Package ‘smartdata’

December 18, 2019

Title   Data Preprocessing
Version 1.0.3
Maintainer Ignacio Cordón <nacho.cordon.castillo@gmail.com>
Description Eases data preprocessing tasks, providing a data flow based on a pipe operator which eases cleansing, transformation, oversampling, or instance/feature selection operations.
Depends R (>= 3.5.0), mice (>= 2.46.0)
Imports functional, checkmate, magrittr, infotheo, MVN, adaptiveGPCA, discretization, outliers, NoiseFiltersR, Boruta, FSelector, lle, unbalanced, RoughSets, class, clusterSim, Amelia, imbalance, DMwR, missForest, missMDA, denoiseR, VIM
Suggests testthat, mlbench, rpart, knitr, rmarkdown
License GPL (>= 2) | file LICENSE
Encoding UTF-8
LazyData true

BugReports http://github.com/ncordon/smartdata/issues
URL http://github.com/ncordon/smartdata
RoxygenNote 7.0.2
VignetteBuilder knitr
NeedsCompilation no
Author Ignacio Cordón [aut, cre],
Francisco Charte [aut],
Julián Luengo [aut],
Salvador García [aut],
Francisco Herrera [aut]
Repository CRAN
Date/Publication 2019-12-18 17:40:02 UTC
R topics documented:

clean_noise ................................................. 2
clean_outliers ............................................. 3
discretize .................................................. 4
feature_selection .......................................... 5
impute_missing ............................................. 7
instance_selection ......................................... 8
normalize .................................................. 9
oversample .................................................. 10
smartdata .................................................. 11
space_transformation ...................................... 12
which_options ............................................. 13

Index 14

---

clean_noise  Noise cleaning wrapper

Description
Noise cleaning wrapper

Usage

clean_noise(dataset, method, class_attr = "Class", ...)

Arguments

dataset  we want to clean noisy instances on
method  selected method of noise cleaning
class_attr  character. Indicates the class attribute or attributes from dataset. Must exist in it.
...  Further arguments for method

Value
The treated dataset (either with noisy instances replaced or erased)

Examples

library("smartdata")
data(iris0, package = "imbalance")

super_iris <- clean_noise(iris, method = "AENN", class_attr = "Species", k = 3)
super_iris <- clean_noise(iris, "GE", class_attr = "Species", k = 5, relabel_th = 2)
super_iris <- clean_noise(iris, "HARF", class_attr = "Species",
num_folds = 10, agree_level = 0.7, num_trees = 5)
super_iris <- clean_noise(iris0, "TomekLinks")
super_iris <- clean_noise(iris, "hybrid", class_attr = "Species",
                        consensus = FALSE, action = "repair")
super_iris <- clean_noise(iris, "Mode", class_attr = "Species", type = "iterative",
                        action = "repair", epsilon = 0.05,
                        num_iterations = 200, alpha = 1, beta = 1)
super_iris <- clean_noise(iris, "INFFC", class_attr = "Species", consensus = FALSE,
                        prob_noisy = 0.2, num_iterations = 3, k = 5, threshold = 0)
super_iris <- clean_noise(iris, "IPF", class_attr = "Species", consensus = FALSE,
                        num_folds = 3, prob_noisy = 0.2,
                        prob_good = 0.5, num_iterations = 3)
super_iris <- clean_noise(iris, "ORBoost", class_attr = "Species",
                        num_boosting = 20, threshold = 11, num_adaboost = 20)
super_iris <- clean_noise(iris, "PF", class_attr = "Species", prob_noisy = 0.01,
                        num_iterations = 5, prob_good = 0.5, theta = 0.8)
super_iris <- clean_noise(iris, "C45robust", class_attr = "Species", num_folds = 5)

---

**clean_outliers**  
*Outliers cleaning wrapper*

**Description**
Outliers cleaning wrapper

**Usage**
clean_outliers(dataset, method, ...)

**Arguments**
- **dataset** we want to clean outliers of
- **method** selected method to clean outliers. Possibilities are:
  - "univariate" detects outliers column by column (an outlier will be an abnormal value inside a column) and fills them with mean or median of the corresponding column
  - "multivariate" detects outliers using a multicolumn approach, so that an outlier will be a whole observation (row). And deletes those observations
  ... further arguments for the method

**Value**
The treated dataset (either with outliers replaced or erased)
Examples

library("smartdata")

super_iris <- clean_outliers(iris, method = "multivariate", type = "adj")
super_iris <- clean_outliers(iris, method = "multivariate", type = "quan")

# Use mean as method to substitute outliers
super_iris <- clean_outliers(iris, method = "univariate", type = "z", prob = 0.9, fill = "mean")
# Use median as method to substitute outliers
super_iris <- clean_outliers(iris, method = "univariate", type = "z", prob = 0.9, fill = "median")
# Use chi-sq instead of z p-values
super_iris <- clean_outliers(iris, method = "univariate", type = "chisq",
                        prob = 0.9, fill = "median")
# Use interquartilic range instead (lim argument is mandatory when using it)
super_iris <- clean_outliers(iris, method = "univariate", type = "iqr",
                        prob = 0.9, fill = "median")

discretize

Discretization wrapper

Description

Discretization wrapper

Usage

discretize(dataset, method, class_attr = NULL, exclude = NULL, ...)

