Package ‘FSinR’

March 8, 2020

Maintainer Alfonso Jiménez-Vílchez <i52jivia@uco.es>

Type Package

Title Feature Selection

Description Feature subset selection algorithms modularized in search algorithms and measure utilities. Full list and more information available at <https://dicits.ugr.es/software/FSinR/>.

Version 1.0.8

Date 2020-02-11

Repository CRAN

License GPL-3

LazyData false

Imports rpart, neuralnet, class, digest, caret, mlbench, Rdpack, GA, dplyr, tidyr, prodlim, rlang, purrr, e1071

RdMacros Rdpack

Encoding UTF-8

RoxygenNote 7.0.2

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Alfonso Jiménez-Vílchez [aut, cre],
Francisco Aragón-Royón [aut],
Adan M. Rodriguez [aut],
Antonio Arauzo-Azofra [aut],
José Manuel Benítez [aut]

Date/Publication 2020-03-08 10:10:02 UTC

R topics documented:

aco *Ant Colony Optimization (Advanced Binary Ant Colony Optimization)*

Description

The Ant Colony Optimization (Advanced Binary Ant Colony Optimization) (Kashef and Nezamabadi-pour 2015) algorithm consists of generating in each iteration a random population of individuals (ants) according to the values of a pheromone matrix (which is updated each iteration according to the paths most followed by the ants) and a heuristic (which determines how good is each path to follow by the ants). The evaluation measure is calculated for each individual. The algorithm ends once the established number of iterations has been reached

Usage

```
aco(
  data,
  class,
  featureSetEval,
  population = 10,
  iter = 10,
  a = 1,
  b = 1,
  p = 0.2,
  q = 1,
  t0 = 0.2,
  tmin = 0,
  tmax = 1,
  mode = 1,
  verbose = FALSE
)
```

Arguments

`data`	• A data frame with the features and the class of the examples. All features must contain numerical values and not character, boolean, or factor type values since heuristics work only with numerical values. Otherwise the algorithm will generate error.
`class`	• The name of the dependent variable
`featureSetEval`	• The measure for evaluate features
`population`	• The number of ants population
`iter`	• The number of iterations
`a`	• Parameter to control the influence of the pheromone (If a=0, no pheromone information is used)
`b`	• Parameter to control the influence of the heuristic (If b=0, the attractiveness of the movements is not taken into account)

p	• Rate of pheromone evaporation
q	• Constant to determine the amount of pheromone deposited by the best ant. This amount is determined by the Q/F equation (for minimization) where F is the cost of the solution (F/Q for maximization)
t0	• Initial pheromone level
tmin	• Minimum pheromone value
tmax	• Maximum pheromone value
mode	• Heuristic information measurement. 1 -> min redundancy (by default). 2-> max-relevance and min-redundancy. 3-> feature-feature. 4-> based on F-score
verbose	• Print the partial results in each iteration

Value

A list is returned containing for each repetition of the algorithm:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

antsIter List that contains as many elements as iterations has the algorithm. Each of the elements in the list are matrices that represent the population in that iteration. In this matrix the individuals and the evaluation measure of each one are shown

pheromoneIter List that contains as many elements as iterations have the algorithm. Each of the elements in the list are matrices that represent the amount of pheromone between the paths of the different features (the reading of the matrix is from the columns to the rows, i.e. from top to bottom) in each iteration

Author(s)

Francisco Aragón Royón

References

Kashef S, Nezamabadi-pour H (2015). "An advanced ACO algorithm for feature subset selection." *Neurocomputing*, **147**, 271 - 279. ISSN 0925-2312, doi: 10.1016/j.neucom.2014.06.067, Advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012), http://www.sciencedirect.com/science/article/pii/S0925231214008601.

Examples

```
## Ant Colony Optimization for iris dataset (filter method)
aco(iris, 'Species', roughsetConsistency, population = 10, iter = 5, verbose = TRUE)
```

`binaryConsistency` *Binary consistency measure*

Description

Calculates the binary consistency, also known as "Sufficiency test" from FOCUS (Almuallim and Dietterich 1991)

Usage

```
binaryConsistency(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples. Feature columns are expected to be factors, as all features should be discrete.
`class`	• The name of the dependent variable
`features`	• The names of the selected features

Value

- The consistency value for the selected features

Author(s)

Adan M. Rodriguez

References

Almuallim H, Dietterich TG (1991). “Learning With Many Irrelevant Features.” In *In Proceedings of the Ninth National Conference on Artificial Intelligence*, 547–552.

Examples

```
binaryConsistency(iris,'Species',c('Sepal.Width', 'Sepal.Length'))
```

breadthFirstSearch *Exhaustive Search. Breadth First Search*

Description

The method searches the whole features subset in breadth first order (Kozen 1992)

Usage

```
breadthFirstSearch(data, class, featureSetEval)
```

Arguments

data	• A data frame with the features and the class of the examples
class	• The name of the dependent variable
featureSetEval	• The measure for evaluate features

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

Author(s)

Adan M. Rodriguez

Francisco Aragón Royón

References

Kozen DC (1992). *Depth-First and Breadth-First Search*. Springer New York, New York, NY. ISBN 978-1-4612-4400-4, doi: 10.1007/9781461244004_4, https://doi.org/10.1007/978-1-4612-4400-4_4.

Examples

```
## Breadth First Search for iris dataset (filter method)
breadthFirstSearch(iris, 'Species', binaryConsistency)
```

`chiSquared` *Chi squared measure*

Description

Calculates the Chi squared value (F.R.S. 1900), evaluating the selected features individually

Usage

```
chiSquared(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`features`	• The feature or features to evalute individually

Value

- The chi squared value for each selected feature

References

F.R.S. KP (1900). "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling." *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **50**(302), 157-175. doi: 10.1080/14786440009463897, https://doi.org/10.1080/14786440009463897 , `https://doi.org/10.1080/14786440009463897`.

