Package ‘praznik’

February 29, 2020

Type Package

Title Tools for Information-Based Feature Selection

Version 8.0.0

Depends R (>= 2.10)

License GPL-3

Description A toolbox of fast, native and parallel implementations of various information-based importance criteria estimators and feature selection filters based on them, inspired by the overview by Brown, Pocock, Zhao and Lujan (2012) <http://www.jmlr.org/papers/v13/brown12a.html>. Contains, among other, minimum redundancy maximal relevancy ('mRMR') method by Peng, Long and Ding (2005) <doi:10.1109/TPAMI.2005.159>; joint mutual information ('JMI') method by Yang and Moody (1999) <http://papers.nips.cc/paper/1779-data-visualization-and-feature-selection-new-algorithms-for-nongaussian-data>; double input symmetrical relevance ('DISR') method by Meyer and Bontempi (2006) <doi:10.1007/11732242_9> as well as joint mutual information maximisation ('JMIM') method by Bennasar, Hicks and Setchi (2015) <doi:10.1016/j.eswa.2015.07.007>.

BugReports https://gitlab.com/mbq/praznik/issues

URL https://gitlab.com/mbq/praznik

Suggests testthat

Encoding UTF-8

RoxygenNote 7.0.2

NeedsCompilation yes

Author Miron B. Kursa [aut, cre] (<https://orcid.org/0000-0001-7672-648X>)

Maintainer Miron B. Kursa <m@mbq.me>

Repository CRAN

Date/Publication 2020-02-29 14:50:02 UTC

R topics documented:

`praznik-package` *Collection of Information-Based Feature Selection Filters*

Description

Praznik is a collection of feature selection filters performing greedy optimisation of mutual information-based usefulness criteria.

Details

In a nutshell, an algorithm of this class requires an information system (X, Y) and a predefined number of features to selected k, and works like this. To start, it estimates mutual information between each feature and the decision, find a feature with maximal such score and stores it as a first on a list of selected features, S. Then, it estimates a value of a certain criterion function $J(X, Y, S)$ for each feature X; this function also depends on Y and the set of already selected features S. As in the first step, the previously unselected feature with a greatest value of the criterion function is selected next. This is repeated until the method would gather k features, or, in case of some methods, when no more informative features can be found. The methods implemented in praznik consider the following criteria.

The mutual information maximisation filter, `MIM`, simply selects top-k features of best mutual information, that is

$$J_{MIM} = I(X;Y).$$

The minimal conditional mutual information maximisation proposed by F. Fleauret, `CMIM`, uses

$$J_{CMIM}(X) = \min(I(X;Y), \min_{W \in S} I(X;Y|W));$$

this method is also effectively identical to the information fragments method.

The minimum redundancy maximal relevancy proposed by H. Peng et al., `MRMR`, uses

$$J_{MRMR} = I(X;Y) - \frac{1}{|S|} \sum_{W \in S} I(X;W).$$

The joint mutual information filter by H. Yang and J. Moody, `JMI`, uses

$$J_{JMI} = \sum_{W \in S} I(X,W;Y).$$

The double input symmetrical relevance filter by P. Meyer and G. Bontempi, `DISR`, uses

$$J_{DISR}(X) = \sum_{W \in S} \frac{I(X,W;Y)}{H(X,W,Y)}.$$

The minimal joint mutual information maximisation filter by M. Bennasar, Y. Hicks and R. Setchi, `JMIM`, uses

$$J_{JMIM} = \min_{W \in S} I(X,W;Y).$$

The minimal normalised joint mutual information maximisation filter by the same authors, `NJMIM`, uses

$$J_{NJMIM} = \min_{W \in S} \frac{I(X,W;Y)}{H(X,W,Y)}.$$

While `CMIM`, `JMIM` and `NJMIM` consider minimal value over already selected features, they may use a somewhat more sophisticated and faster algorithm.

The package also provides functions for scoring features.

`miScores` returns

$$I(X;Y).$$

`cmiScores` returns, for a given condition vector Z,

$$I(X;Y|Z).$$

`jmiScores` returns

$$I(X,Z;Y).$$

`njmiScores` returns

$$\frac{I(X,Z;Y)}{H(X,Y,Z)}.$$

These function have also their `*Matrix` counterparts, which efficiently build a matrix of scores between all pairs of features. This is especially useful for network inference tasks.

