Package ‘noisemodel’

October 17, 2022

Version 1.0.2
Title Noise Models for Classification Datasets
Description Implementation of models for the controlled introduction of errors in classification datasets. This package contains the noise models described in Saez (2022) <doi:10.3390/math10203736> that allow corrupting class labels, attributes and both simultaneously.
License GPL (>= 3)
Depends R (>= 3.5.0)
Imports caret, mnet, e1071, FNN, classInt, ggplot2, ExtDist, lsr, stringr, RColorBrewer, RSNNs, C50
Suggests knitr, rmarkdown, testthat (>= 3.0.0)
VignetteBuilder knitr
Encoding UTF-8
RoxygenNote 7.2.1
NeedsCompilation no
Author José A. Sáez [aut, cre]
Maintainer José A. Sáez <joseasaezm@ugr.es>
Repository CRAN
Date/Publication 2022-10-17 06:20:02 UTC
Config/testthat/edition 3

R topics documented:

asy_def_ln ................................................................. 3
asy_int_an ................................................................. 5
asy_spa_in ................................................................. 7
asy_uni_an ................................................................. 9
asy_uni_In ................................................................. 11
attn_uni_In ............................................................... 12
boud_gau_an .............................................................. 14
clu_vot_in ................................................................. 16
R topics documented:

diris2D ................................................................. 18
exp_cuni_ln .......................................................... 19
exp_bor_ln ............................................................. 20
fra_bdir_ln ........................................................... 22
gam_borLn ............................................................. 24
gaum_bor_ln ........................................................... 26
gau_bor_ln .............................................................. 28
glev_uni_ln ............................................................. 30
hubp_uni_ln ............................................................ 32
imp_int_an .............................................................. 33
iris2D ................................................................. 35
irs_bdir_ln ............................................................. 36
lap_bor_ln .............................................................. 38
larm_uni_ln ............................................................. 40
maj_udir_ln ............................................................. 42
mind_bdir_ln ........................................................... 44
minp_uni_ln ............................................................. 46
mis_pre_ln .............................................................. 47
mulc_udir_ln ........................................................... 49
nei_bor_ln .............................................................. 51
nlin_bor_ln ............................................................. 53
oned_uni_ln ............................................................. 54
opes_idnn_ln ........................................................... 57
opes_idu_ln ............................................................. 59
pai_bdir_ln ............................................................. 61
plot.ndmodel .......................................................... 63
pmd_con_ln ............................................................. 64
print.ndmodel ........................................................... 66
qua_uni_ln ............................................................. 67
sco_con_ln ............................................................. 69
sigb_uni_ln ............................................................. 71
smam_bor-ln ........................................................... 72
smu_cuni_ln ............................................................ 74
summary.ndmodel ....................................................... 76
symd_gau_an ........................................................... 77
symd_gimg_an ........................................................... 79
symd_rpix_an ........................................................... 81
symd Uni-an ............................................................ 83
sym_adj Ln .............................................................. 84
sym_cen Ln .............................................................. 86
sym_con Ln .............................................................. 88
sym_cuni_an ............................................................. 90
sym_cuni_cn ............................................................ 91
sym_cuni Ln ............................................................. 93
sym_ddef Ln ............................................................ 95
sym_def Ln .............................................................. 97
sym_dia Ln .............................................................. 99
sym_dran Ln ............................................................ 101
asy_def_ln

Description

Introduction of Asymmetric default label noise into a classification dataset.

Usage

```r
## Default S3 method:
asy_def_ln(x, y, level, def = 1, order = levels(y), sortid = TRUE, ...)

## S3 method for class 'formula'
asy_def Ln(formula, data, ...)
```

Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double vector with the noise levels in [0,1] to be introduced into each class.
- `def`: an integer with the index of the default class (default: 1).
- `order`: a character vector indicating the order of the classes (default: `levels(y)`).
sortid a logical indicating if the indices must be sorted at the output (default: TRUE).
...
formula a formula with the output class and, at least, one input attribute.
data a data frame in which to interpret the variables in the formula.

details

Asymmetric default label noise randomly selects \((\text{level}[i] \cdot 100)\)% of the samples of each class \(C[i]\) in the dataset - the order of the class labels is determined by order. Then, the labels of these samples are replaced by a fixed label \((C[\text{def}])\) within the set of class labels.

value

An object of class \textit{ndmodel} with elements:

- \textit{xnoise}: a data frame with the noisy input attributes.
- \textit{ynoise}: a factor vector with the noisy output class.
- \textit{numnoise}: an integer vector with the amount of noisy samples per class.
- \textit{idnoise}: an integer vector list with the indices of noisy samples.
- \textit{numclean}: an integer vector with the amount of clean samples per class.
- \textit{idclean}: an integer vector list with the indices of clean samples.
- \textit{distr}: an integer vector with the samples per class in the original data.
- \textit{model}: the full name of the noise introduction model used.
- \textit{param}: a list of the argument values.
- \textit{call}: the function call.

note

Noise model adapted from the papers in References.

references


see also

\texttt{sym_nean_ln}, \texttt{print.ndmodel}, \texttt{summary.ndmodel}, \texttt{plot.ndmodel}
Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- asy_def_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)],
                       level = c(0.1, 0.2, 0.3), order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- asy_def_ln(formula = Species ~ ., data = iris2D,
                      level = c(0.1, 0.2, 0.3), order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

asy_int_an

Asymmetric interval-based attribute noise

Description

Introduction of Asymmetric interval-based attribute noise into a classification dataset.

Usage

```r
## Default S3 method:
asy_int_an(x, y, level, nbins = 10, sortid = TRUE, ...)

## S3 method for class 'formula'
asy_int_an(formula, data, ...)
```

Arguments

- `x` a data frame of input attributes.
- `y` a factor vector with the output class of each sample.
- `level` a double vector with the noise levels in [0,1] to be introduced into each attribute.
- `nbins` an integer with the number of bins to create (default: 10).
- `sortid` a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...` other options to pass to the function.
- `formula` a formula with the output class and, at least, one input attribute.
- `data` a data frame in which to interpret the variables in the formula.
Details

Asymmetric interval-based attribute noise corrupts \((\text{level}_i \times 100)\)% of the values for each attribute \(A[i]\) in the dataset. In order to corrupt an attribute \(A[i]\), \((\text{level}_i \times 100)\)% of the samples in the dataset are chosen. To corrupt a value in numeric attributes, the attribute is split into equal-frequency intervals, one of its closest intervals is picked out and a random value within the interval is chosen as noisy. For nominal attributes, a random value within the domain is selected.

Value

An object of class \textit{ndmodel} with elements:

- \texttt{xnoise} a data frame with the noisy input attributes.
- \texttt{ynoise} a factor vector with the noisy output class.
- \texttt{numnoise} an integer vector with the amount of noisy samples per attribute.
- \texttt{idnoise} an integer vector list with the indices of noisy samples per attribute.
- \texttt{numclean} an integer vector with the amount of clean samples per attribute.
- \texttt{idclean} an integer vector list with the indices of clean samples per attribute.
- \texttt{distr} an integer vector with the samples per class in the original data.
- \texttt{model} the full name of the noise introduction model used.
- \texttt{param} a list of the argument values.
- \texttt{call} the function call.

Note

Noise model adapted from the papers in References.

References


See Also

\textit{asy_uni_an}, \textit{symd_gimg_an}, \textit{print.ndmodel}, \textit{summary.ndmodel}, \textit{plot.ndmodel}

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- asy_int_an(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)],
                     level = c(0.1, 0.2))

# show results
```
asy_spa_ln

summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- asy_int_an(formula = Species ~ ., data = iris2D,
            level = c(0.1, 0.2))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

asy_spa_ln  
Asymmetric sparse label noise

Description

Introduction of Asymmetric sparse label noise into a classification dataset.

Usage

## Default S3 method:
asy_spa_ln(x, y, levelO, levelE, order = levels(y), sortid = TRUE, ...)

## S3 method for class 'formula'
asy_spa_ln(formula, data, ...)

Arguments

x       a data frame of input attributes.
y       a factor vector with the output class of each sample.
levelO  a double with the noise level in [0,1] to be introduced into each odd class.
levelE  a double with the noise level in [0,1] to be introduced into each even class.
order   a character vector indicating the order of the classes (default: levels(y)).
sortid  a logical indicating if the indices must be sorted at the output (default: TRUE).
...     other options to pass to the function.
formula a formula with the output class and, at least, one input attribute.
data a data frame in which to interpret the variables in the formula.

details

Asymmetric sparse label noise randomly selects (levelO·100)% of the samples in each odd class and (levelE·100)% of the samples in each even class -the order of the class labels is determined by order. Then, each odd class is flipped to the next class, whereas each even class is flipped to the previous class. If the dataset has an odd number of classes, the last class is not corrupted.
Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

`mind_bdir_ln`, `fra_bdir_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- asy_spa_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)],
                     levelO = 0.1, levelE = 0.3, order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- asy_spa_ln(formula = Species ~ ., data = iris2D,
                      levelO = 0.1, levelE = 0.3, order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
```
**asy_uni_an**

### Description

Introduction of *Asymmetric uniform attribute noise* into a classification dataset.