Arguments

dataset we want to perform discretization on
method selected method of discretization
class_attr character. Indicates the class attribute or attributes from dataset. Must exist in it.
exclude character. Vector of attributes to exclude from the discretization
...
  Further arguments for method

Value

The discretized dataset
**Examples**

```r
library("smartdata")

super_iris <- discretize(iris, method = "chi_merge", class_attr = "Species", exclude = "Sepal.Length")
super_iris <- discretize(iris, method = "chi_merge", class_attr = "Species", alpha = 0.7)
super_iris <- discretize(iris, method = "chi2", "Species", alpha = 0.7, delta = 0.1)
super_iris <- discretize(iris, method = "chi2", class_attr = "Species")
super_iris <- discretize(iris, method = "extended_chi2", class_attr = "Species")
super_iris <- discretize(iris, method = "ameva", class_attr = "Species")
super_iris <- discretize(iris, method = "CAIM", class_attr = "Species")
super_iris <- discretize(iris, method = "CACC", class_attr = "Species")
super_iris <- discretize(iris, method = "equalwidth", num_bins = nrow(iris) / 2)
super_iris <- discretize(iris, method = "equalfreq", num_bins = nrow(iris) / 2)
```

---

**feature_selection**  
**Feature selection wrapper**

**Description**

Feature selection wrapper

**Usage**

```r
feature_selection(dataset, method, class_attr = NULL, exclude = NULL, ...)
```

**Arguments**

- `dataset`  
  we want to do feature selection on

- `method`  
  selected method of feature selection

- `class_attr`  
  character. Indicates the class attribute or attributes from `dataset`. Must exist in it.

- `exclude`  
  character. Vector of attributes to exclude from the feature selection process

- `...`  
  Further arguments for `method`

**Value**

The treated dataset (either with noisy instances replaced or erased)
Examples

library("smartdata")
library("rpart")
data(ecoli1, package = "imbalance")
data(HouseVotes84, package = "mlbench")

# Extracted from FSelector::best.first.search documentation
evaluator <- function(subset) {
  k <- 5
  splits <- runif(nrow(iris))
  results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- iris[test.idx, , drop=FALSE]
    train <- iris[train.idx, , drop=FALSE]
    tree <- rpart(FSelector::as.simple.formula(subset, "Species"), train)
    error.rate = sum(test$Species != predict(tree, test, type="c")) / nrow(test)
    return(1 - error.rate)
  })
  print(subset)
  print(mean(results))
  return(mean(results))
}

super_iris <- feature_selection(iris, "Boruta", class_attr = "Species")
super_iris <- feature_selection(iris, "chi_squared", class_attr = "Species", num_features = 3)
# Pick 3 attributes from the continuous ones
super_ecoli <- feature_selection(ecoli1, "information_gain", class_attr = "Class", num_features = 3)
super_ecoli <- feature_selection(ecoli1, "gain_ratio", class_attr = "Class", num_features = 3)
super_ecoli <- feature_selection(ecoli1, "sym_uncertainty", class_attr = "Class", num_features = 3)
super_votes <- feature_selection(HouseVotes84, "oneR", exclude = c("V1", "V2"), class_attr = "Class", num_features = 3)
super_votes <- feature_selection(iris, "RF_importance", class_attr = "Species", num_features = 3, type = 2)

super_iris <- feature_selection(iris, "best_first_search", exclude = "Species", eval_fun = evaluator)
super_iris <- feature_selection(iris, "forward_search", exclude = "Species", eval_fun = evaluator)
super_iris <- feature_selection(iris, "backward_search", exclude = "Species", eval_fun = evaluator)
super_iris <- feature_selection(iris, "cfs", class_attr = "Species")
super_iris <- feature_selection(iris, "consistency", class_attr = "Species")
impute_missing  Missing values imputation wrapper

**Description**

Missing values imputation wrapper

**Usage**

`impute_missing(dataset, method, exclude = NULL, ...)`

**Arguments**

- `dataset` we want to impute missing values on
- `method` selected method of missing values imputation
- `exclude` character. Vector of attributes to exclude from the missing values treatment
- `...` Further arguments for method