Estimation of mutual information and its generalisations is a hard task; still, praznik aims at speed and simplicity and hence only offers basic, maximum likelihood estimator applicable on discrete data. For convenience, praznik automatically and silently coerces non-factor inputs into about ten equally-spaced bins, following the heuristic often used in literature.

Additionally, praznik has a limited, experimental support for replacing entropic statistics with Gini impurity-based; in such framework, entropy is replaced by Gini impurity

$$g(X) := 1 - \sum_x p_x^2,$$

which leads to an impurity gain

$$G(X;Y) := g(Y) - E(g(Y)|X) = \sum_{xy} \frac{p_{xy}^2}{p_x} - \sum_y p_y^2,$$

a counterpart of mutual information or information gain. It does not possess most of elegant properties of mutual information, yet values of both are usually highly correlated; moreover, Gini gain is computationally easier to calculate, hence it often replaces MI in performance-sensitive applications, like optimising splits in decision trees.

In a present version, praznik includes `impScores` for generating values of G for all features (an analog of `miScores`, as well as `JIM`, a Gini gain-based feature selection method otherwise identical to `JMI`.

Author(s)

Maintainer: Miron B. Kursa <`m@mbq.me`> (ORCID)

References

"Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection" G. Brown et al. JMLR (2012).

See Also

Useful links:

- `https://gitlab.com/mbq/praznik`
- Report bugs at `https://gitlab.com/mbq/praznik/issues`

`CMI` *Conditional mutual information maximisation filter*

Description

The method starts with a feature of a maximal mutual information with the decision Y. Then, it greedily adds feature X with a maximal value of the following criterion:

$$J(X) = I(X;Y|S),$$

where S is the set of already selected features.

Usage

```
CMI(X, Y, k = 3, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`k`	Number of attributes to select. Must not exceed `ncol(X)`.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A list with two elements: `selection`, a vector of indices of the selected features in the selection order, and `score`, a vector of corresponding feature scores. Names of both vectors will correspond to the names of features in `X`. Both vectors will be at most of a length `k`, as the selection may stop sooner, even during initial selection, in which case both vectors will be empty.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
data(MadelonD)
CMI(MadelonD$X,MadelonD$Y,20)
```

`CMIM` *Minimal conditional mutual information maximisation filter*

Description

The method starts with a feature of a maximal mutual information with the decision Y. Then, it greedily adds feature X with a maximal value of the following criterion:

$$J(X) = \min(I(X;Y), \min_{W \in S} I(X;Y|W)),$$

where S is the set of already selected features.

Usage

```
CMIM(X, Y, k = 3, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`k`	Number of attributes to select. Must not exceed `ncol(X)`.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A list with two elements: `selection`, a vector of indices of the selected features in the selection order, and `score`, a vector of corresponding feature scores. Names of both vectors will correspond to the names of features in `X`. Both vectors will be at most of a length `k`, as the selection may stop sooner, even during initial selection, in which case both vectors will be empty.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than

2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

References

"Fast Binary Feature Selection using Conditional Mutual Information Maximisation" F. Fleuret, JMLR (2004)

"Object recognition with informative features and linear classification" M. Vidal-Naquet and S. Ullman, IEEE Conference on Computer Vision and Pattern Recognition (2003).

Examples

```
data(MadelonD)
CMIM(MadelonD$X,MadelonD$Y,20)
```

`cmiMatrix` *Conditional mutual information matrix*

Description

Calculates conditional mutual information between each two features given another one, that is

$$I(X_i;X_j|Z).$$

Usage

```
cmiMatrix(X, Z, zeroDiag = TRUE, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Z`	Condition; should be given as a factor, but other options are accepted, as for features.
`zeroDiag`	Boolean flag, whether the diagonal should be filled with zeroes or with feature entropies.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical matrix with scores, with row and column names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
cmiMatrix(iris[,-5],iris[,5])
```

`cmiScores` *Conditional mutual information scores*

Description

Calculates conditional mutual information between each features and the decision, that is

$$I(X;Y|Z).$$

Usage