### Usage

```r
## Default S3 method:
asy_uni_an(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
asy_uni_an(formula, data, ...)
```

### Arguments

**x**
- a data frame of input attributes.

**y**
- a factor vector with the output class of each sample.

**level**
- a double vector with the noise levels in \([0,1]\) to be introduced into each attribute.

**sortid**
- a logical indicating if the indices must be sorted at the output (default: `TRUE`).

**...**
- other options to pass to the function.

**formula**
- a formula with the output class and, at least, one input attribute.

**data**
- a data frame in which to interpret the variables in the formula.

### Details

*Asymmetric uniform attribute noise* corrupts \((\text{level}[i] \cdot 100)\)% of the values for each attribute \(A[i]\) in the dataset. In order to corrupt an attribute \(A[i]\), \((\text{level}[i] \cdot 100)\)% of the samples in the dataset are chosen. Then, their values for \(A[i]\) are replaced by random different ones between the minimum and maximum of the domain of the attribute following a uniform distribution (for numerical attributes) or choosing a random value (for nominal attributes).

### Value

An object of class `ndmodel` with elements:

**xnoise**
- a data frame with the noisy input attributes.

**ynoise**
- a factor vector with the noisy output class.

**numnoise**
- an integer vector with the amount of noisy samples per attribute.

**idnoise**
- an integer vector list with the indices of noisy samples per attribute.

**numclean**
- an integer vector with the amount of clean samples per attribute.
idclean an integer vector list with the indices of clean samples per attribute.
distr an integer vector with the samples per class in the original data.
model the full name of the noise introduction model used.
param a list of the argument values.
call the function call.

Note
Noise model adapted from the papers in References.

References

See Also
symd_gimg_an, unc_vgau_an, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- asy_uni_an(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)],
                     level = c(0.1, 0.2))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- asy_uni_an(formula = Species ~ ., data = iris2D,
                     level = c(0.1, 0.2))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
**asy_uni_ln**  

**Asymmetric uniform label noise**

**Description**

Introduction of *Asymmetric uniform label noise* into a classification dataset.

**Usage**

```r
## Default S3 method:
asy_uni_ln(x, y, level, order = levels(y), sortid = TRUE, ...)

## S3 method for class 'formula'
asy_uni_ln(formula, data, ...)
```

**Arguments**

- `x`  
a data frame of input attributes.
- `y`  
a factor vector with the output class of each sample.
- `level`  
a double vector with the noise levels in [0,1] to be introduced into each class.
- `order`  
a character vector indicating the order of the classes (default: `levels(y)`).
- `sortid`  
a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `formula`  
a formula with the output class and, at least, one input attribute.
- `data`  
a data frame in which to interpret the variables in the formula.

**Details**

*Asymmetric uniform label noise* randomly selects \( \text{level}[i] \times 100\)% of the samples of each class \( C[i] \) in the dataset -the order of the class labels is determined by `order`. Finally, the labels of these samples are randomly replaced by other different ones within the set of class labels.

**Value**

An object of class `ndmodel` with elements:

- `xnoise`  
a data frame with the noisy input attributes.
- `ynoise`  
a factor vector with the noisy output class.
- `numnoise`  
an integer vector with the amount of noisy samples per class.
- `idnoise`  
an integer vector list with the indices of noisy samples.
- `numclean`  
an integer vector with the amount of clean samples per class.
- `idclean`  
an integer vector list with the indices of clean samples.
- `distr`  
an integer vector with the samples per class in the original data.
- `model`  
the full name of the noise introduction model used.
- `param`  
a list of the argument values.
- `call`  
the function call.
Note

Noise model adapted from the papers in References.

References


See Also

maj_udir_ln, asy_def_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- asy_uni_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)],
  level = c(0.1, 0.2, 0.3), order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- asy_uni_ln(formula = Species ~ ., data = iris2D,
  level = c(0.1, 0.2, 0.3), order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

attm_uni_ln

Attribute-mean uniform label noise

Description

Introduction of Attribute-mean uniform label noise into a classification dataset.

Usage

## Default S3 method:
attm_uni_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
attm_uni_ln(formula, data, ...)
Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in [0,1] with the noise level to be introduced.
- **sortid**: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

Details

For each sample, its distance to the mean of each attribute is computed. Then, \((\text{level} \cdot 100)\)% of the samples in the dataset are randomly selected to be mislabeled, more likely choosing samples whose features are generally close to the mean. The labels of these samples are randomly replaced by other different ones within the set of class labels.

Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

Note

Noise model adapted from the papers in References

References


See Also

`qua_uni_ln, exps_cuni_ln, print.ndmodel, summary.ndmodel, plot.ndmodel`
Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- attm_uni_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- attm_uni_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**boud_gau_an**

*Boundary/dependent Gaussian attribute noise*

**Description**

Introduction of *Boundary/dependent Gaussian attribute noise* into a classification dataset.

**Usage**

```r
## Default S3 method:
boud_gau_an(x, y, level, k = 0.2, sortid = TRUE, ...)

## S3 method for class 'formula'
boud_gau_an(formula, data, ...)
```

**Arguments**

- `x` a data frame of input attributes.
- `y` a factor vector with the output class of each sample.
- `level` a double in [0,1] with the noise level to be introduced.
- `k` a double in [0,1] with the scale used for the standard deviation (default: 0.2).
- `sortid` a logical indicating if the indices must be sorted at the output (default: TRUE).
- `formula` a formula with the output class and, at least, one input attribute.
- `data` a data frame in which to interpret the variables in the formula.
Details

Boundary/dependent Gaussian attribute noise corrupts (level·100)% samples among the ((level+0.1)·100)% of samples closest to the decision boundary. Their attribute values are corrupted by adding a random number that follows a Gaussian distribution of mean = 0 and standard deviation = (max-min)·k, being max and min the limits of the attribute domain. For nominal attributes, a random value is chosen.

Value

An object of class ndmodel with elements:

- `xnoise` a data frame with the noisy input attributes.
- `ynoise` a factor vector with the noisy output class.
- `numnoise` an integer vector with the amount of noisy samples per attribute.
- `idnoise` an integer vector list with the indices of noisy samples per attribute.
- `numclean` an integer vector with the amount of clean samples per attribute.
- `idclean` an integer vector list with the indices of clean samples per attribute.
- `distr` an integer vector with the samples per class in the original data.
- `model` the full name of the noise introduction model used.
- `param` a list of the argument values.
- `call` the function call.

Note

Noise model adapted from the papers in References.

References


See Also

- `imp_int_an`, `asy_int_an`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- boud_gau_an(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)
```
# usage of the method for class formula
set.seed(9)
outfrm <- boud_gau_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

clu_vot_ln

## Usage

### Default S3 method:
clu_vot_ln(x, y, k = nlevels(y), sortid = TRUE, ...)

### S3 method for class 'formula'
clu_vot_ln(formula, data, ...)

## Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **k**: an integer with the number of clusters (default: nlevels(y)).
- **sortid**: a logical indicating if the indices must be sorted at the output (default: TRUE).
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

## Details

*Clustering-based voting label noise* divides the dataset into $k$ clusters. Then, the labels of each cluster are relabeled with the majority class among its samples.

## Value

An object of class ndmodel with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
clu_vot_ln

idnoise an integer vector list with the indices of noisy samples.
numclean an integer vector with the amount of clean samples per class.
idclean an integer vector list with the indices of clean samples.
distr an integer vector with the samples per class in the original data.
model the full name of the noise introduction model used.
param a list of the argument values.
call the function call.

Note

Noise model adapted from the papers in References, which considers \( k \)-means as unsupervised clustering method.

References


See Also

sco_con_ln, mis_pre_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- clu_vot_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)])

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- clu_vot_ln(formula = Species ~., data = iris2D)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
diris2D

**diris2D dataset**

**Description**

Discretized version of the iris2D dataset.

**Usage**

`data(diris2D)`

**Format**

A data.frame with 103 samples (rows) and 3 variables (columns) named Petal.Length, Petal.Width and Species.

**Source**

Data collected by E. Anderson (1935).

**References**


**See Also**

`iris2D, print.ndmodel, summary.ndmodel, plot.ndmodel`

**Examples**

```r
# load the dataset
data(diris2D)

# noise introduction
set.seed(9)
outdef <- sym_uni_ln(x = diris2D[,-ncol(diris2D)], y = diris2D[,ncol(diris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
```
Exponential/smudge completely-uniform label noise

Description

Introduction of Exponential/smudge completely-uniform label noise into a classification dataset.

Usage

