(
  D = 10,
  NP = 10,
  F = 0.5,
  CR = 0.9,
  nfes = 1000,
  features,
  data
)
```

Arguments

`D`	Dimension of the problem (default: 10).
`NP`	Population size (default: 10).
`F`	The differential weight, controlling the amplification of the difference vector (default: 0.5).
`CR`	The crossover probability, determining the probability of a component being replaced (default: 0.9).
`nfes`	The maximum number of function evaluations (default: 1000).
`features`	A list containing information about features, including type and bounds.
`data`	A data frame representing instances in the dataset.

Value

A list containing the best solution, its fitness value, and the number of function evaluations and list of identified association rules.

evaluate *Evaluate a candidate solution.*

Description

This function takes a canditate solution (vector), list of features, and instances, and evaluates the fitness of an association rule by calculating support and confidence.

Usage

```
evaluate(solution, features, instances)
```

Arguments

solution	A vector representing a candidate solution for the association rule.
features	A list containing information about features, including type and bounds.
instances	A data frame representing instances in the dataset.

Value

The fitness of the association rule and identified rule.

extract_feature_info *Extract feature information from a dataset.*

Description

This function analyzes the given dataset and extracts information about each feature.

Usage

```
extract_feature_info(data)
```

Arguments

data	The dataset to analyze.

Value

A list containing information about each feature.

feature_borders *Get the lower and upper bounds of a feature.*

Description

This function retrieves the lower and upper bounds of a feature from the features list.

Usage

```
feature_borders(features, name)
```

Arguments

features	A list containing information about features, including type and bounds.
name	The name of the feature.

Value

A list containing the lower and upper bounds of the feature.

feature_position *Get the position of a feature.*

Description

This function returns the position of a feature in the vector, considering the type of the feature.

Usage

```
feature_position(features, feature)
```

Arguments

features	The features list.
feature	The name of the feature to find.

Value

The position of the feature.

Examples

```
features <- list(
  feature1 = list(type = "numerical"),
  feature2 = list(type = "categorical"),
  feature3 = list(type = "numerical")
)
position <- feature_position(features, "feature2")
```

`fixBorders` *Fix Borders of a Numeric Vector*

Description

This function takes a numeric vector as input and ensures that all values greater than 1.0 are set to 1.0, and all values less than 0.0 are set to 0.0.

Usage

```
fixBorders(vector)
```

Arguments

`vector`	A numeric vector to be processed.

Value

A numeric vector with borders fixed. Values greater than 1.0 are replaced with 1.0, and values less than 0.0 are replaced with 0.0.

Examples

```
original_vector <- c(1.19007417, 0.33135271, -0.5, 1.5, 0.0)
fixed_vector <- fixBorders(original_vector)
print(fixed_vector)
```

print_association_rules *Print Numerical Association Rules*

Description

This function prints association rules including antecedent, consequence, support, confidence, and fitness.

Usage

```
print_association_rules(rules)
```

Arguments

rules	A list containing association rules.

Value

Prints the association rules.

print_feature_info *Print feature information extracted from a dataset.*

Description

This function prints the information extracted about each feature.

Usage

```
print_feature_info(feature_info)
```

Arguments

feature_info	The list containing information about each feature.

Value

A message is printed to the console for each feature, providing information about the feature's type, and additional details such as lower and upper bounds for numerical features or categories for categorical features. No explicit return value is generated.

print_rule_parts *Print Rule Parts*

Description

This function prints the parts of an association rule, including name, type, border1, border2, and value.

Usage

```
print_rule_parts(parts)
```

Arguments

parts	A list containing parts of an association rule.

Value

Prints the rule parts.

problem_dimension *Calculate the dimension of the problem based on feature information.*

Description

This function takes a list of feature information and calculates the dimension based on the type of each feature. Method is inspired by referenced paper.

Usage

```
problem_dimension(feature_info)
```

Arguments

feature_info	A list containing information about each feature.

Value

The calculated dimension based on the feature types.

References

Fister, I., Iglesias, A., Galvez, A., Del Ser, J., Osaba, E., & Fister, I. (2018). Differential evolution for association rule mining using categorical and numerical attributes. In *Intelligent Data Engineering and Automated Learning–IDEAL 2018: 19th International Conference, Madrid, Spain, November 21–23, 2018, Proceedings, Part I* (pp. 79–88). Springer.

read_dataset *Read a dataset from a CSV file.*

Description

This function reads a CSV file and returns the dataset as a data frame.

Usage

```
read_dataset(dataset_path)
```

Arguments

dataset_path	The path to the CSV file.

Value

A data frame representing the dataset.

rs *Simple Random Search*

Description

This function generates a vector of random solutions for a specified length.

Usage

```
rs(candidate_len)
```

Arguments

candidate_len	The length of the vector of random solutions.

Value

A vector of random solutions between 0 and 1.

Examples

```
candidate_len <- 10
random_solutions <- rs(candidate_len)
print(random_solutions)
```

`supp_conf` *Calculate support and confidence for an association rule.*

Description

This function calculates the support and confidence for the given antecedent and consequent in the dataset instances.

Usage

```
supp_conf(antecedent, consequent, instances, features)
```

Arguments

`antecedent`	The antecedent part of the association rule.
`consequent`	The consequent part of the association rule.
`instances`	A data frame representing instances in the dataset.
`features`	A list containing information about features, including type and bounds.

Value

A list containing support and confidence values.

`write_association_rules_to_csv`
Write Association Rules to CSV file

Description

This function writes association rules to a CSV file.

Usage

```
write_association_rules_to_csv(rules, file_path)
```

Arguments

`rules`	A list of association rules.
`file_path`	The file path for the CSV output.

Value

No explicit return value. The function writes association rules to a CSV file specified by the ‘file_path‘ parameter. A message is printed to the console indicating the successful completion of the writing process.

Index