**Value**

The treated dataset (either with noisy instances replaced or erased)

**Examples**

```r
library("smartdata")
data(africa, package = "Amelia")
data(nhanes, package = "mice")
data(ozone, package = "missMDA")
data(vnf, package = "missMDA")
data(orange, package = "missMDA")
data(sleep, package = "VIM")

super_nhanes <- impute_missing(nhanes, "gibbs_sampling")
super_nhanes <- impute_missing(nhanes, "gibbs_sampling", exclude = "chl")
# Use a different method for every column
impute_methods <- c("pmm", "midastouch", "norm_nob", "norm_boot")
super_nhanes <- impute_missing(nhanes, "gibbs_sampling", imputation = impute_methods)
super_nhanes <- impute_missing(nhanes, "central_imputation")
super_africa <- impute_missing(africa, "knn_imputation")
# Execute knn imputation with non default value for k
super_africa <- impute_missing(africa, "knn_imputation", k = 5)
super_africa <- impute_missing(africa, "expect_maximization", exclude = "country")
super_africa <- impute_missing(africa, "rf_imputation", num_iterations = 15, num_trees = 200, bootstrap = FALSE)
# Examples of calls to 'PCA imputation' with wholly numeric datasets
super_orange <- impute_missing(orange, "PCA_imputation", num_dimensions = 5, imputation = "EM")
```
super_orange <- impute_missing(orange, "PCA_imputation", num_dimensions = 5,
   imputation = "Regularized")

super_orange <- impute_missing(orange, "PCA_imputation", num_dimensions = 5,
   imputation = "Regularized", random_init = TRUE)

# Examples of calls to 'MCA imputation' with wholly categorical datasets

super_vnf <- impute_missing(vnf, "MCA_imputation", num_dimensions = 5,
   imputation = "EM")
super_vnf <- impute_missing(vnf, "MCA_imputation", num_dimensions = 5,
   imputation = "Regularized")
super_vnf <- impute_missing(vnf, "MCA_imputation", num_dimensions = 5,
   imputation = "Regularized", random_init = TRUE)

# Examples of calls to 'FAMD imputation' with hybrid datasets

super_ozone <- impute_missing(ozone, "FAMD_imputation", num_dimensions = 5,
   imputation = "EM", exclude = c("Ne12", "Vx15"))
super_ozone <- impute_missing(ozone, "FAMD_imputation", num_dimensions = 5,
   imputation = "Regularized")
super_ozone <- impute_missing(ozone, "FAMD_imputation", num_dimensions = 5,
   imputation = "Regularized", random_init = TRUE)

# Examples of hotdeck, iterative robust and regression imputations

super_sleep <- impute_missing(sleep, "hotdeck")
super_sleep <- impute_missing(sleep, "iterative_robust", initialization = "median",
   num_iterations = 1000)
super_sleep <- impute_missing(sleep, "regression_imputation",
   formula = Dream+NonD~BodyWgt+BrainWgt)

# Examples of adaptative shrinkage imputation

super_ozone <- impute_missing(ozone, "ATN", sigma = 2.2)
super_ozone <- impute_missing(ozone, "ATN", lambda = 0.025, gamma = 2.5)
super_ozone <- impute_missing(ozone, "ATN", tune = "SURE")

---

**instance_selection**  
**Instance selection wrapper**

**Description**

Instance selection wrapper

**Usage**

instance_selection(dataset, method, class_attr = "Class", ...)

**Arguments**

- **dataset** we want to perform an instance selection on
- **method** selected method of instance selection
- **class_attr** character. Indicates the class attribute from dataset. Must exist in it
- ... Further arguments for method

**Value**

A filtered dataset with same shape as the input to the function

**Examples**

```r
library("smartdata")

super_iris <- instance_selection(iris, method = "CNN", class_attr = "Species")
# Use k = 2 instead of default k
super_iris <- instance_selection(iris, method = "CNN", class_attr = "Species", k = 2)
# Use Edited Nearest Neighbor as method to select observations
super_iris <- instance_selection(iris, method = "ENN", class_attr = "Species", k = 3)
super_iris <- instance_selection(iris, method = "multiedit", class_attr = "Species",
                               k = 3, num_folds = 5, null_passes = 8)
# Use default arguments for multiedit
super_iris <- instance_selection(iris, method = "multiedit", class_attr = "Species")
super_iris <- instance_selection(iris, method = "FRIS", class_attr = "Species")
# FRIS method with fuzzy granularity of 2
super_iris <- instance_selection(iris, method = "FRIS", class_attr = "Species", alpha = 2)
# FRIS method with Dubois Prade implicator
super_iris <- instance_selection(iris, method = "FRIS", "Species", implicator_type = "dubois_prade")
# FRIS method with lower threshold (that is, less observations are removed)
super_iris <- instance_selection(iris, method = "FRIS", class_attr = "Species", threshold = 0.6)
```