```r
## Default S3 method:
exps_cuni_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
exps_cuni_ln(formula, data, ...)
```

Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in [0,1] with the lambda value.
- **sortid**: a logical indicating if the indices must be sorted at the output (default: TRUE).
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.
- **...**: other options to pass to the function.

Details

Exponential/smudge completely-uniform label noise includes an additional attribute (smudge) in the dataset with random values in [0,1]. This attribute is used to compute the mislabeling probability for each sample based on an exponential function (in which `level` is used as lambda). It selects samples in the dataset based on these probabilities. Finally, the labels of these samples are randomly replaced by others within the set of class labels (this model can choose the original label of a sample as noisy).

Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector list with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
distr an integer vector with the samples per class in the original data.
model the full name of the noise introduction model used.
param a list of the argument values.
call the function call.

Note
Noise model adapted from the papers in References.

References

See Also
opes_idu_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- exps_cuni_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.8)

# show results
summary(outdef, showid = TRUE)
plot(outdef, pca = TRUE)

# usage of the method for class formula
set.seed(9)
outfrm <- exps_cuni_ln(formula = Species ~ ., data = iris2D, level = 0.8)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

exp_bor_ln

Exponential borderline label noise

Description
Introduction of Exponential borderline label noise into a classification dataset.
exp_bor_ln

Usage

## Default S3 method:
ex_bor_ln(x, y, level, rate = 1, k = 1, sortid = TRUE, ...)

## S3 method for class 'formula'
ex_bor_ln(formula, data, ...)

Arguments

x a data frame of input attributes.
y a factor vector with the output class of each sample.
level a double in [0,1] with the noise level to be introduced.
rate a double with the rate for the exponential distribution (default: 1).
k an integer with the number of nearest neighbors to be used (default: 1).
sortid a logical indicating if the indices must be sorted at the output (default: TRUE).
... other options to pass to the function.
formula a formula with the output class and, at least, one input attribute.
data a data frame in which to interpret the variables in the formula.

Details

Exponential borderline label noise uses an SVM to induce the decision border in the dataset. For each sample, its distance to the decision border is computed. Then, an exponential distribution with parameter rate is used to compute the value for the probability density function associated to each distance. Finally, (level·100)% of the samples in the dataset are randomly selected to be mislabeled according to their values of the probability density function. For each noisy sample, the majority class among its k-nearest neighbors of a different class is chosen as the new label.

Value

An object of class ndmodel with elements:

xnoise a data frame with the noisy input attributes.
ynoise a factor vector with the noisy output class.
umnoise an integer vector with the amount of noisy samples per class.
idnoise an integer vector list with the indices of noisy samples.
umclean an integer vector with the amount of clean samples per class.
idclean an integer vector list with the indices of clean samples.
distr an integer vector with the samples per class in the original data.
model the full name of the noise introduction model used.
param a list of the argument values.
call the function call.
Note

Noise model adapted from the papers in References, considering SVM with linear kernel as classifier, a mislabeling process using the neighborhood of noisy samples and a noise level to control the number of errors in the data.

References


See Also

pmd_con_ln, clu_vot_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- exp_bor_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- exp_bor_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

fra_bdir_ln

*Fraud bidirectional label noise*

Description

Introduction of *Fraud bidirectional label noise* into a classification dataset.

Usage

```r
## Default S3 method:
frac_bdir_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
frac_bdir_ln(formula, data, ...)
```
fra_bdir_ln

Arguments

- **x**  
  a data frame of input attributes.

- **y**  
  a factor vector with the output class of each sample.

- **level**  
  a double in [0,1] with the noise level to be introduced.

- **sortid**  
  a logical indicating if the indices must be sorted at the output (default: TRUE).

- **formula**  
  a formula with the output class and, at least, one input attribute.

- **data**  
  a data frame in which to interpret the variables in the formula.

Details

*Fraud bidirectional label noise* randomly selects (level·100)\% of the samples from the minority class in the dataset and level·10 samples from the majority class. Then, minority class samples are mislabeled as belonging to the majority class and majority class samples are mislabeled as belonging to the minority class. In case of ties determining minority and majority classes, a random class is chosen among them.

Value

An object of class `ndmodel` with elements:

- **xnoise**  
  a data frame with the noisy input attributes.

- **ynoise**  
  a factor vector with the noisy output class.

- **numnoise**  
  an integer vector with the amount of noisy samples per class.

- **idnoise**  
  an integer vector list with the indices of noisy samples.

- **numclean**  
  an integer vector with the amount of clean samples per class.

- **idclean**  
  an integer vector list with the indices of clean samples.

- **distr**  
  an integer vector with the samples per class in the original data.

- **model**  
  the full name of the noise introduction model used.

- **param**  
  a list of the argument values.

- **call**  
  the function call.

Note

Noise model adapted from the papers in References.

References


See Also

`irs_bdir_ln`, `pai_bdir_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`
Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- fra_bdir_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- fra_bdir_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**gam_bor_ln**  
*Gamma borderline label noise*

**Description**

Introduction of *Gamma borderline label noise* into a classification dataset.

**Usage**

```r
## Default S3 method:
gam_bor_ln(x, y, level, shape = 1, rate = 0.5, k = 1, sortid = TRUE, ...)

## S3 method for class 'formula'
gam_bor_ln(formula, data, ...)
```

**Arguments**

- `x`  
a data frame of input attributes.
- `y`  
a factor vector with the output class of each sample.
- `level`  
a double in [0,1] with the noise level to be introduced.
- `shape`  
a double with the shape for the gamma distribution (default: 1)
- `rate`  
a double with the rate for the gamma distribution (default: 0.5).
- `k`  
an integer with the number of nearest neighbors to be used (default: 1).
- `sortid`  
a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...`  
other options to pass to the function.
- `formula`  
a formula with the output class and, at least, one input attribute.
- `data`  
a data frame in which to interpret the variables in the formula.
gam_bor_ln

Details

*Gamma borderline label noise* uses an SVM to induce the decision border in the dataset. For each sample, its distance to the decision border is computed. Then, a gamma distribution with parameters (shape, rate) is used to compute the value for the probability density function associated to each distance. Finally, \((\text{level}\cdot 100)\)% of the samples in the dataset are randomly selected to be mislabeled according to their values of the probability density function. For each noisy sample, the majority class among its \(k\)-nearest neighbors of a different class is chosen as the new label.

Value

An object of class `ndmodel` with elements:

- `xnoise` a data frame with the noisy input attributes.
- `ynoise` a factor vector with the noisy output class.
- `numnoise` an integer vector with the amount of noisy samples per class.
- `idnoise` an integer vector list with the indices of noisy samples.
- `numclean` an integer vector with the amount of clean samples per class.
- `idclean` an integer vector list with the indices of clean samples.
- `distr` an integer vector with the samples per class in the original data.
- `model` the full name of the noise introduction model used.
- `param` a list of the argument values.
- `call` the function call.

Note

Noise model adapted from the papers in References, considering SVM with linear kernel as classifier, a mislabeling process using the neighborhood of noisy samples and a noise level to control the number of errors in the data.

References


See Also

- `exp_bor_ln`, `pmd_con_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- gam_bor_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)
```
# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- gam_bor_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

gaum_bor_ln  

**Gaussian-mixture borderline label noise**

### Description

Introduction of *Gaussian-mixture borderline label noise* into a classification dataset.

### Usage

```r
## Default S3 method:
gaum_bor_ln(
x,  
y,  
level,  
mean = c(0, 2),  
sd = c(sqrt(0.5), sqrt(0.5)),  
w = c(0.5, 0.5),  
k = 1,  
sortid = TRUE,  
...  
)
```

```r
## S3 method for class 'formula'
gaum_bor_ln(formula, data, ...)
```

### Arguments

- **x**
a data frame of input attributes.
- **y**
a factor vector with the output class of each sample.
- **level**
a double in [0,1] with the noise level to be introduced.
- **mean**
a double vector with the mean for each Gaussian distribution (default: `c(0,2)`).
- **sd**
a double vector with the standard deviation for each Gaussian distribution (default: `c(sqrt(0.5),sqrt(0.5))`).
- **w**
a double vector with the weight for each Gaussian distribution (default: `c(0.5,0.5)`).
gaum_bor_ln

k
an integer with the number of nearest neighbors to be used (default: 1).
sortid
a logical indicating if the indices must be sorted at the output (default: TRUE).
...
other options to pass to the function.
formula
a formula with the output class and, at least, one input attribute.
data
a data frame in which to interpret the variables in the formula.

Details

Gaussian-mixture borderline label noise uses an SVM to induce the decision border in the dataset. For each sample, its distance to the decision border is computed. Then, a Gaussian mixture distribution with parameters (mean, sd) and weights $w$ is used to compute the value for the probability density function associated to each distance. Finally, $(1 - level\%)$ of the samples in the dataset are randomly selected to be mislabeled according to their values of the probability density function. For each noisy sample, the majority class among its k-nearest neighbors of a different class is chosen as the new label.

Value

An object of class ndmodel with elements:

- xnoise: a data frame with the noisy input attributes.
- ynoise: a factor vector with the noisy output class.
- numnoise: an integer vector with the amount of noisy samples per class.
- idnoise: an integer vector list with the indices of noisy samples.
- numclean: an integer vector with the amount of clean samples per class.
- idclean: an integer vector list with the indices of clean samples.
- distr: an integer vector with the samples per class in the original data.
- model: the full name of the noise introduction model used.
- param: a list of the argument values.
- call: the function call.

Note

Noise model adapted from the papers in References, considering SVM with linear kernel as classifier, a mislabeling process using the neighborhood of noisy samples and a noise level to control the number of errors in the data.

References


See Also

gau_bor_ln, sigb_uni_ln, print.ndmodel, summary.ndmodel, plot.ndmodel
Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- gau_bor_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- gau_bor_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

gau_bor_ln

Gaussian borderline label noise

Description

Introduction of Gaussian borderline label noise into a classification dataset.