Examples

```
chiSquared(iris,'Species','Sepal.Length')
```

`cramer` *Cramer V measure*

Description

Calculates Cramer's V value (Cramer 1946), evaluating features individually

Usage

```
cramer(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`features`	• The feature or features to evalute individually

Value

- Cramer's V value for each selected feature

References

Cramer H (1946). *Mathematical methods of statistics / by Harald Cramer* . Princeton University Press Princeton . ISBN ISBN 0-691-08004-6.

Examples

```
cramer(iris,'Species','Sepal.Length')
```

`deepFirstSearch` *Exhaustive Search. Deep First Search*

Description

The method searches the whole features subset in deep first order (Kozen 1992)

Usage

```
deepFirstSearch(data, class, featureSetEval)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureSetEval`	• The measure for evaluate features

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

Author(s)

Francisco Aragón Royón

References

Kozen DC (1992). *Depth-First and Breadth-First Search*. Springer New York, New York, NY. ISBN 978-1-4612-4400-4, doi: 10.1007/9781461244004_4, https://doi.org/10.1007/978-1-4612-4400-4_4.

Examples

```
## Deep First Search for iris dataset (filter method)
deepFirstSearch(iris, 'Species', binaryConsistency)
```

`determinationCoefficient` *R Squared, to continous features*

Description

This measure calculates the determinantion coefficient (Dodge 2008) of continuous features

Usage

```
determinationCoefficient(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`features`	• The names of the selected features

Value

- The R squared value for the selected features

Author(s)

Adan M. Rodriguez

References

Dodge Y (2008). *Coefficient of Determination*. Springer New York, New York, NY. ISBN 978-0-387-32833-1, doi: 10.1007/9780387328331_62, https://doi.org/10.1007/978-0-387-32833-1_62.

Examples

```
determinationCoefficient(iris,'Species',c('Sepal.Width', 'Sepal.Length'))
```

`entropy` *Entropy*

Description

Calculates the entropy value, using the information theory.

Usage

```
entropy(x)
```

Arguments

x	• Collection of values

Value

- Entropy value

Author(s)

Adan M. Rodriguez

Alfonso Jiménez-Vílchez

Examples

```
entropy(iris$Sepal.Length)
```

`entropyJ` *EntropyJ*

Description

Calculates the entropyJ value, using the information theory.

Usage

```
entropyJ(x)
```

Arguments

x	• Collection of values

Value

- EntropyJ value

Author(s)

Adan M. Rodriguez

Alfonso Jiménez-Vílchez

Examples

```
entropyJ(iris$Sepal.Length)
```

`fscore` *F-score measure*

Description

Evaluates a feature using the F-score approach defined in (Wang et al. 2018).

Usage

```
fscore(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`features`	• The name of the selected feature (only 1 feature)

Value

- The value of the function for the selected feature

References

Wang D, Zhang Z, Bai R, Mao Y (2018). “A hybrid system with filter approach and multiple population genetic algorithm for feature selection in credit scoring.” *Journal of Computational and Applied Mathematics*, **329**, 307–321. ISSN 0377-0427, doi: 10.1016/j.cam.2017.04.036, The International Conference on Information and Computational Science, 2–6 August 2016, Dalian, China, `http://www.sciencedirect.com/science/article/pii/S0377042717302078`.

Examples

```
fscore(ToothGrowth, 'supp', c('len'))
```

ga *Genetic Algorithm*

Description

The ga method (Yang and Honavar 1998) starts with an initial population of solutions and at each step applies a series of operators to the individuals in order to obtain new and better population of individuals. These operators are selection, crossing and mutation methods. This method uses the GA package implementation (Scrucca 2013) (Scrucca 2017).

Usage

```
ga(
  data,
  class,
  featureSetEval,
  popSize = 20,
  pcrossover = 0.8,
  pmutation = 0.1,
  maxiter = 100,
  run = 100,
  verbose = FALSE
)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureSetEval`	• The measure for evaluate features
`popSize`	• The popuplation size
`pcrossover`	• The probability of crossover between individuals
`pmutation`	• The probability of mutation between individuals
`maxiter`	• The number of iterations
`run`	• Number of consecutive iterations without fitness improvement to stop the algorithm
`verbose`	• Print the partial results in each iteration. This functionality is not available if the objective of the evaluation method is to minimize the target value (e.g. regression methods)

Value

A list is returned containing for each repetition of the algorithm:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

population Matrix with the population of the last iteration of the algorithm along with the evaluation measure of each individual

Author(s)

Francisco Aragón Royón

References

Scrucca L (2013). "GA: A Package for Genetic Algorithms in R." *Journal of Statistical Software*, **53**(4), 1–37. http://www.jstatsoft.org/v53/i04/.

Scrucca L (2017). "On some extensions to GA package: hybrid optimisation, parallelisation and islands evolution." *The R Journal*, **9**(1), 187–206. https://journal.r-project.org/archive/2017/RJ-2017-008.

Yang J, Honavar V (1998). "Feature subset selection using a genetic algorithm." In *Feature extraction, construction and selection*, 117–136. Springer.

Examples