```
cmiScores(X, Y, Z, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`Z`	Condition; should be given as a factor, but other options are accepted, as for features.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical vector with conditional mutual information scores, with names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
cmiScores(iris[,-5],iris$Species,iris$Sepal.Length)
```

DISR *Double input symmetrical relevance filter*

Description

The method starts with a feature of a maximal mutual information with the decision Y. Then, it greedily adds feature X with a maximal value of the following criterion:

$$J(X) = \sum_{W\in S} \frac{I(X,W;Y)}{H(X,W,Y)},$$

where S is the set of already selected features.

Usage

```
DISR(X, Y, k = 3, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`k`	Number of attributes to select. Must not exceed `ncol(X)`.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A list with two elements: `selection`, a vector of indices of the selected features in the selection order, and `score`, a vector of corresponding feature scores. Names of both vectors will correspond to the names of features in `X`. Both vectors will be at most of a length `k`, as the selection may stop sooner, even during initial selection, in which case both vectors will be empty.

Note

DISR is a normalised version of `JMI`; `JMIM` and `NJMIM` are modifications of JMI and DISR in which minimal joint information over already selected features is used instead of a sum.

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

References

"On the Use of Variable Complementarity for Feature Selection in Cancer Classification" P. Meyer and G. Bontempi, (2006)

Examples

```
data(MadelonD)
DISR(MadelonD$X,MadelonD$Y,20)
```

`dnmiMatrix` *Directional normalised mutual information matrix*

Description

Calculates directed normalised mutual information between each two features, that is

$$\frac{I(X_i, X_j)}{H(X_j)}.$$

Usage

```
dnmiMatrix(X, zeroDiag = TRUE, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`zeroDiag`	Boolean flag, whether the diagonal should be filled with zeroes or with feature entropies.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical matrix with scores, with row and column names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
dnmiMatrix(iris)
```

`hScores` *Entropy scores*

Description

Calculates entropy of each feature, that is

$$H(X).$$

Usage

```
hScores(X, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical vector with entropy scores, with names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
hScores(iris[,-5])
```

`impScores` *Gini impurity scores*

Description

Calculates Gini impurity between each feature and the decision, that is

$$G(X;Y)=\sum_{xy}\frac{p_{xy}^2}{p_x}-\sum_{y}p_y^2.$$

Usage

```
impScores(X, Y, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical vector with Gini impurity scores, with names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
impScores(iris[,-5],iris$Species)
```

`jhScores` *Joint entropy scores*

Description

Calculates joint entropy of each feature and a condition `Y`, that is

$$H(X,Y).$$

Usage

```
jhScores(X, Y, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical vector with entropy scores, with names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
jhScores(iris[,-5],iris[,5])
```

`JIM` *Joint impurity filter*

Description

The method starts with a feature of a maximal impurity gain with the decision Y. Then, it greedily adds feature X with a maximal value of the following criterion:

$$J(X) = \sum_{W \in S} G(X, W; Y),$$

where S is the set of already selected features, and

$$G(X;Y) = \sum_{xy} \frac{p_{xy}^2}{p_x} - \sum_y p_y^2$$

is the Gini impurity gain from partitioning Y according to X.

Usage

```
JIM(X, Y, k = 3, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`k`	Number of attributes to select. Must not exceed `ncol(X)`.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A list with two elements: `selection`, a vector of indices of the selected features in the selection order, and `score`, a vector of corresponding feature scores. Names of both vectors will correspond to the names of features in `X`. Both vectors will be at most of a length `k`, as the selection may stop sooner, even during initial selection, in which case both vectors will be empty.

Note

This is an impurity-based version of `JMI`; expect similar results in slightly shorter time.

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
data(MadelonD)
JIM(MadelonD$X,MadelonD$Y,20)
```

JMI *Joint mutual information filter*

Description

The method starts with a feature of a maximal mutual information with the decision Y. Then, it greedily adds feature X with a maximal value of the following criterion:

$$J(X) = \sum_{W \in S} I(X, W; Y),$$

where S is the set of already selected features.