Usage

## Default S3 method:
gau_bor_ln(x, y, level, mean = 0, sd = 1, k = 1, sortid = TRUE, ...)

## S3 method for class 'formula'
gau_bor_ln(formula, data, ...)

Arguments

x a data frame of input attributes.
y a factor vector with the output class of each sample.
level a double in [0,1] with the noise level to be introduced.
mean a double with the mean for the Gaussian distribution (default: 0).
sd a double with the standard deviation for the Gaussian distribution (default: 1).
k an integer with the number of nearest neighbors to be used (default: 1).
sortid a logical indicating if the indices must be sorted at the output (default: TRUE).
... other options to pass to the function.
formula a formula with the output class and, at least, one input attribute.
data a data frame in which to interpret the variables in the formula.
Details

**Gaussian borderline label noise** uses an SVM to induce the decision border in the dataset. For each sample, its distance to the decision border is computed. Then, a Gaussian distribution with parameters (**mean**, **sd**) is used to compute the value for the probability density function associated to each distance. Finally, (**level**·100)% of the samples in the dataset are randomly selected to be mislabeled according to their values of the probability density function. For each noisy sample, the majority class among its k-nearest neighbors of a different class is chosen as the new label.

Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.

Note

Noise model adapted from the papers in References to multiclass data, considering SVM with linear kernel as classifier, a mislabeling process using the neighborhood of noisy samples and a noise level to control the number of errors in the data.

References


See Also

- `sigb_uni_ln`, `larm_uni_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- gau_bor_ln(x = iris2D[,ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)
# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- gau_bor_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

glev_uni_ln  

Gaussian-level uniform label noise

Description

Introduction of *Gaussian-level uniform label noise* into a classification dataset.

Usage

## Default S3 method:
glev_uni_ln(x, y, level, sd = 0.01, sortid = TRUE, ...)

## S3 method for class `formula`
glev_uni_ln(formula, data, ...)

Arguments

- **x**  
a data frame of input attributes.
- **y**  
a factor vector with the output class of each sample.
- **level**  
a double in [0,1] with the noise level to be introduced.
- **sd**  
a double with the standard deviation for the Gaussian distribution (default: 0.01).
- **sortid**  
a logical indicating if the indices must be sorted at the output (default: TRUE).
- **...**  
other options to pass to the function.
- **formula**  
a formula with the output class and, at least, one input attribute.
- **data**  
a data frame in which to interpret the variables in the formula.

Details

For each sample, *Gaussian-level uniform label noise* assigns a random probability following a Gaussian distribution of mean = level and standard deviation sd. Noisy samples are chosen according to these probabilities. The labels of these samples are randomly replaced by other different ones within the set of class labels.
Value

An object of class ndmodel with elements:

- xnoise: a data frame with the noisy input attributes.
- ynoise: a factor vector with the noisy output class.
- numnoise: an integer vector with the amount of noisy samples per class.
- idnoise: an integer vector list with the indices of noisy samples.
- numclean: an integer vector with the amount of clean samples per class.
- idclean: an integer vector list with the indices of clean samples.
- distr: an integer vector with the samples per class in the original data.
- model: the full name of the noise introduction model used.
- param: a list of the argument values.
- call: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

sym_hienc_ln, sym_nexc_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- glev_uni_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- glev_uni_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
**hubp_uni_ln**

**Hubness-proportional uniform label noise**

**Description**

Introduction of *Hubness-proportional uniform label noise* into a classification dataset.

**Usage**

```r
## Default S3 method:
hubp_uni_ln(x, y, level, k = 3, sortid = TRUE, ...)

## S3 method for class 'formula'
hubp_uni_ln(formula, data, ...)
```

**Arguments**

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `k`: an integer with the number of neighbors to compute the hubness of each sample (default: 3).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

**Details**

*Hubness-proportional uniform label noise* is based on the presence of hubs in the dataset. It selects (level·100)% of the samples in the dataset using a discrete probability distribution based on the concept of hubness, which is computed using the nearest neighbors of each sample. Then, the class labels of these samples are randomly replaced by different ones from the `c` classes.

**Value**

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
imp_int_an

- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.

**Note**

Noise model adapted from the papers in References.

**References**


**See Also**

`smu_cuni_ln`, `onedUniLn`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

**Examples**

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- hubp_uni_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- hubp_uni_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**Description**

Introduction of *Importance interval-based attribute noise* into a classification dataset.
Usage

## Default S3 method:
imp_int_an(x, y, level, nbins = 10, ascending = TRUE, sortid = TRUE, ...)

## S3 method for class 'formula'
imp_int_an(formula, data, ...)

Arguments

- **x**
  - a data frame of input attributes.
- **y**
  - a factor vector with the output class of each sample.
- **level**
  - a double vector with the noise levels in [0,1] to be introduced into each attribute.
- **nbins**
  - an integer with the number of bins to create (default: 10).
- **ascending**
  - a boolean indicating how noise levels are assigned to attributes:
    - TRUE: the lowest noise level is assigned to the most important attribute (default value).
    - FALSE: the highest noise level is assigned to the most important attribute.
- **sortid**
  - a logical indicating if the indices must be sorted at the output (default: TRUE).
- **...**
  - other options to pass to the function.
- **formula**
  - a formula with the output class and, at least, one input attribute.
- **data**
  - a data frame in which to interpret the variables in the formula.

Details

The values in `level` are ordered and assigned to attributes according to their information gain (using the ordering given by `ascending`). Then, Importance interval-based attribute noise corrupts (level[i]-100)% of the values for each attribute $A[i]$ in the dataset. In order to corrupt each attribute $A[i]$, (level[i]-100)% of the samples in the dataset are chosen. To corrupt a value in numeric attributes, the attribute is split into equal-frequency intervals, one of its closest intervals is picked out and a random value within the interval is chosen as noisy. For nominal attributes, a random value within the domain is chosen.

Value

An object of class `ndmodel` with elements:

- **xnoise**
  - a data frame with the noisy input attributes.
- **ynoise**
  - a factor vector with the noisy output class.
- **numnoise**
  - an integer vector with the amount of noisy samples per attribute.
- **idnoise**
  - an integer vector list with the indices of noisy samples per attribute.
- **numclean**
  - an integer vector with the amount of clean samples per attribute.
- **idclean**
  - an integer vector list with the indices of clean samples per attribute.
- **distr**
  - an integer vector with the samples per class in the original data.
- **model**
  - the full name of the noise introduction model used.
- **param**
  - a list of the argument values.
- **call**
  - the function call.
Note

Noise model adapted from the papers in References.

References


See Also

assy_int_an, asy_uni_an, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- imp_int_an(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)],
level = c(0.1, 0.2))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- imp_int_an(formula = Species ~ ., data = iris2D,
level = c(0.1, 0.2))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

iris2D

iris2D dataset

Description

A 2-dimensional version of the well-known iris dataset. It maintains the attributes Petal.Length and Petal.Width, which give the measurements in centimeters of the petal length and width of iris flowers belonging to three different species (setosa, versicolor and virginica). Duplicate and contradictory samples are removed from the dataset, resulting in a total of 103 samples.

Usage

data(iris2D)
## Format

A data.frame with 103 samples (rows) and 3 variables (columns) named Petal.Length, Petal.Width and Species.

## Source

Data collected by E. Anderson (1935).

## References


## See Also

`sym_uni_ln`, `sym_uni_an`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

## Examples

```r
library(ggplot2)
library(RColorBrewer)
data(iris2D)
ggplot(data = iris2D, aes(x = iris2D[,1], y = iris2D[,2], color = iris2D[,3])) +
  geom_point(stroke = 0.5) +
  xlim(min(iris2D[,1]), max(iris2D[,1])) +
  ylim(min(iris2D[,2]), max(iris2D[,2])) +
  xlab(names(iris2D)[1]) +
  ylab(names(iris2D)[2]) +
  labs(color='Species') +
  scale_color_manual(values = brewer.pal(3, "Dark2")) +
  theme(panel.border = element_rect(colour = "black", fill=NA),
        aspect.ratio = 1,
        axis.text = element_text(colour = 1, size = 12),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black"))
```

---

**irs_bdir_ln**  
**IR-stable bidirectional label noise**

## Description

Introduction of *IR-stable bidirectional label noise* into a classification dataset.
Usage

```r
## Default S3 method:
irs_bdir_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
irs_bdir_ln(formula, data, ...)
```

Arguments

- `x` a data frame of input attributes.
- `y` a factor vector with the output class of each sample.
- `level` a double in [0,1] with the noise level to be introduced.
- `sortid` a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `...` other options to pass to the function.
- `formula` a formula with the output class and, at least, one input attribute.
- `data` a data frame in which to interpret the variables in the formula.

Details

IR-stable bidirectional label noise randomly selects `(level·100)%` of the samples from the minority class in the dataset and the same amount of samples from the majority class. Then, minority class samples are mislabeled as belonging to the majority class and majority class samples are mislabeled as belonging to the minority class. In case of ties determining minority and majority classes, a random class is chosen among them.

