# Package ‘FeatureSelection’

September 11, 2016

**Type** Package

**Title** A package for feature selection

**Version** 0.1.0

**Date** 2016-09-01

**Author** Adan M. Rodriguez <i32muroa@gmail.com>

**Maintainer** Adan M. Rodriguez <i32muroa@gmail.com>

**Description** This package provides a lot of feature selection algorithms, given a dataset. These algorithms, are formed by a search method and a measurement evaluation.

**Repository** CRAN

**License** GPL-2

**LazyData** false

**Imports** rpart, neuralnet, class, digest, RUnit

**Depends** rpart, neuralnet, class, digest, RUnit

**RoxygenNote** 5.0.1

**NeedsCompilation** no

## `Consistency based measures` *Consistency based measures, for discrete features only*

### Description

These measures calculates the binary consistency, rough sets consistency, inconsistent examples consistency or inconsistent examples pairs consistency value, using hash tables

### Usage

```
binaryConsistency(data, class, features)
  roughsetConsistency(data, class, features)
  IEPConsistency(data, class, features)
  IEConsistency(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 consistency value for the selected features

### Author(s)

Adan M. Rodriguez

### Examples

```
  ## Not run: data <- data(Zoo)
    binaryConsistency(data, 'type', c('catsize', 'domestic'))
    IEPConsistency(data, 'type', 'tail')
    IEConsistency(data, 'type', 'catsize')
    roughsetConsistency(data, 'type', c('tail', 'domestic', 'catsize'))

## End(Not run)
```

## `Cutting criteria` *Feature Selection using cutting criteria*

### Description

These algorithms, take those features that exceed a certain or fulfill certain curting criteria.

### Usage

```
selectKBest(data, class, featureSetEval, k)
selectPercentile(data, class, featureSetEval, percentile)
selectThreshold(data, class, featureSetEval, threshold)
selectThresholdRange(data, class, featureSetEval, p.threshold)
selectDifference(data, class, featureSetEval, d.threshold)
selectSlope(data, class, featureSetEval, s.threshold)
```

### 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 |
| `k` | Number (positive integer) of returned features |
| `percentile` | Number (positive integer) between 0 and 100 |
| `threshold` | Number between 0 and 1 |
| `p.threshold` | Number between 0 and 1 |
| `d.threshold` | Number between 0 and 1 |
| `s.threshold` | Number between 0 and 1, to calculate the slope |

### Details

`selectKBest`: The selected features will be the 'k' ones with greater evaluation

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

`selecThreshold`: Selects the features whose evaluation is over a user given threshold

`selecThresholdRange`: Selects the features whose evaluation is over a threshold, where this threshold is given as a fraction of the range of evaluation function

`selecDifference`: Selects features (in descending order) until evalu- ation difference is over a threshold.

`selecSlope`: Selects features (in descending order) until the slope to the next feature is over a threshold.

### Value

A character vector of selected features

### Author(s)

Adan M. Rodriguez

### Examples

```
  ## Not run: data <- data(Zoo)
    selectKbest(data, 'type', roughsetConsistency, 4)
    selectPercentile(data, 'type', giniIndex, 90)
    selectThreshold(data, 'type', mutualInformation, 0.5)
    selectThresholdRange(data, 'type', determinationCoefficient, 0.3)
    selectDifference(data, 'type', chiSquared, 0.1)
    selectSlope(data, 'type', IEPconsistency, 0.8)

## End(Not run)
```

---

## `Exhaustive Search` *Exhaustive search*

---

### Description

This exhaustive search, searches the whole features subset in breadth first order

### 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 character vector of selected features

### Author(s)

Adan M. Rodriguez

### Examples

```
  ## Not run: data <- data(HouseVotes84)
  breadthFirstSearch(data, 'Class', binaryConsistency)
## End(Not run)
```

## `Gini index measure` *Gini index measure, for discrete features only*

### Description

This measure calculates the gini index 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 features |

### Value

The Gini index value for the selected features

### Author(s)

Adan M. Rodriguez

### Examples

```
## Not run: data <- data(Zoo)
        giniIndex(data, 'type', c('tail', 'domestic'))
## End(Not run)
```

## `Information based measures` *Information based measures, for discrete features only*

### Description

These measures calculates the mutual information, gain ratio or symmetrical uncertain value, using the information theory.

### Usage

```
mutualInformation(data, class, features)
      gainRatio(data, class, features)
      symmetricalUncertain(data, class, features)
      entropy(x)
      entropyJ(x)
```

### 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 |
| `x` | The name of the feature or the class to calculate entropy |

### Value

The mutual information, gain ratio or symmetrical uncertain value for the selected features

### Author(s)

Adan M. Rodriguez

### Examples

```
  ## Not run: data <- data(HouseVotes84)
          mutualInformation(data, 'Class', c('V1', 'V2'))
          gainRatio(data, 'Class', 'V10')
          symmetricalUncertain(data, 'Class', 'V1')
## End(Not run)
```

---

## `Measures based on Chi squared test` *Chi squared and Cramer V measures*

---

### Description

These measures calculates the Chi squared or the Cramer V value, evaluating the selected features individually

### Usage

```
chiSquared(data, class, features)
  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

The chi squared or cramer V value for each selected features

### Author(s)

Adan M. Rodriguez

### Examples

```
  ## Not run: data <- data(Zoo)
    chiSquared(data, 'type', c('catsize', 'tail'))
    cramer(data, 'type', 'domestic')
## End(Not run)
```

---

## `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 |

### Value

The dataframe with the independent variables or features normalized

### Author(s)

Adan M. Rodriguez

### Examples

```
## Not run: data <- data(iris)
          normalization(data)
## End(Not run)
```

## `R squared measure` *R Squared, to continous features*

### Description

This measure calculates the determinantion coefficient 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

### Examples

```
data <- data(mtcars)
  ## Not run: determinationCoefficient(data, 'mpg', c('cyl', 'disp' , 'vs'))
```

## `Sequential Search` *Sequential search*

### Description

These algorithms implement sequential searchs for searching features in the subset space of features.

### Usage

```
sfs(data, class, featureSetEval)
sbs(data, class, featureSetEval)
sffs(data, class, featureSetEval)
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 |

### Details

`sfs(Sequential Forward Selection)`: The sfs method 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.

`sbs(Sequential Backward Selection)`: The sbs method starts with all the features and removes a single feature at each step with a view to improving the evaluation of the set.

`sffs(Sequential Floating Forward Selection)`: The sffs method 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.

`sfbs(Sequential Floating Backward Selection)`: The sfbs method 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.

### Value

A character vector of selected features

### Author(s)

Adan M. Rodriguez

### Examples

```
  ## Not run: data <- data(Zoo)
  sfs(data, 'type', roughsetConsistency)
  sbs(data, 'type', giniIndex)
  sffs(data, 'type', mutualInformation)
  sfbs(data, 'type', determinationCoefficient)

## End(Not run)
```

---

## `Wrapper measure` *Wrapper measure, for regression and classification problems*

---

### Description

This measure calculates CCR for classification problems or MSE for regression problems

**Usage**

```
wrapperParameters(k.fold, type=c("lm", "rpart", "neuralnet", "knn"))
```

**Arguments**

| | |
|---|---|
| `k.fold` | Number of folds for the cross-validation |
| `type` | The name of the selected learning algorithm |

**Value**

Object type wrapper for use in a search method

**Author(s)**

Adan M. Rodriguez

**Examples**

```
  ## Not run: data <- data(Zoo)
            wp <- wrapperParameters(3, 'rpart')
            sfs(data, 'type', wr)

## End(Not run)
```