Usage

```
JMI(X, Y, k = 3, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`k`	Number of attributes to select. Must not exceed `ncol(X)`.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A list with two elements: `selection`, a vector of indices of the selected features in the selection order, and `score`, a vector of corresponding feature scores. Names of both vectors will correspond to the names of features in `X`. Both vectors will be at most of a length `k`, as the selection may stop sooner, even during initial selection, in which case both vectors will be empty.

Note

`DISR` is a normalised version of JMI; `JMIM` and `NJMIM` are modifications of JMI and DISR in which minimal joint information over already selected features is used instead of a sum.

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

References

"Data Visualization and Feature Selection: New Algorithms for Nongaussian Data H. Yang and J. Moody, NIPS (1999)

Examples

```
data(MadelonD)
JMI(MadelonD$X,MadelonD$Y,20)
```

JMIM *Minimal joint mutual information maximisation filter*

Description

The method starts with a feature of a maximal mutual information with the decision Y. Then, it greedily adds feature X with a maximal value of the following criterion:

$$J(X) = \min_{W \in S} I(X, W; Y),$$

where S is the set of already selected features.

Usage

```
JMIM(X, Y, k = 3, threads = 0)
```

Arguments

X	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
Y	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
k	Number of attributes to select. Must not exceed `ncol(X)`.
threads	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A list with two elements: `selection`, a vector of indices of the selected features in the selection order, and `score`, a vector of corresponding feature scores. Names of both vectors will correspond to the names of features in `X`. Both vectors will be at most of a length `k`, as the selection may stop sooner, even during initial selection, in which case both vectors will be empty.

Note

NJMIM is a normalised version of JMIM; JMI and DISR are modifications of JMIM and NJMIM in which a sum of joint information over already selected features is used instead of a minimum.

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

References

"Feature selection using Joint Mutual Information Maximisation" M. Bennasar, Y. Hicks and R. Setchi, (2015)

Examples

```
data(MadelonD)
JMIM(MadelonD$X,MadelonD$Y,20)
```

---

jmiMatrix *Joint mutual information matrix*

Description

Calculates mutual information between each feature and a joint mix of each other feature with a given feature, that is

$$I(X_i; X_j, Z).$$

Usage

```
jmiMatrix(X, Z, zeroDiag = TRUE, threads = 0)
```

Arguments

X	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. NAs are not allowed.
Z	Condition; should be given as a factor, but other options are accepted, as for features.

`zeroDiag`	Boolean flag, whether the diagonal should be filled with zeroes or with feature entropies.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical matrix with scores, with row and column names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
jmiMatrix(iris[,-5],iris[,5])
```

`jmiScores` *Joint mutual information scores*

Description

Calculates joint mutual information between each feature joint with some other vector `Z` with the decision, that is

$$I(X,Z;Y).$$

This is the same as conditional mutual information between X and Y plus a constant that depends on Y and Z, that is

$$I(X,Z;Y)=I(X;Y|Z)+I(Y;Z).$$

Usage

```
jmiScores(X, Y, Z, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`Z`	Other vector; should be given as a factor, but other options are accepted, as for features.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical vector with joint mutual information scores, with names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
jmiScores(iris[,-5],iris$Species,iris$Sepal.Length)
```

`joinf` *Join factors*

Description

Convenience function for joining factors.

Usage

```
joinf(...)
```

Arguments

`...`	One or more features to merge. Given as single vectors or data.frames. Accepted feature types are factor (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation). `NA`s are not allowed.

Value

Joint factor, with levels `l1` to `l<n>`. Vacant combinations are dropped.

Note

You can pass a single vector to this function to see how praznik interprets it.

Examples

```
joinf(c(1,2,1,2),c(1,1,2,2))
```

`kInverse` *Inverse Kendall transform*

Description

This function attempts to reverse Kendall transformation using a simple ranking agreement method, which always restores original ranking if the input corresponds to one, or a reasonable best-effort guess if not. Namely, each objects gets a score based on its relation with each other object, 2 points for a win (`'>'`) and 1 point for a tie (`'='`); these scores are used to calculate ranks. This function can also be directly given greater-than scores, for instance confidence scores from some classifier trained on Kendall-transformed data.

Usage

```
kInverse(x)
```

Arguments

`x`	A Kendall-transformed feature to be converted back into a ranking. To be interpreted as a such, it must be a factor with levels being a subset of `'<'`, `'>'` or `'='`. Alternatively, it may be a numeric vector of greater-than scores.

Value

Vector of ranks corresponding to `x`.

Note

An order of elements in `x` is crucial; if it is not the same as generated by the `kTransform`, results will be wrong. This function cannot assert that the order is correct.