Value

An object of class `ndmodel` with elements:

- `xnoise` a data frame with the noisy input attributes.
- `ynoise` a factor vector with the noisy output class.
- `numnoise` an integer vector with the amount of noisy samples per class.
- `idnoise` an integer vector list with the indices of noisy samples.
- `numclean` an integer vector with the amount of clean samples per class.
- `idclean` an integer vector list with the indices of clean samples.
- `distr` an integer vector with the samples per class in the original data.
- `model` the full name of the noise introduction model used.
- `param` a list of the argument values.
- `call` the function call.

Note

Noise model adapted from the papers in References.
References


See Also

`pai_bdir_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- lap_bor_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- lap_bor_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

`lap_bor_ln`  
**Laplace borderline label noise**

Description

Introduction of *Laplace borderline label noise* into a classification dataset.

Usage

```r
## Default S3 method:
lap_bor_ln(x, y, level, mu = 0, b = 1, k = 1, sortid = TRUE, ...)

## S3 method for class 'formula'
lap_bor_ln(formula, data, ...)
```
Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in [0,1] with the noise level to be introduced.
- **mu**: a double with the location for the Laplace distribution (default: 0).
- **b**: a double with the scale for the Laplace distribution (default: 1).
- **k**: an integer with the number of nearest neighbors to be used (default: 1).
- **sortid**: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

Details

**Laplace borderline label noise** uses an SVM to induce the decision border in the dataset. For each sample, its distance to the decision border is computed. Then, a Laplace distribution with parameters (**mu**, **b**) is used to compute the value for the probability density function associated to each distance. Finally, `(level·100)%` of the samples in the dataset are randomly selected to be mislabeled according to their values of the probability density function. For each noisy sample, the majority class among its `k`-nearest neighbors of a different class is chosen as the new label.

Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

Note

Noise model adapted from the papers in References to multiclass data, considering SVM with linear kernel as classifier, a mislabeling process using the neighborhood of noisy samples and a noise level to control the number of errors in the data.
References


See Also

ugau_bor_ln, gaum_bor_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- lap_bor_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- lap_bor_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

larm_uni_ln

Large-margin uniform label noise

Description

Introduction of Large-margin uniform label noise into a classification dataset.

Usage

## Default S3 method:
larm_uni_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
larm_uni_ln(formula, data, ...)

Arguments

- **x**  
  A data frame of input attributes.

- **y**  
  A factor vector with the output class of each sample.

- **level**  
  A double in [0,1] with the noise level to be introduced.

- **sortid**  
  A logical indicating if the indices must be sorted at the output (default: TRUE).

- **formula**  
  A formula with the output class and, at least, one input attribute.

- **data**  
  A data frame in which to interpret the variables in the formula.

Details

**Large-margin uniform label noise** uses an SVM to induce the decision border in the dataset. For each sample, its distance to the decision border is computed. Then, the samples are ordered according to their distance and (level·100)% of the most distant correctly classified samples to the decision boundary are selected to be mislabeled with a random different class.

Value

An object of class ndmodel with elements:

- **xnoise**  
  A data frame with the noisy input attributes.

- **ynoise**  
  A factor vector with the noisy output class.

- **numnoise**  
  An integer vector with the amount of noisy samples per class.

- **idnoise**  
  An integer vector list with the indices of noisy samples.

- **numclean**  
  An integer vector with the amount of clean samples per class.

- **idclean**  
  An integer vector list with the indices of clean samples.

- **distr**  
  An integer vector with the samples per class in the original data.

- **model**  
  The full name of the noise introduction model used.

- **param**  
  A list of the argument values.

- **call**  
  The function call.

Note

Noise model adapted from the papers in References to multiclass data, considering SVM with linear kernel as classifier.

References


See Also

hubp_uni_ln, smu_cuni_ln, print.ndmodel, summary.ndmodel, plot.ndmodel
Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- larm_uni_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.3)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- larm_uni_ln(formula = Species ~ ., data = iris2D, level = 0.3)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

**maj_udir_ln**

**Majority-class unidirectional label noise**

**Description**

Introduction of *Majority-class unidirectional label noise* into a classification dataset.

**Usage**

```r
## Default S3 method:
maj_udir_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
maj_udir_ln(formula, data, ...)
```

**Arguments**

- `x` a data frame of input attributes.
- `y` a factor vector with the output class of each sample.
- `level` a double in \([0,1]\) with the noise level to be introduced.
- `sortid` a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `...` other options to pass to the function.
- `formula` a formula with the output class and, at least, one input attribute.
- `data` a data frame in which to interpret the variables in the formula.
Details

Let \( A \) be the majority class and \( B \) be the second majority class in the dataset. The *Majority-class unidirectional label noise* introduction model randomly selects \((\text{level} \cdot 100)\%\) of the samples of \( A \) and labels them as \( B \).

Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.

Note

Noise model adapted from the papers in References to multiclass data.

References


See Also

asy_def_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- maj_udir_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)
```
# usage of the method for class formula
set.seed(9)
outfrm <- maj_udir_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

mind_bdir_ln  

Minority-driven bidirectional label noise

Description
Introduction of Minority-driven bidirectional label noise into a classification dataset.

Usage

```r
## Default S3 method:
mind_bdir_ln(x, y, level, pos = 0.1, sortid = TRUE, ...)
```

```r
## S3 method for class 'formula'
mind_bdir_ln(formula, data, ...)
```

Arguments

- `x`  
a data frame of input attributes.
- `y`  
a factor vector with the output class of each sample.
- `level`  
a double in [0,1] with the noise level to be introduced.
- `pos`  
a double in [0,1] with the proportion of samples from the positive class (default: 0.1).
- `sortid`  
a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...`  
other options to pass to the function.
- `formula`  
a formula with the output class and, at least, one input attribute.
- `data`  
a data frame in which to interpret the variables in the formula.

Details

Minority-driven bidirectional label noise randomly selects $n = 2m \cdot \text{level}$ samples in the dataset (with $m$ the number of samples in the minority class), making sure that $n \cdot \text{pos}$ samples belong to the minority class and the rest to the majority class. Then, minority class samples are mislabeled as belonging to the majority class and majority class samples are mislabeled as belonging to the minority class. In case of ties determining minority and majority classes, a random class is chosen among them.
Value

An object of class ndmodel with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.

Note

Noise model adapted from the papers in References to multiclass data.

References


See Also

fra_bdir_ln, irs_bdir_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- mind_bdir_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.5)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- mind_bdir_ln(formula = Species ~ ., data = iris2D, level = 0.5)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```
Description

Introduction of *Minority-proportional uniform label noise* into a classification dataset.

Usage

```r
## Default S3 method:
minp_uni_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
minp_uni_ln(formula, data, ...)
```

Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

Details

Given a dataset, assume the original class distribution of class $i$ is $p_i$ and the distribution of the minority class is $p_m$. Let `level` be the noise level, *Minority-proportional uniform label noise* introduces noise proportionally to different classes, where a sample with its label $i$ has a probability $(p_m/p_i)\cdot \text{level}$ to be corrupted as another random class. That is, the least common class is used as the baseline for noise introduction.

Value

An object of class `nmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
- `distr`: an integer vector with the samples per class in the original data.
model the full name of the noise introduction model used.
param a list of the argument values.
call the function call.

Note

Noise model adapted from the papers in References.

References


See Also

asy_uni_ln, maj_udir_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- minp_uni_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- minp_uni_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

Description

Introduction of *Misclassification prediction label noise* into a classification dataset.
Usage

## Default S3 method:
mis_pre_ln(x, y, sortid = TRUE, ...)

## S3 method for class 'formula'
mis_pre_ln(formula, data, ...)

Arguments

x
  a data frame of input attributes.

y
  a factor vector with the output class of each sample.

sortid
  a logical indicating if the indices must be sorted at the output (default: TRUE).

... other options to pass to the function.

formula
  a formula with the output class and, at least, one input attribute.

data
  a data frame in which to interpret the variables in the formula.

Details

Misclassification prediction label noise creates a Multi-Layer Perceptron (MLP) model from the
data set and relabels each sample with the class predicted by the classifier.

Value

An object of class ndmodel with elements:

xnoise
  a data frame with the noisy input attributes.

ynoise
  a factor vector with the noisy output class.

numnoise
  an integer vector with the amount of noisy samples per class.

idnoise
  an integer vector list with the indices of noisy samples.

numclean
  an integer vector with the amount of clean samples per class.

idclean
  an integer vector list with the indices of clean samples.

distr
  an integer vector with the samples per class in the original data.

model
  the full name of the noise introduction model used.

param
  a list of the argument values.

call
  the function call.

Note

Noise model adapted from the papers in References.

References

Q. Wang, B. Han, T. Liu, G. Niu, J. Yang, and C. Gong. Tackling instance-dependent label noise
via a universal probabilistic model. In Proc. 35th AAAI Conference on Artificial Intelligence,
mulc_udir_ln

**See Also**

```r
smam_bor_ln, nlin_bor_ln, print.ndmodel, summary.ndmodel, plot.ndmodel
```

**Examples**

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- mis_pre_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)])

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- mis_pre_ln(formula = Species ~ ., data = iris2D)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**mulc_udir_ln**  
*Multiple-class unidirectional label noise*

**Description**

Introduction of **Multiple-class unidirectional label noise** into a classification dataset.

**Usage**

```r
## Default S3 method:
mulc_udir_ln(x, y, level, goal, order = levels(y), sortid = TRUE, ...)