```
## Genetic algorithm for iris dataset (filter method)
ga(iris, 'Species', roughsetConsistency, popSize = 10, maxiter=5, verbose=TRUE)
```

`gainRatio` *The gain ratio measure*

Description

This measure calculates the gain ratio value (Quinlan 1986), using the information theory.

Usage

```
gainRatio(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples. Feature columns are expected to be factors, as all features should be discrete.
`class`	• The name of the dependent variable
`features`	• The names of the selected features

Value

- The gain ratio value for the selected features.

Author(s)

Adan M. Rodriguez

References

Quinlan JR (1986). "Induction of decision trees." *Machine Learning*, **1**(1), 81–106. ISSN 1573-0565, doi: 10.1007/BF00116251, `https://doi.org/10.1007/BF00116251`.

Examples

```
gainRatio(iris,'Species',c('Sepal.Width', 'Sepal.Length'))
```

---

get.data.frame.from.formula *get.data.frame.from.formula*

Description

get.data.frame.from.formula

Usage

```
get.data.frame.from.formula(formula, data)
```

Arguments

`formula`	• formula
`data`	• data

Value

- data.frame

---

giniIndex *Gini index measure*

Description

This measure calculates the gini index (Ceriani and Verme 2012) of discrete features

Usage

```
giniIndex(data, class, features)
```

Arguments

- data
  - A data frame with the features and the class of the examples
- class
  - The name of the dependent variable
- features
  - The names of the selected feature

Value

- The Gini index value for the selected features

Author(s)

Adan M. Rodriguez

References

Ceriani L, Verme P (2012). "The origins of the Gini index: extracts from Variabilità e Mutabilità (1912) by Corrado Gini." *The Journal of Economic Inequality*, **10**(3), 421–443. ISSN 1573-8701, doi: 10.1007/s108880119188x, https://doi.org/10.1007/s10888-011-9188-x.

Examples

```
giniIndex(iris,'Species',c('Sepal.Width', 'Sepal.Length'))
```

---

hc *Hill-Climbing*

Description

The hc (Russell and Norvig 2009) method starts with a certain set of features and in each iteration it searches among its neighbors to advance towards a better solution. The method ends as soon as no better solutions are found

Usage

```
hc(
  data,
  class,
  featureSetEval,
  start = NULL,
  nneigh = length(data) - 1,
  repeats = 1,
  verbose = FALSE
)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureSetEval`	• The measure for evaluate features
`start`	• Binary vector with the set of initial features
`nneigh`	• Number of neighbors to evaluate in each iteration of the algorithm. By default: all posibles. It is important to note that a high value of this parameter considerably increases the computation time.
`repeats`	• Number of repetitions of the algorithm
`verbose`	• Print the partial results in each iteration

Value

A list is returned containing for each repetition of the algorithm:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

initialVector The vector with which the algorithm started

initialFitness The evaluation measure of the initial vector

trace Matrix with the results of each iteration. It contains the number of the iteration, the best set of features selected by the algorithm up to that iteration (1: selected, 0: not selected) and the value of the evaluation measure obtained from that best set of features

Author(s)

Francisco Aragón Royón

References

Russell S, Norvig P (2009). *Artificial Intelligence: A Modern Approach*, 3rd edition. Prentice Hall Press, Upper Saddle River, NJ, USA. ISBN 0136042597, 9780136042594.

Examples

```
## Hill-Climbing method for iris dataset (filter method)
hc(iris, 'Species', roughsetConsistency)
```

`IEConsistency` *Inconsistent Examples consistency measure*

Description

Calculates the inconsistent examples consistency value (Dash and Liu 2003), using hash tables

Usage

```
IEConsistency(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples. Feature columns are expected to be factors, as all features should be discrete.
`class`	• The name of the dependent variable
`features`	• The names of the selected features

Value

- The consistency value for the selected features

Author(s)

Adan M. Rodriguez

References

Dash M, Liu H (2003). "Consistency-based Search in Feature Selection." *Artif. Intell.*, **151**(1-2), 155–176. ISSN 0004-3702, doi: 10.1016/S00043702(03)000791, `http://dx.doi.org/10.1016/S0004-3702(03)00079-1`.