---

**normalize**

**Normalization wrapper**

**Description**

Normalization wrapper

**Usage**

```r
normalize(dataset, method, exclude = NULL, ...)
```

**Arguments**

- **dataset** we want to perform normalization on
- **method** selected method of normalization
- **exclude** character. Vector of attributes to exclude from the normalization
- ... Further arguments for method
oversample

Value
The normalized dataset

Examples
library("smartdata")

    super_iris <- normalize(iris, method = "min_max", exclude = "Species", by = "column")
    # Use default parameter by = "row"
    super_iris <- normalize(iris, method = "min_max", exclude = c("Sepal.Length", "Species"))
    super_iris <- normalize(iris, method = "min_max", exclude = "Species", by = "row")
    super_iris <- normalize(iris, method = "z_score", exclude = "Species", by = "row")
    super_iris <- normalize(iris, method = "sd_quotient", exclude = "Species", by = "row")

oversample

Oversampling wrapper

Description
Oversampling wrapper

Usage
oversample(dataset, method, class_attr = "Class", ...)

Arguments
dataset we want to perform oversampling on
method selected method of oversampling
class_attr character. Indicates the class attribute or attributes from dataset. Must exist in it.
...
Further arguments for method

Value
An oversampled dataset

Examples
library("smartdata")
data(iris0, package = "imbalance")

    super_iris <- oversample(iris0, method = "MWMOTE", class_attr = "Class",
                              ratio = 0.8, filtering = TRUE)
    super_iris <- oversample(iris0, method = "SMOTE", class_attr = "Class", ratio = 0.6)
    super_iris <- oversample(iris0, method = "PDFOS", class_attr = "Class", ratio = 0.6)
    super_iris <- oversample(iris0, method = "RWO", class_attr = "Class", ratio = 0.8)
```r
super_iris <- oversample(iris0, method = "SLMOTE", class_attr = "Class")
```

### Description

Provides a pipe interface that integrates a collection of the most used data preprocessing libraries, providing oversampling, instance and feature selection, normalization, discretization, space transformation and outliers/noise/missing values treatment.

**Method to do oversampling** `oversample`

NA

**Method to do instance selection** `instance_selection`

NA

**Method to do feature selection** `feature_selection`

NA

**Method to do normalization** `normalize`

NA

**Method to do discretization** `discretize`

NA

**Method to do space transformation** `space_transformation`

NA

**Method to treat outliers** `clean_outliers`

NA

**Method to treat missing values** `impute_missing`

NA

**Method to treat noise** `clean_noise`

NA
space_transformation

Space transformation wrapper

Description

Space transformation wrapper

Usage

space_transformation(dataset, method, exclude = NULL, ...)

Arguments

dataset   we want to do space transformation on
method    selected method for space transformation
exclude   character. Vector of attributes to exclude from the space transformation process
...       Further arguments for method

Value

The transformed dataset

Examples

library("smartdata")
data(ecoli1, package = "imbalance")
data(AntibioticSmall, package = "adaptiveGPCA")
antibiotics <- data.frame(AntibioticSmall$X)

super_ecoli <- space_transformation(ecoli1, "lle_knn", k = 3, num_features = 2, regularization = 1, exclude = c("Mcg", "Alm1"))

super_ecoli <- space_transformation(ecoli1, "lle_epsilon", epsilon = 0.99, num_features = 3)

super_antibiotics <- space_transformation(antibiotics, "adaptative_gpca", similarity = AntibioticSmall$Q, num_features = 2)
**which_options**

*Prints options for a wrapper or a certain preprocessing method*

**Description**

Prints options for a wrapper or a certain preprocessing method

**Usage**

```r
which_options(preprocess, method = NULL)
```

**Arguments**

- **preprocess**: Possible preprocessing: 'oversample', 'clean_noise', 'instance_selection', 'feature_selection', 'normalize', 'discretize', 'space_transformation', 'clean_outliers', 'impute_missing', 'clean_noise'
- **method**: For the preprocessing method

**Value**

Prints valid preprocessings for the selected wrapper or options for a given preprocessing

**Examples**

```r
which_options("oversample")
which_options("clean_noise", method = "edgeWeight")
which_options("clean_noise", method = "ENG")
which_options("impute_missing", method = "gibbs_sampling")
```
Index

clean_noise, 2, I
clean_outliers, 3, I

discretize, 4, II

feature_selection, 5, II

impute_missing, 7, II
instance_selection, 8, II

normalize, 9, II

oversample, 10, II

smartdata, 11
space_transformation, II, 12

which_options, 13