## S3 method for class 'formula'
mulc_udir_ln(formula, data, ...)
```

**Arguments**

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `goal`: an integer vector with the indices of noisy classes for each class.
- `order`: a character vector indicating the order of the classes (default: `levels(y)`).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
... other options to pass to the function.

- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

### Details

**Multiple-class unidirectional label noise** introduction model randomly selects (level\*100)% of the samples of each class $c$ with $\text{goal}[c] \neq \text{NA}$. Then, the labels $c$ of these samples are replaced by the class indicated in $\text{goal}[c]$. The order of indices in $\text{goal}$ is determined by $\text{order}$.

### Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

### Note

Noise model adapted from the papers in References.

### References


### See Also

`minp_uni_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

### Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- mulc_udir_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1,
```
nei_bor_ln

```
goal = c(NA, 1, 2), order = c("virginica", "setosa", "versicolor"))
# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- mulc_udir_ln(formula = Species ~ ., data = iris2D, level = 0.1,
goal = c(NA, 1, 2), order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**Description**

Introduction of *Neighborwise borderline label noise* into a classification dataset.

**Usage**

```
## Default S3 method:
nei_bor_ln(x, y, level, k = 1, sortid = TRUE, ...)

## S3 method for class 'formula'
nei_bor_ln(formula, data, ...)
```

**Arguments**

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `k`: an integer with the number of nearest neighbors to be used (default: 1).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

**Details**

For each sample in the dataset, *Neighborwise borderline label noise* computes the ratio of two distances: the distance to its nearest neighbor from the same class and the distance to its nearest neighbor from another class. Then, these values are ordered in descending order and the first (level·100)% of them are used to determine the noisy samples. For each noisy sample, the majority class among its k-nearest neighbors of a different class is chosen as the new label.
Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

Note

Noise model adapted from the papers in References, considering a mislabeling process using the neighborhood of noisy samples.

References


See Also

`ulap_bor_ln`, `lap_bor_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- nei_bor_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- nei_bor_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```
Description

Introduction of *Non-linearwise borderline label noise* into a classification dataset.

Usage

```r
## Default S3 method:
nlin_bor_ln(x, y, level, k = 1, sortid = TRUE, ...)

## S3 method for class 'formula'
nlin_bor_ln(formula, data, ...)
```

Arguments

- `x` a data frame of input attributes.
- `y` a factor vector with the output class of each sample.
- `level` a double in [0,1] with the noise level to be introduced.
- `k` an integer with the number of nearest neighbors to be used (default: 1).
- `sortid` a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...` other options to pass to the function.
- `formula` a formula with the output class and, at least, one input attribute.
- `data` a data frame in which to interpret the variables in the formula.

Details

*Non-linearwise borderline label noise* uses an SVM to induce the decision border in the dataset. Then, for each sample, its distance to the decision border is computed. Finally, the distances obtained are ordered in ascending order and the first (level·100)% of them are used to determine the noisy samples. For each noisy sample, the majority class among its k-nearest neighbors of a different class is chosen as the new label.

Value

An object of class `nmodel` with elements:

- `xnoise` a data frame with the noisy input attributes.
- `ynoise` a factor vector with the noisy output class.
- `numnoise` an integer vector with the amount of noisy samples per class.
- `idnoise` an integer vector list with the indices of noisy samples.
- `numclean` an integer vector list with the amounts of clean samples per class.
- `idclean` an integer vector list with the indices of clean samples.
oned_uni_ln

Description

Introduction of One-dimensional uniform label noise into a classification dataset.

oned_uni_ln

One-dimensional uniform label noise

Description

Introduction of One-dimensional uniform label noise into a classification dataset.

oned_uni_ln

One-dimensional uniform label noise

Description

Introduction of One-dimensional uniform label noise into a classification dataset.
oned_uni_ln

Usage

## Default S3 method:
oned_uni_ln(
  x,
  y,
  level,
  att,
  lower,
  upper,
  order = levels(y),
  sortid = TRUE,
  ...
)

## S3 method for class 'formula'
oned_uni_ln(formula, data, ...)

Arguments

x a data frame of input attributes.
y a factor vector with the output class of each sample.
level a double in [0,1] with the noise level to be introduced.
att an integer with the index of the attribute determining noisy samples.
lower a vector with the lower bound to determine the noisy region of each class.
upper a vector with the upper bound to determine the noisy region of each class.
order a character vector indicating the order of the classes (default: levels(y)).
sortid a logical indicating if the indices must be sorted at the output (default: TRUE).
... other options to pass to the function.
formula a formula with the output class and, at least, one input attribute.
data a data frame in which to interpret the variables in the formula.

Details

One-dimensional uniform label noise is based on the introduction of noise according to the values of the attribute att. Samples of class i with the attribute att falling between lower[i] and upper[i] have a probability level of being mislabeled. The labels of these samples are randomly replaced by other different ones within the set of class labels. The order of the class labels is determined by order.

Value

An object of class ndmodel with elements:
xnoise a data frame with the noisy input attributes.
ynoise a factor vector with the noisy output class.
numnoise an integer vector with the amount of noisy samples per class.
idnoise an integer vector list with the indices of noisy samples.
numclean an integer vector with the amount of clean samples per class.
idclean an integer vector list with the indices of clean samples.
distr an integer vector with the samples per class in the original data.
model the full name of the noise introduction model used.
param a list of the argument values.
call the function call.

Note
Noise model adapted from the papers in References to multiclass data, considering a noise level to control the number of errors in the data

References

See Also
attm_uni_ln, qua_uni_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- oned_uni_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)],
      level = 0.5, att = 1, lower = c(1.5, 2, 6), upper = c(2, 4, 7))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- oned_uni_ln(formula = Species ~ ., data = iris2D,
      level = 0.5, att = 1, lower = c(1.5, 2, 6), upper = c(2, 4, 7))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
Description

Introduction of Open-set ID/nearest-neighbor label noise into a classification dataset.

Usage

```r
## Default S3 method:
opes_idnn_ln(
  x,
  y,
  level,
  openset = c(1),
  order = levels(y),
  sortid = TRUE,
  ...
)

## S3 method for class 'formula'
opes_idnn_ln(formula, data, ...)
```

Arguments

- `x`  
  a data frame of input attributes.

- `y`  
  a factor vector with the output class of each sample.

- `level`  
  a double with the noise level in [0,1] to be introduced.

- `openset`  
  an integer vector with the indices of classes in the open set (default: c(1)).

- `order`  
  a character vector indicating the order of the classes (default: levels(y)).

- `sortid`  
  a logical indicating if the indices must be sorted at the output (default: TRUE).

- `formula`  
  a formula with the output class and, at least, one input attribute.

- `data`  
  a data frame in which to interpret the variables in the formula.

Details

Open-set ID/nearest-neighbor label noise corrupts (level·100)% of the samples with classes in openset. Then, the labels of these samples are replaced by the label of the nearest sample of a different in-distribution class. The order of the class labels for the indices in openset is determined by order.
Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

- `opes_idu_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- opes_idnn_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)],
    level = 0.4, order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- opes_idnn_ln(formula = Species ~ ., data = iris2D,
    level = 0.4, order = c("virginica", "setosa", "versicolor"))
```
# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

## opes_idu_ln

### Open-set ID/uniform label noise

**Description**

Introduction of *Open-set ID/uniform label noise* into a classification dataset.

**Usage**

```r
# Default S3 method:
opes_idu_ln(x, y, level, openset = c(1), order = levels(y), sortid = TRUE, ...)

# S3 method for class 'formula'
opes_idu_ln(formula, data, ...)
```

**Arguments**

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double with the noise level in \([0,1]\) to be introduced.
- `openset`: an integer vector with the indices of classes in the open set (default: `c(1)`).
- `order`: a character vector indicating the order of the classes (default: `levels(y)`).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

**Details**

*Open-set ID/uniform label noise* corrupts \((\text{level} \times 100)\%\) of the samples with classes in `openset`. For each sample selected, a label from in-distribution classes is randomly chosen. The order of the class labels for the indices in `openset` is determined by `order`.

**Value**

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
numclean an integer vector with the amount of clean samples per class.
idclean an integer vector list with the indices of clean samples.
distr an integer vector with the samples per class in the original data.
model the full name of the noise introduction model used.
param a list of the argument values.
call the function call.

Note
Noise model adapted from the papers in References.

References

See Also
asy_spa_ln, mind_bdir_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- opes_idu_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)],
    level = 0.4, order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- opes_idu_ln(formula = Species ~ ., data = iris2D,
    level = 0.4, order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
Pairwise bidirectional label noise

Description
Introduction of *Pairwise bidirectional label noise* into a classification dataset.

Usage

```
## Default S3 method:
pai_bdir_ln(x, y, level, pairs, order = levels(y), sortid = TRUE, ...)

## S3 method for class 'formula'
pai_bdir_ln(formula, data, ...)
```

Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `pairs`: a list of integer vectors with the indices of classes to corrupt.
- `order`: a character vector indicating the order of the classes (default: `levels(y)`).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

Details
For each vector \((c1, c2)\) in `pairs`, *Pairwise bidirectional label noise* randomly selects \((level \cdot 100)\)% of the samples from class \(c1\) in the dataset and \((level \cdot 100)\)% of the samples from class \(c2\). Then, \(c1\) samples are mislabeled as belonging to \(c2\) and \(c2\) samples are mislabeled as belonging to \(c1\). The order of the class labels is determined by `order`.