Examples

```
IEConsistency(iris,'Species',c('Sepal.Width', 'Sepal.Length'))
```

`IEPConsistency` *Inconsistent Examples Pairs consistency measure*

Description

Calculates the inconsistent examples pairs consistency value, using hash tables (Arauzo-Azofra et al. 2007)

Usage

```
IEPConsistency(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples. Feature columns are expected to be factors, as all features should be discrete.
`class`	• The name of the dependent variable
`features`	• The names of the selected features

Value

- The consistency value for the selected features

Author(s)

Adan M. Rodriguez

References

Arauzo-Azofra A, Benitez JM, Castro JL (2007). “Consistency measures for feature selection.” *Journal of Intelligent Information Systems*, **30**(3), 273–292. ISSN 1573-7675, doi: 10.1007/s10844-00700370, `http://dx.doi.org/10.1007/s10844-007-0037-0`.

Examples

```
IEPConsistency(iris,'Species',c('Sepal.Width', 'Sepal.Length'))
```

`Jd` *Jd evaluation measure*

Description

Applies the discriminant function designed by Narendra and Fukunaga (Narendra and Fukunaga 1977) to evaluate a set of features.

Usage

```
Jd(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`features`	• The names of the selected features

Value

- The value of the function for the selected features

Author(s)

Alfonso Jiménez-Vílchez

References

Narendra P, Fukunaga K (1977). "A Branch and Bound Algorithm for Feature Subset Selection." *IEEE Transactions on Computers*, **26**(9), 917–922. ISSN 0018-9340, doi: 10.1109/TC.1977.1674939.

Examples

```
Jd(ToothGrowth,'supp',c('len','dose'))
```

`LCC` *Linear Consistency-Constrained algorithm*

Description

Linear Consistency-Constrained algorithm described in (Shin and Xu 2009).

Usage

```
LCC(
  data,
  class,
  featureSetEval,
  featureEval = symmetricalUncertain,
  threshold = 0.9
)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureSetEval`	• The measure for evaluate feature sets
`featureEval`	• The measure for evaluate individual features
`threshold`	• Threshold

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

Author(s)

Alfonso Jiménez-Vílchez

References

Shin K, Xu XM (2009). "Consistency-Based Feature Selection." In Velásquez JD, Ríos SA, Howlett RJ, Jain LC (eds.), *Knowledge-Based and Intelligent Information and Engineering Systems*, 342–350. ISBN 978-3-642-04595-0.

Examples

```
## sfbs method for iris dataset (filter method)
LCC(iris, 'Species', IEConsistency)
```

lvw *Las Vegas Wrapper*

Description

The lvw method (Liu and Setiono 1996) starts with a certain set of features and in each step a new set is randomly generated, if the new set is better it is saved as the best solution. The algorithm ends when there are no improvements in a certain number of iterations.

Usage

```
lvw(
  data,
  class,
  featureSetEval,
  start = sample(0:1, ncol(data) - 1, replace = TRUE),
  K = 50,
  verbose = FALSE
)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureSetEval`	• The measure for evaluate features
`start`	• Binary vector with the set of initial features (1: selected and 0: unselected) for the algorithm
`K`	• The maximum number of iterations without improvement to finalize the algorithm
`verbose`	• Print the partial results in each iteration

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

initialVector The vector with which the algorithm started

initialFitness The evaluation measure of the initial vector

trace Matrix with the results of each iteration. It contains the number of the iteration, the value of k, the best set of features selected by the algorithm up to that iteration (1: selected, 0: not selected) and the value of the evaluation measure obtained from that best set of features

Author(s)

Francisco Aragón Royón

References

Liu H, Setiono R (1996). "Feature Selection And Classification - A Probabilistic Wrapper Approach." In *in Proceedings of the 9th International Conference on Industrial and Engineering Applications of AI and ES*, 419–424.

Examples

```
## lvw method for iris dataset (filter method)
lvw(iris, 'Species', roughsetConsistency, K=15, verbose=TRUE)
```

MDLC — *MDLC evaluation measure*

Description

Applies the Minimum-Description_Length-Criterion (MDLC) (Sheinvald et al. 1990) to evaluate a set of features.

Usage

```
MDLC(data, class, features)
```

Arguments

data	• A data frame with the features and the class of the examples
class	• The name of the dependent variable
features	• The names of the selected features

Value

- MDLC value for the selected features

Author(s)

Alfonso Jiménez-Vílchez

References

Sheinvald J, Dom B, Niblack W (1990). "A modeling approach to feature selection." In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume i, 535–539 vol. doi: 10.1109/ICPR.1990.118160.

Examples

```
MDLC(iris,'Species',c('Sepal.Width', 'Sepal.Length'))
```

`mutualInformation` *The mutual information measure*

Description

This measure calculates the mutual information value, using the information theory (Qian and Shu 2015).

Usage

```
mutualInformation(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples. Feature columns are expected to be factors, as all features should be discrete.
`class`	• The name of the dependent variable
`features`	• The names of the selected features

Value

- The mutual information value for the selected features

Author(s)

Adan M. Rodriguez

References

Qian W, Shu W (2015). “Mutual information criterion for feature selection from incomplete data.” *Neurocomputing*, **168**, 210–220. ISSN 18728286, doi: 10.1016/j.neucom.2015.05.105, `http://dx.doi.org/10.1016/j.neucom.2015.05.105`.

Examples

```
mutualInformation(iris,'Species',c('Sepal.Width', 'Sepal.Length'))
```

normalization — *Normalize a data frame*

Description

Takes in any data frame and normalize the data of their features

Usage

```
normalization(data, class)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The dependent variable

Details

Normalize the data (without the class)

Value

- The dataframe with the independent variables or features normalized

Author(s)

Adan M. Rodriguez

Examples

```
normalization(iris,'Species')
```

normalize.min.max — *normalize.min.max*

Description

normalize.min.max

Usage

```
normalize.min.max(data)
```

Arguments

`data`	• data

Value

- normalized data

---

relief *Relief*

Description

The relief algorithm (Kira and Rendell 1992) finds weights of continous and discrete attributes basing on a distance between instances. Adapted from Piotr Romanski's Fselector package (Romanski and Kotthoff 2018).