Examples

```
kInverse(kTransform(1:7))
```

`kTransform` *Kendall transformation*

Description

Kendall transformation

Usage

```
kTransform(x)
```

Arguments

`x`	Vector or data frame to be Kendall-transformed; allowed feature types are numeric, integer (treated as numeric), ordered factor, logical and unordered factor with two or less levels. `NA` and non-finite values are allowed; `NaN` is treated as `NA`.

Value

A transformed vector or data frame with transformed columns.

Examples

```
kTransform(data.frame(Asc=1:3,Desc=3:1,Vsh=c(2,1,2)))
```

`MadelonD` *Pre-discretised Madelon dataset*

Description

Madelon is a synthetic data set from the NIPS 2003 feature selection challenge, generated by Isabelle Guyon. It contains 480 irrelevant and 20 relevant features, including 5 informative and 15 redundant. In this version, the originally numerical features have been pre-cut into 10 bins, as well as their names have been altered to reveal 20 relevant features (as identified by the Boruta method).

Usage

```
data(MadelonD)
```

Format

A list with two elements, `X` containing a data frame with predictors, and `Y`, the decision. Features are in the same order as in the original data; the names of relevant ones start with `Rel`, while of irrelevant ones with `Irr`.

Source

`https://archive.ics.uci.edu/ml/datasets/Madelon`

`MIM` *Mutual information maximisation filter*

Description

Calculates mutual information between all features and the decision, then returns top k.

Usage

```
MIM(X, Y, k = 3, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`k`	Number of attributes to select. Must not exceed `ncol(X)`.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A list with two elements: `selection`, a vector of indices of the selected features in the selection order, and `score`, a vector of corresponding feature scores. Names of both vectors will correspond to the names of features in `X`. Both vectors will be at most of a length `k`, as the selection may stop sooner, even during initial selection, in which case both vectors will be empty.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than

2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
data(MadelonD)
MIM(MadelonD$X,MadelonD$Y,20)
```

---

miMatrix *Mutual information matrix*

Description

Calculates mutual information between each two features, that is

$$I(X_i, X_j).$$

Usage

```
miMatrix(X, zeroDiag = TRUE, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`zeroDiag`	Boolean flag, whether the diagonal should be filled with zeroes or with feature entropies.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical matrix with scores, with row and column names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
miMatrix(iris)
```

`miScores` *Mutual information scores*

Description

Calculates mutual information between each feature and the decision, that is

$$I(X,Y).$$

Usage

```
miScores(X, Y, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical vector with mutual information scores, with names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
miScores(iris[,-5],iris$Species)
```

`MRMR` *Minimum redundancy maximal relevancy filter*

Description

The method starts with a feature of a maximal mutual information with the decision Y. Then, it greedily adds feature X with a maximal value of the following criterion:

$$J(X) = I(X;Y) - \frac{1}{|S|}\sum_{W\in S} I(X;W),$$

where S is the set of already selected features.

Usage

```
MRMR(X, Y, k = if (positive) ncol(X) else 3, positive = FALSE, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`k`	Number of attributes to select. Must not exceed `ncol(X)`.
`positive`	If true, algorithm won't return features with negative scores (i.e., with redundancy term higher than the relevance therm). In that case, `k` controls the maximal number of returned features, and is set to 'ncol(X)' by default.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A list with two elements: `selection`, a vector of indices of the selected features in the selection order, and `score`, a vector of corresponding feature scores. Names of both vectors will correspond to the names of features in `X`. Both vectors will be at most of a length `k`, as the selection may stop sooner, even during initial selection, in which case both vectors will be empty.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

References

"Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy" H. Peng et al. IEEE Pattern Analysis and Machine Intelligence (PAMI) (2005)

Examples

```
data(MadelonD)
MRMR(MadelonD$X,MadelonD$Y,20)
```

`NJMIM` *Minimal normalised joint mutual information maximisation filter*

Description

The method starts with a feature of a maximal mutual information with the decision Y. Then, it greedily adds feature X with a maximal value of the following criterion:

$$J(X) = \min_{W \in S} \frac{I(X,W;Y)}{H(X,W,Y)},$$

where S is the set of already selected features.

Usage

```
NJMIM(X, Y, k = 3, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.

`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`k`	Number of attributes to select. Must not exceed `ncol(X)`.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A list with two elements: `selection`, a vector of indices of the selected features in the selection order, and `score`, a vector of corresponding feature scores. Names of both vectors will correspond to the names of features in `X`. Both vectors will be at most of a length `k`, as the selection may stop sooner, even during initial selection, in which case both vectors will be empty.