Value
An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
pai_bdir_ln

distr an integer vector with the samples per class in the original data.
model the full name of the noise introduction model used.
param a list of the argument values.
call the function call.

Note
Noise model adapted from the papers in References.

References

See Also
print.ndmodel, summary.ndmodel, plot.ndmodel

Examples
# load the dataset
data(iris2D)

# create new class with some samples
class <- as.character(iris2D$Species)
class[iris2D$Petal.Length > 6] <- "newclass"
iris2D$Species <- as.factor(class)

# usage of the default method
set.seed(9)
outdef <- pai_bdir_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)],
  level = 0.1, pairs = list(c(1,2), c(3,4)),
  order = c("virginica", "setosa", "newclass", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- pai_bdir_ln(formula = Species ~ ., data = iris2D,
  level = 0.1, pairs = list(c(1,2), c(3,4)),
  order = c("virginica", "setosa", "newclass", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
Description

Representation of the dataset contained in an object of class ndmodel after the application of a noise introduction model.

Usage

```r
## S3 method for class 'ndmodel'
plot(x, ..., noise = NA, xvar = 1, yvar = 2, pca = FALSE)
```

Arguments

- `x`: an object of class ndmodel.
- `...`: other options to pass to the function.
- `noise`: a logical indicating which samples to show. The valid options are:
  - TRUE: to show only the noisy samples.
  - FALSE: to show only the clean samples.
  - NA: to show both the clean and noisy samples (default value).
- `xvar`: an integer with the index of the input attribute (if pca = FALSE) or the principal component (if pca = TRUE) to represent in the x axis (default: 1).
- `yvar`: an integer with the index of the input attribute (if pca = FALSE) or the principal component (if pca = TRUE) to represent in the y axis (default: 2).
- `pca`: a logical indicating if PCA must be used (default: FALSE).

Details

This function performs a two-dimensional representation using the ggplot2 package of the dataset contained in the object `x` of class ndmodel. Each of the classes in the dataset (available in `x$ynoise`) is represented by a different color. There are two options to represent the input attributes of the samples on the x and y axes of the graph:

- If `pca = FALSE`, the values in the graph are taken from the current attribute values found in `x$xnoise`. In this case, `xvar` and `yvar` indicate the indices of the attributes to show in the x and y axes, respectively.
- If `pca = TRUE`, the values in the graph are taken after performing a PCA over `x$xnoise`. In this case, `xvar` and `yvar` indicate the index of the principal component according to the variance explained to show in the x and y axes, respectively.

Finally, the parameter `noise` is used to indicate which samples (noisy, clean or all) to show. Clean samples are represented by circles in the graph, while noisy samples are represented by crosses.
Value

An object of class ggplot and gg with the graph created using the ggplot2 package.

See Also

print.ndmodel, summary.ndmodel, sym_uni_ln, sym_cuni_ln, sym_uni_an

Examples

# load the dataset
data(iris)

# apply the noise introduction model
set.seed(9)
output <- sym_uni_ln(x = iris[, -ncol(iris)], y = iris[, ncol(iris)], level = 0.1)

# plots for all the samples, the clean samples and the noisy samples using PCA
plot(output, pca = TRUE)
plot(output, noise = FALSE, pca = TRUE)
plot(output, noise = TRUE, pca = TRUE)

# plots using the Petal.Length and Petal.Width variables
plot(output, xvar = 3, yvar = 4)
plot(output, noise = FALSE, xvar = 3, yvar = 4)
plot(output, noise = TRUE, xvar = 3, yvar = 4)

pmd_con_ln

PMD-based confidence label noise

Description

Introduction of PMD-based confidence label noise into a classification dataset.

Usage

## Default S3 method:
pmd_con_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
pmd_con_ln(formula, data, ...)

Arguments

x  a data frame of input attributes.
y  a factor vector with the output class of each sample.
level  a double in [0,1] with the noise level to be introduced.
sortid  a logical indicating if the indices must be sorted at the output (default: TRUE).
... other options to pass to the function.

formula a formula with the output class and, at least, one input attribute.
data a data frame in which to interpret the variables in the formula.

Details

PMD-based confidence label noise approximates the probability of noise using the confidence prediction of a neural network. These predictions are used to estimate the mislabeling probability and the most possible noisy class label for each sample. Finally, \((\text{level}\times100)\%\) of the samples in the dataset are randomly selected to be mislabeled according to their values of probability computed.

Value

An object of class ndmodel with elements:

- xnoise a data frame with the noisy input attributes.
- ynoise a factor vector with the noisy output class.
- numnoise an integer vector with the amount of noisy samples per class.
- idnoise an integer vector list with the indices of noisy samples.
- numclean an integer vector with the amount of clean samples per class.
- idclean an integer vector list with the indices of clean samples.
- distr an integer vector with the samples per class in the original data.
- model the full name of the noise introduction model used.
- param a list of the argument values.
- call the function call.

Note

Noise model adapted from the papers in References.

References


See Also

clu_vot_ln, sco_con_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- pmd_con_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- pmd_con_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

**print.ndmodel**  
*Print function for class ndmodel*

**Description**

This method displays the basic information about the noise introduction process contained in an object of class ndmodel.

**Usage**

```r
## S3 method for class 'ndmodel'
print(x, ...)
```

**Arguments**

- `x` an object of class ndmodel.
- `...` other options to pass to the function.

**Details**

This function presents the basic information of the noise introduction process and the resulting noisy dataset contained in the object `x` of class ndmodel. The information offered is as follows:

- the name of the noise introduction model.
- the parameters associated with the noise model.
- the number of noisy and clean samples in the dataset.

**Value**

This function does not return any value.

**See Also**

`summary.ndmodel, plot.ndmodel, sym_uni_ln, sym_cuni_ln, sym_uni_an`
### Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_uni_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
print(outdef)
```

---

**qua_uni_ln**  
*Quadrant-based uniform label noise*

### Description

Introduction of *Quadrant-based uniform label noise* into a classification dataset.

### Usage

```r
## Default S3 method:
qua_uni_ln(x, y, level, att1 = 1, att2 = 2, sortid = TRUE, ...)

## S3 method for class 'formula'
qua_uni_ln(formula, data, ...)
```

### Arguments

- `x`  
a data frame of input attributes.
- `y`  
a factor vector with the output class of each sample.
- `level`  
a double vector with the noise levels in [0,1] in each quadrant.
- `att1`  
an integer with the index of the first attribute forming the quadrants (default: 1).
- `att2`  
an integer with the index of the second attribute forming the quadrants (default: 2).
- `sortid`  
a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...`  
other options to pass to the function.
- `formula`  
a formula with the output class and, at least, one input attribute.
- `data`  
a data frame in which to interpret the variables in the formula.
Details
For each sample, the probability of flipping its label is based on which quadrant (with respect to the attributes \texttt{att1} and \texttt{att2}) the sample falls in. The probability of mislabeling for each quadrant is expressed with the argument \texttt{level}, whose length is equal to 4. Let \( m1 \) and \( m2 \) be the mean values of the domain of \texttt{att1} and \texttt{att2}, respectively. Each quadrant is defined as follows: values \( \leq m1 \) and \( \leq m2 \) (first quadrant); values \( \leq m1 \) and \( > m2 \) (second quadrant); values \( > m1 \) and \( \leq m2 \) (third quadrant); and values \( > m1 \) and \( > m2 \) (fourth quadrant). Finally, the labels of these samples are randomly replaced by other different ones within the set of class labels.

Value
An object of class \texttt{ndmodel} with elements:

- \texttt{xnoise} a data frame with the noisy input attributes.
- \texttt{ynoise} a factor vector with the noisy output class.
- \texttt{numnoise} an integer vector with the amount of noisy samples per class.
- \texttt{idnoise} an integer vector list with the indices of noisy samples.
- \texttt{numclean} an integer vector with the amount of clean samples per class.
- \texttt{idclean} an integer vector list with the indices of clean samples.
- \texttt{distr} an integer vector with the samples per class in the original data.
- \texttt{model} the full name of the noise introduction model used.
- \texttt{param} a list of the argument values.
- \texttt{call} the function call.

Note
Noise model adapted from the papers in References.

References

See Also
\texttt{exps_cuni_ln}, \texttt{print.ndmodel}, \texttt{summary.ndmodel}, \texttt{plot.ndmodel}

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- qua_uni_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)],
                      level = c(0.05, 0.15, 0.20, 0.4))
```
sco_con_ln

Score-based confidence label noise

Description

Introduction of Score-based confidence label noise into a classification dataset.

Usage

## Default S3 method:
sco_con_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
sco_con_ln(formula, data, ...)

Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in \([0,1]\) with the noise level to be introduced.
- **sortid**: a logical indicating if the indices must be sorted at the output (default: TRUE).
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.
- **...**: other options to pass to the function.