Usage

```
relief(data, class, features, neighbours.count = 5, sample.size = 10)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`features`	• The feature or features to evalute individually
`neighbours.count`	• number of neighbours to find for every sampled instance
`sample.size`	• number of instances to sample

Details

relief classification and regression continous and discrete data

Value

- a data.frame containing the worth of attributes in the first column and their names as row names

Author(s)

Alfonso Jiménez-Vílchez

References

Kira K, Rendell LA (1992). "A practical approach to feature selection." In *Machine Learning Proceedings 1992*, 249–256. Elsevier.

Romanski P, Kotthoff L (2018). *FSelector: Selecting Attributes*. R package version 0.31, https://CRAN.R-project.org/package=FSelector.

Examples

```
relief(iris, 'Species', c('Sepal.Width', 'Sepal.Length'))
```

RFSM *RFSM evaluation measure*

Description

Feature set measure based on relief. Described in (Arauzo-Azofra et al. 2004)

Usage

```
RFSM(data, class, features, m = 5, k = 4)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`features`	• The names of the selected features
`m`	• Number of iterations
`k`	• Number of neighbours

Value

- The value of the function for the selected features

Author(s)

Alfonso Jiménez-Vílchez

References

Arauzo-Azofra A, Ben\'itez J, Castro J (2004). "A feature set measure based on Relief." *Proceedings of the 5th International Conference on Recent Advances in Soft Computing*.

Examples

```
RFSM(iris, 'Species', c('Sepal.Width','Sepal.Length'))
```

roughsetConsistency *Rough Set consistency measure*

Description

Calculates the rough sets consistency value (Pawlak 1982) (Pawlak 1991), using hash tables

Usage

```
roughsetConsistency(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples. Feature columns are expected to be factors, as all features should be discrete.
`class`	• The name of the dependent variable
`features`	• The names of the selected features

Value

- The consistency value for the selected features

Author(s)

Adan M. Rodriguez

References

Pawlak Z (1982). "Rough sets." *International Journal of Computer \& Information Sciences*, **11**, 341–356. ISSN 0091-7036, doi: 10.1007/BF01001956, http://www.sciencedirect.com/science/article/pii/S0377042717302078.

Pawlak Z (1991). *Rough sets: Theoretical aspects of reasoning about data*, volume 9(1). Springer, Dordrecht. doi: 10.1007/9789401135344, http://dx.doi.org/10.1007/978-94-011-3534-4.

Examples

```
roughsetConsistency(iris,'Species',c('Sepal.Width', 'Sepal.Length'))
```

sa *Simulated Annealing*

Description

The sa method (Kirkpatrick et al. 1983) starts with a certain set of features and in each iteration modifies an element of the previous feature vector and decreases the temperature. If the energy of the new feature vector is better than that of the old vector, it is accepted and moved towards it, otherwise it is moved towards the new vector according to an acceptance probability. The algorithm ends when the minimum temperature has been reached. Additionally, a number of internal iterations can be performed within each iteration of the algorithm. In this case, the same temperature value of the outer iteration is used for the inner iterations

Usage

```
sa(
  data,
  class,
  featureSetEval,
  start = sample(0:1, ncol(data) - 1, replace = TRUE),
  temperature = 1,
  temperature_min = 0.01,
  reduction = 0.6,
  innerIter = 1,
  verbose = FALSE
)
```

Arguments

`data`
- A data frame with the features and the class of the examples

`class`
- The name of the dependent variable

`featureSetEval`
- The measure for evaluate features

`start`
- Binary vector with the set of initial features

`temperature`
- Temperature initial

`temperature_min`
- Temperature to stops in the outer loop

`reduction`
- Temperature reduction in the outer loop

`innerIter`
- Number of iterations of inner loop. By default no inner iterations are established

`verbose`
- Print the partial results in each iteration

Value

A list is returned containing for each repetition of the algorithm:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

initialVector The vector with which the algorithm started

initialEnergy The evaluation measure of the initial vector

traceOutter Matrix with the results of each iteration. Contains the number of the iteration, the value of the temperature, the subset of features of the iteration, its evaluation measure and whether there has been a movement from the previous iteration to obtain the subset of features in the current iteration.

traceInner List containing as many lists as outer iterations have been performed. In each iteration of these lists the same values are shown as for traceOutter but referring to each internal iteration.

Author(s)

Francisco Aragón Royón

References

Kirkpatrick S, Gelatt CD, Vecchi MP (1983). "Optimization by simulated annealing." *SCIENCE*, **220**(4598), 671–680. doi: 10.1126/science.220.4598.671 , `http://dx.doi.org/10.1126/science.220.4598.671`.

Examples