Note

NJMIM is a normalised version of `JMIM`; `JMI` and `DISR` are modifications of JMIM and NJMIM in which a sum of joint information over already selected features is used instead of a minimum. It stops returning features when the best score reaches 0.

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

References

"Feature selection using Joint Mutual Information Maximisation" M. Bennasar, Y. Hicks and R. Setchi, (2015)

Examples

```
data(MadelonD)
NJMIM(MadelonD$X,MadelonD$Y,20)
```

`njmiMatrix` *Normalised joint mutual information matrix*

Description

Calculates normalised mutual information between each feature and a joint mix of each other feature with a given feature, that is

$$\frac{I(X_i;X_j,Z)}{H(X_i,X_j,Z)}.$$

Usage

```
njmiMatrix(X, Z, zeroDiag = TRUE, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Z`	Condition; should be given as a factor, but other options are accepted, as for features.
`zeroDiag`	Boolean flag, whether the diagonal should be filled with zeroes or with feature entropies.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical matrix with scores, with row and column names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
njmiMatrix(iris[,-5],iris[,5])
```

`njmiScores` *Normalised joint mutual information scores*

Description

Calculated normalised mutual information between each feature joint with some other vector `Z` and the decision, that is

$$\frac{I(X,Z;Y)}{H(X,Y,Z)}.$$

This is the same as in the criterion used by `DISR` and `NJMIM`.

Usage

```
njmiScores(X, Y, Z, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`Y`	Decision attribute; should be given as a factor, but other options are accepted, exactly like for attributes. `NA`s are not allowed.
`Z`	Other vector; should be given as a factor, but other options are accepted, as for features.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical vector with the normalised joint mutual information scores, with names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
njmiScores(iris[,-5],iris$Species,iris$Sepal.Length)
```

`nmiMatrix` *Normalised mutual information matrix*

Description

Calculates normalised mutual information between each two features, that is

$$\frac{I(X_i,X_j)}{H(X_i,X_j)}.$$

Usage

```
nmiMatrix(X, zeroDiag = TRUE, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`zeroDiag`	Boolean flag, whether the diagonal should be filled with zeroes or with feature entropies.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A numerical matrix with scores, with row and column names copied from `X`.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

Examples

```
nmiMatrix(iris)
```

`triScores` *Mutual information of feature triples*

Description

Calculates mutual information of each triple of features, that is

$$I(X_i; X_j; X_k).$$

Usage

```
triScores(X, threads = 0)
```

Arguments

`X`	Attribute table, given as a data frame with either factors (preferred), booleans, integers (treated as categorical) or reals (which undergo automatic categorisation; see below for details). Single vector will be interpreted as a data.frame with one column. `NA`s are not allowed.
`threads`	Number of threads to use; default value, 0, means all available to OpenMP.

Value

A data frame with four columns; first three (`Var1`, `Var2` and `Var3`) are names of features, fourth, `MI` is the value of the mutual information. The order of features does not matter, hence only

$$n(n-1)(n-2)/6$$

unique, sorted triples are evaluated.

Note

The method requires input to be discrete to use empirical estimators of distribution, and, consequently, information gain or entropy. To allow smoother user experience, praznik automatically coerces non-factor vectors in inputs, which requires additional time, memory and may yield confusing results – the best practice is to convert data to factors prior to feeding them in this function. Real attributes are cut into about 10 equally-spaced bins, following the heuristic often used in literature. Precise number of cuts depends on the number of objects; namely, it is $n/3$, but never less than 2 and never more than 10. Integers (which technically are also numeric) are treated as categorical variables (for compatibility with similar software), so in a very different way – one should be aware that an actually numeric attribute which happens to be an integer could be coerced into a n-level categorical, which would have a perfect mutual information score and would likely become a very disruptive false positive.

In a current version, the maximal number of features accepted is 2345, which gives a bit less than 2^32 triples. The equation used for calculation is

$$I(X_i;X_j;X_k)=I(X_i;X_k)+I(X_j;X_k)-I(X_i,X_j;X_k).$$

Henceforth, please mind that rounding errors may occur and influence reproducibility.

Examples

```
triScores(iris)
```

Index