```
## Simulated Annealing for iris dataset (filter method)
sa(iris, 'Species', roughsetConsistency, temperature = 5, temperature_min=0.01,
   reduction=0.6, verbose=TRUE)
```

---

sbs *Sequential Backward Selection*

Description

The SBS method (Marill and Green 1963) starts with all the features and removes a single feature at each step with a view to improving the evaluation of the set.

Usage

```
sbs(data, class, featureSetEval, stopCriterion = -1, stop = FALSE)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureSetEval`	• The measure for evaluate features
`stopCriterion`	• Define a maximum number of iterations. Disabled if the value is -1 (default: -1)
`stop`	• If true, the function stops if next iteration does not improve current results (default: FALSE)

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

Author(s)

Adan M. Rodriguez

Alfonso Jiménez-Vílchez

Francisco Aragón Royón

References

Marill T, Green D (1963). "On the effectiveness of receptors in recognition systems." *Information Theory, IEEE Transactions on*, **9**(1), 11–17. doi: 10.1109/TIT.1963.1057810, `http://dx.doi.org/10.1109/TIT.1963.1057810`.

Examples

```
## sbs method for iris dataset (filter method)
sbs(iris, 'Species', giniIndex)
```

`selectDifference` *Select difference*

Description

Selects features (in descending order from the best evaluation measure to the lowest) until evaluation difference is over a threshold.

Usage

```
selectDifference(data, class, featureEval, d.threshold = 0.1)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureEval`	• The measure used to evaluate features
`d.threshold`	• Number between 0 and 1, to calculate the slope

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

featuresSelected The names of the returned features sorted according to the result of the evaluation measure

valuePerFeature The evaluation measures of the returned features

Author(s)

Adan M. Rodriguez

Francisco Aragón Royón

Examples

```
## Select Difference for iris dataset (filter method)
# Selects features in descending order as long as the difference between them is less than 0.1
selectDifference(iris, 'Species', chiSquared, 0.1)
```

---

selectKBest *Select K best*

Description

Takes the 'k' features with the greatest evaluations

Usage

```
selectKBest(data, class, featureEval, k = 1)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureEval`	• The measure used to evaluate features
`k`	• Number (positive integer) of returned features

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

featuresSelected The names of the k returned features sorted according to the result of the evaluation measure

valuePerFeature The evaluation measures of the k returned features

Author(s)

Adan M. Rodriguez

Francisco Aragón Royón

Examples

```
## Select K best for iris dataset (filter method)
selectKBest(iris, 'Species', roughsetConsistency, 2) # 2 best features
```

---

selectPercentile *Select Percentile*

Description

Selects a fraction, given as a percentage, of the total number of available features

Usage

```
selectPercentile(data, class, featureEval, percentile = 10)
```

Arguments

`data` • A data frame with the features and the class of the examples

`class` • The name of the dependent variable

`featureEval` • The measure used to evaluate features

`percentile` • Number (positive integer) between 0 and 100

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

featuresSelected The names of the returned features sorted according to the result of the evaluation measure

valuePerFeature The evaluation measures of the returned features

Author(s)

Adan M. Rodriguez

Francisco Aragón Royón

Examples

```
## Select Percentile for iris dataset (filter method)
selectPercentile(iris, 'Species', giniIndex, 80) # 80% best features
```

`selectSlope` *Select slope*

Description

Selects features (in descending order from the best evaluation measure to the lowest) until the slope to the next feature is over a threshold. The slope is calculated as: (s.threshold) / (number of features)

Usage

```
selectSlope(data, class, featureEval, s.threshold = 0.8)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureEval`	• The measure used to evaluate features
`s.threshold`	• Number between 0 and 1

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

featuresSelected The names of the returned features sorted according to the result of the evaluation measure

valuePerFeature The evaluation measures of the returned features

Author(s)

Adan M. Rodriguez

Examples

```
## Select Slope for iris dataset (filter method)
selectSlope(iris, 'Species', IEPConsistency, 0.8)
```

selectThreshold *Select threshold*

Description

Selects the features whose evaluation is over/under a user given threshold (It depends on the method that generates the evaluation measure. For example: under for regression methods, over for classification methods, etc.). Features that do not satisfy the threshold, will be removed

Usage

```
selectThreshold(data, class, featureEval, threshold = 0.5)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureEval`	• The measure used to evaluate features
`threshold`	• Number between 0 and 1

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

featuresSelected The names of the returned features sorted according to the result of the evaluation measure

valuePerFeature The evaluation measures of the returned features

Author(s)

Adan M. Rodriguez

Francisco Aragón Royón

Examples

```
## Select Threshold for iris dataset (filter method)
# Features with a evaluation measure higher than 0.7
selectThreshold(iris, 'Species', mutualInformation, 0.7)
```

selectThresholdRange *Select threshold range*

Description

Selects the features whose evaluation is over a threshold, where this threshold is given as: (((min - max) * p.threshold) + max)

Usage

```
selectThresholdRange(data, class, featureEval, p.threshold = 0.3)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureEval`	• The measure used to evaluate features
`p.threshold`	• Number between 0 and 1

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

featuresSelected The names of the returned features sorted according to the result of the evaluation measure

valuePerFeature The evaluation measures of the returned features

Author(s)

Adan M. Rodriguez

Francisco Aragón Royón

Examples

```
## Select Threshold range for iris dataset (filter method)
selectThresholdRange(iris, 'Species', determinationCoefficient, 0.3)
```

sfbs *Sequential Floating Backward Selection*

Description

The sfbs method (Pudil et al. 1994) starts with all the features and removes a single feature at each step with a view to improving the evaluation of the set. In addition, it checks whether adding any of the removed features, improve the value of the set.

Usage

```
sfbs(data, class, featureSetEval)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureSetEval`	• The measure for evaluate features

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

Author(s)

Adan M. Rodriguez

Francisco Aragón Royón

References

Pudil P, Novovičová J, Kittler J (1994). "Floating search methods in feature selection." *Pattern recognition letters*, **15**(11), 1119–1125.

Examples

```
## sfbs method for iris dataset (filter method)
sfbs(iris, 'Species', determinationCoefficient)
```

sffs *Sequential Floating Forward Selection*

Description

The sffs method (Pudil et al. 1994) starts with an empty set of features and add a single feature at each step with a view to improving the evaluation of the set. In addition, it checks whether removing any of the included features, improve the value of the set.

Usage

```
sffs(data, class, featureSetEval)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureSetEval`	• The measure for evaluate features

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

Author(s)

Adan M. Rodriguez

Francisco Aragón Royón

References

Pudil P, Novovičová J, Kittler J (1994). "Floating search methods in feature selection." *Pattern recognition letters*, **15**(11), 1119–1125.

Examples

```
## sffs method for mtcars dataset (filter method)
sffs(mtcars, 'mpg', mutualInformation)
```

sfs *Sequential Forward Selection*

Description

The SFS method (Whitney 1971) starts with an empty set of features and add a single feature at each step with a view to improving the evaluation of the set.

Usage

```
sfs(data, class, featureSetEval, stopCriterion = -1, stop = FALSE)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureSetEval`	• The measure for evaluate features
`stopCriterion`	• Define a maximum number of iterations. Disabled if the value is -1 (default: -1)
`stop`	• If true, the function stops if next iteration does not improve current results (default: FALSE)

Value

A list is returned containing:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

Author(s)

Adan M. Rodriguez

Alfonso Jiménez-Vílchez

Francisco Aragón Royón

References

Whitney AW (1971). "A Direct Method of Nonparametric Measurement Selection." *IEEE Trans. Comput.*, **20**(9), 1100–1103. ISSN 0018-9340, doi: 10.1109/TC.1971.223410, `http://dx.doi.org/10.1109/T-C.1971.223410`.

Examples

```
## sfs method for iris dataset (filter method)
sfs(iris, 'Species', roughsetConsistency)
```

symmetricalUncertain *Symmetrical uncertain measure*

Description

This measure calculates the symmetrical uncertain value (Witten and Frank 2005), using the information theory.

Usage

```
symmetricalUncertain(data, class, features)
```

Arguments

`data`	• A data frame with the features and the class of the examples. Feature columns are expected to be factors, as all features should be discrete.
`class`	• The name of the dependent variable
`features`	• The names of the selected features

Value

- The symmetrical uncertain value for the selected features

Author(s)

Adan M. Rodriguez

References

Witten IH, Frank E (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition. Morgan Kaufmann, San Francisco.

Examples

```
symmetricalUncertain(iris,'Species',c('Sepal.Width', 'Sepal.Length'))
```

ts *Tabu Search*

Description

The Tabu Search(Glover 1986) (Glover 1989) method starts with a certain set of features and in each iteration it searches among its neighbors to advance towards a better solution. The method has a memory (tabu list) that prevents returning to recently visited neighbors. The method ends when a certain number of iterations are performed, or when a certain number of iterations are performed without improvement, or when there are no possible neighbors. Once the method is finished, an intensification phase can be carried out that begins in the space of the best solutions found, or a diversification phase can be carried out in which solutions not previously visited are explored.

Usage

```
ts(
  data,
  class,
  featureSetEval,
  start = NULL,
  numNeigh = (ncol(data) - 1),
  tamTabuList = 5,
  iter = 100,
  iterNoImprovement = NULL,
  intensification = NULL,
  iterIntensification = 50,
  interPercentaje = 75,
  tamIntermediateMemory = 5,
  diversification = NULL,
  iterDiversification = 50,
  forgetTabuList = TRUE,
  verbose = FALSE
)
```

Arguments

data	• A data frame with the features and the class of the examples
class	• The name of the dependent variable
featureSetEval	• The measure for evaluate features
start	• Binary vector with the set of initial features
numNeigh	• The number of neighbor to consider in each iteration. By default: all posibles. It is important to note that a high value of this parameter considerably increases the computation time.
tamTabuList	• The size of the tabu list. By default: 5
iter	• The number of iterations of the algorithm. By default: 100

`iterNoImprovement`
- Number of iterations without improvement to start/reset the intensification/diversification phase. By default, it is not taken into account (all iterations are performed)

`intensification`
- Number of times the intensification phase is applied. None by default

`iterIntensification`
- Number of iterations of the intensification phase

`interPercentaje`
- Percentage of the most significant features to be taken into account in the intensification phase

`tamIntermediateMemory`
- Number of best solutions saved in the intermediate memory

`diversification`
- Number of times the diversification phase is applied. None by default

`iterDiversification`
- Number of iterations of the diversification phase

`forgetTabuList`
- Forget tabu list for intensification/diversification phases. By default: TRUE

`verbose`
- Print the partial results in each iteration

Value

A list is returned containing for each repetition of the algorithm:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

basicStage List containing the best neighbour in each iteration along with its obtained evaluation measure, and the content of the taboo list in each iteration

intensificationStage List containing for each repetition of the intensification phase the best neighbour in each iteration along with its obtained evaluation measure, and the content of the taboo list in each iteration.

diversificationStage List containing for each repetition of the diversification phase the best neighbour in each iteration along with its obtained evaluation measure, and the content of the taboo list in each iteration.

Author(s)

Francisco Aragón Royón

References

Glover F (1986). "Future Paths for Integer Programming and Links to Artificial Intelligence." *Comput. Oper. Res.*, **13**(5), 533–549. ISSN 0305-0548, doi: 10.1016/03050548(86)900481, `http://dx.doi.org/10.1016/0305-0548(86)90048-1`.

Glover F (1989). "Tabu Search—Part I." *ORSA Journal on Computing*, **1**(3), 190-206. doi: 10.1287/ijoc.1.3.190, https://doi.org/10.1287/ijoc.1.3.190, `https://doi.org/10.1287/ijoc.1.3.190`.

Examples

```
## Taboo-Search algorithm for iris dataset (filter method)
ts(iris, 'Species', roughsetConsistency, iter = 5)
```

---

woa *Whale Optimization Algorithm (Binary Whale Optimization Algorithm)*

Description

Binary Whale Optimization Algorithm (Kumar and Kumar 2018) is an algorithm that simulates the social behavior of humpback whales. This algorithm employs a binary version of the bubble-net hunting strategy. The algorithm starts with an initial population of individuals, and in each iteration updates the individuals according to several possible actions: Encircling prey, Bubble-net attacking or Search for prey

Usage

```
woa(data, class, featureSetEval, population = 10, iter = 10, verbose = FALSE)
```

Arguments

`data`	• A data frame with the features and the class of the examples
`class`	• The name of the dependent variable
`featureSetEval`	• The measure for evaluate features
`population`	• The number of whales population
`iter`	• The number of iterations of the algorithm
`verbose`	• Print the partial results in each iteration

Value

A list is returned containing for each repetition of the algorithm:

bestFeatures A vector with all features. Selected features are marked with 1, unselected features are marked with 0

bestFitness Evaluation measure obtained with the feature selection

popIter List that contains as many elements as iterations has the algorithm. Each of the elements in the list are matrices that represent the population in that iteration. In this matrix the individuals and the evaluation measure of each one are shown

Author(s)

Francisco Aragón Royón

References

Kumar V, Kumar D (2018). "Binary whale optimization algorithm and its application to unit commitment problem." *Neural Computing and Applications*. ISSN 1433-3058, doi: 10.1007/s00521-01837963, https://doi.org/10.1007/s00521-018-3796-3.

Examples

```
## Whale Optimization Algorithm for iris dataset (filter method)
woa(iris, 'Species', roughsetConsistency, population = 10, iter = 5, verbose = TRUE)
```

---

wrapperGenerator *Wrapper measure generator*

Description

Generates a wrapper function to be used as an evaluator (Kohavi and John 1997), given a learner algorithm and related customizable parameters (from Jed Wing et al. 2018). More specifically, the result of calling this function is another function that is used as an evaluator in the search methods, although you can also call it up to generate an evaluation measure individually.

Usage

```
wrapperGenerator(learner, resamplingParams, fittingParams)
```

Arguments

`learner`
- Learner to be used. The models available are the models available in caret: http://topepo.github.io/caret/available-models.html

`resamplingParams`
- Control parameters for evaluating the impact of model tuning parameters. The arguments are the same as those of the caret trainControl function

`fittingParams`
- Control parameters for choose the best model across the parameters. The arguments are the same as those of the caret train function (minus the parameters: x, y, form, data, method and trainControl).

Details

generaWrapper

Value

Returns a wrapper function that is used to generate an evaluation measure

Author(s)

Alfonso Jiménez-Vílchez

Francisco Aragón Royón

References

Kohavi R, John GH (1997). "Wrappers for feature subset selection." *Artificial intelligence*, **97**(1-2), 273–324.

from Jed Wing MKC, Weston S, Williams A, Keefer C, Engelhardt A, Cooper T, Mayer Z, Kenkel B, the R Core Team, Benesty M, Lescarbeau R, Ziem A, Scrucca L, Tang Y, Candan C, Hunt. T (2018). *caret: Classification and Regression Training*. R package version 6.0-80, https://CRAN.R-project.org/package=caret.

Examples

```
# Values for the caret trainControl function
resamplingParams <- list(method = "repeatedcv", repeats = 3)
# Values for the caret train function
fittingParams <- list(preProc = c("center", "scale"), metric="Accuracy",
                      tuneGrid = expand.grid(k = c(1:20)))
# Generation of the wrapper function
wrapper <- wrapperGenerator("knn", resamplingParams, fittingParams)
# The function call generates the evaluation measure
wrapper(iris, 'Species', c('Sepal.Length', 'Sepal.Width', 'Petal.Length'))
```

Index