Details

Score-based confidence label noise follows the intuition that hard samples are more likely to be mislabeled. Given the confidence per class of each sample, if it is predicted with a different class with a high probability, it means that it is hard to clearly distinguish the sample from this class. The confidence information is used to compute a mislabeling score for each sample and its potential noisy label. Finally, \((\text{level} \cdot 100)\%\) of the samples with the highest mislabeling scores are chosen as noisy.
Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

- `mis_pre_ln`, `smam_bor_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sco_con_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sco_con_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```
Description

Introduction of \textit{Sigmoid-bounded uniform label noise} into a classification dataset.

Usage

### Default S3 method:
\begin{verbatim}
sigb_uni_ln(x, y, level, order = levels(y), sortid = TRUE, ...)
\end{verbatim}

### S3 method for class 'formula'
\begin{verbatim}
sigb_uni_ln(formula, data, ...)
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{x} a data frame of input attributes.
\item \texttt{y} a factor vector with the output class of each sample.
\item \texttt{level} a double vector with the noise levels in \([0,1]\) to be introduced into each class.
\item \texttt{order} a character vector indicating the order of the classes (default: \texttt{levels(y)}).
\item \texttt{sortid} a logical indicating if the indices must be sorted at the output (default: \texttt{TRUE}).
\item \texttt{...} other options to pass to the function.
\item \texttt{formula} a formula with the output class and, at least, one input attribute.
\item \texttt{data} a data frame in which to interpret the variables in the formula.
\end{itemize}

Details

\textit{Sigmoid-bounded uniform label noise} generates bounded instance-dependent and label-dependent label noise at random using a weight for each sample in the dataset to compute its noise probability through a sigmoid function. Note that this noise model considers the maximum noise level per class given by \texttt{level}, so the current noise level in each class may be lower than that specified. The order of the class labels is determined by \texttt{order}.

Value

An object of class \texttt{ndmodel} with elements:

\begin{itemize}
\item \texttt{xnoise} a data frame with the noisy input attributes.
\item \texttt{ynoise} a factor vector with the noisy output class.
\item \texttt{numnoise} an integer vector with the amount of noisy samples per class.
\item \texttt{idnoise} an integer vector list with the indices of noisy samples.
\item \texttt{numclean} an integer vector with the amount of clean samples per class.
\item \texttt{idclean} an integer vector list with the indices of clean samples.
\end{itemize}
Small-margin borderline label noise

Description

Introduction of Small-margin borderline label noise into a classification dataset.

distr an integer vector with the samples per class in the original data.
model the full name of the noise introduction model used.
param a list of the argument values.
call the function call.

Note

Noise model adapted from the papers in References to multiclass data.

References


See Also

larm_uni_ln, hubp_uni_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sigb_uni_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)],
    level = c(0.1, 0.2, 0.3))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sigb_uni_ln(formula = Species ~ ., data = iris2D,
    level = c(0.1, 0.2, 0.3))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
smam_bor_ln

Usage

## Default S3 method:
smam_bor_ln(x, y, level, k = 1, sortid = TRUE, ...)

## S3 method for class 'formula'
smam_bor_ln(formula, data, ...)

Arguments

x       a data frame of input attributes.
y       a factor vector with the output class of each sample.
level   a double in [0,1] with the noise level to be introduced.
k       an integer with the number of nearest neighbors to be used (default: 1).
sortid  a logical indicating if the indices must be sorted at the output (default: TRUE).
...     other options to pass to the function.
formula a formula with the output class and, at least, one input attribute.
data    a data frame in which to interpret the variables in the formula.

Details

*Small-margin borderline label noise* uses an SVM to induce the decision border in the dataset. For each sample, its distance to the decision border is computed. Then, the samples are ordered according to their distance and (level·100)% of the closest correctly classified samples to the decision boundary are selected to be mislabeled. For each noisy sample, the majority class among its k-nearest neighbors of a different class is chosen as the new label.

Value

An object of class ndmodel with elements:

xnoise    a data frame with the noisy input attributes.
ynoise    a factor vector with the noisy output class.
umnoise   an integer vector with the amount of noisy samples per class.
idnoise   an integer vector list with the indices of noisy samples.
umclean  an integer vector with the amount of clean samples per class.
idclean  an integer vector list with the indices of clean samples.
distr     an integer vector with the samples per class in the original data.
model     the full name of the noise introduction model used.
param     a list of the argument values.
call      the function call.

Note

Noise model adapted from the papers in References to multiclass data, considering SVM with linear kernel as classifier and a mislabeling process using the neighborhood of noisy samples.
References


See Also

nlin_bor_ln, nei_bor_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- smam_bor_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- smam_bor_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

smu_cuni_ln Smudge-based completely-uniform label noise

Description

Introduction of Smudge-based completely-uniform label noise into a classification dataset.

Usage

## Default S3 method:
smu_cuni_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
smu_cuni_ln(formula, data, ...)
Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in \([0,1]\) with the noise level to be introduced.
- **sortid**: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

Details

*Smudge-based completely-uniform label noise* randomly selects \((\text{level} \cdot 100)\)% of the samples in the dataset with independence of their class. Then, the labels of these samples are randomly replaced by others within the set of class labels. An additional attribute `smudge` is included in the dataset with value equal to 1 in mislabeled samples and equal to 0 in clean samples.

Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

`oned_uni_ln`, `attm_uni_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`
Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- smu_cuni_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef, pca = TRUE)

# usage of the method for class formula
set.seed(9)
outfrm <- smu_cuni_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

summary.ndmodel

**Summary function for class ndmodel**

Description

This method displays a summary containing information about the noise introduction process contained in an object of class ndmodel.

Usage

```r
## S3 method for class 'ndmodel'
summary(object, ..., showid = FALSE)
```

Arguments

- `object`: an object of class ndmodel.
- `...`: other options to pass to the function.
- `showid`: a logical indicating if the indices of noisy samples must be displayed (default: FALSE).

Details

This function presents a summary containing information of the noise introduction process and the resulting noisy dataset contained in the object object of class ndmodel. The information offered is as follows:

- the function call.
- the name of the noise introduction model.
the parameters associated with the noise model.
• the number of noisy and clean samples in the dataset.
• the number of noisy samples per class/attribute.
• the number of clean samples per class/attribute.
• the indices of the noisy samples (if `showid = TRUE`).

Value

A list with the elements of `object`, including the `showid` argument.

See Also

`print.ndmodel`, `plot.ndmodel`, `sym_uni_ln`, `sym_cuni_ln`, `sym_uni_an`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_uni_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
```

---

**symd_gau_an**

Symmetric/dependent Gaussian attribute noise

Description

Introduction of *Symmetric/dependent Gaussian attribute noise* into a classification dataset.

Usage

```r
## Default S3 method:
symd_gau_an(x, y, level, k = 0.2, sortid = TRUE, ...)

## S3 method for class 'formula'
symd_gau_an(formula, data, ...)
```
Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in [0,1] with the noise level to be introduced.
- **k**: a double in [0,1] with the scale used for the standard deviation (default: 0.2).
- **sortid**: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

Details

Symmetric/dependent Gaussian attribute noise corrupts `(level·100)%` of the samples in the dataset. Their attribute values are modified adding a random value that follows a Gaussian distribution of mean $= 0$ and standard deviation $= (\text{max} - \text{min})·k$, being max and min the limits of the attribute domain. For nominal attributes, a random value is chosen.

Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per attribute.
- **idnoise**: an integer vector list with the indices of noisy samples per attribute.
- **numclean**: an integer vector with the amount of clean samples per attribute.
- **idclean**: an integer vector list with the indices of clean samples per attribute.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

(sym_gau_an, sym_int_an, print.ndmodel, summary.ndmodel, plot.ndmodel)
Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- symd_gau_an(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- symd_gau_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

symd_gimg_an

Symmetric/dependent Gaussian-image attribute noise

Description

Introduction of Symmetric/dependent Gaussian-image attribute noise into a classification dataset.

Usage

## Default S3 method:
symd_gimg_an(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
symd_gimg_an(formula, data, ...)

Arguments

x a data frame of input attributes.
y a factor vector with the output class of each sample.
level a double in [0,1] with the noise level to be introduced.
sortid a logical indicating if the indices must be sorted at the output (default: TRUE).
... other options to pass to the function.
formula a formula with the output class and, at least, one input attribute.
data a data frame in which to interpret the variables in the formula.
Details

Symmetric/dependent Gaussian-image attribute noise corrupts \((1 \times 100)\)% of the samples in the dataset. For each sample, a Gaussian distribution (with matching mean and variance to the original sample) is used to generate random attribute values for that sample.

Value

An object of class \(\text{ndmodel}\) with elements:

- \(\text{xnoise}\): a data frame with the noisy input attributes.
- \(\text{ynoise}\): a factor vector with the noisy output class.
- \(\text{numnoise}\): an integer vector with the amount of noisy samples per attribute.
- \(\text{idnoise}\): an integer vector list with the indices of noisy samples per attribute.
- \(\text{numclean}\): an integer vector with the amount of clean samples per attribute.
- \(\text{idclean}\): an integer vector list with the indices of clean samples per attribute.
- \(\text{distr}\): an integer vector with the samples per class in the original data.
- \(\text{model}\): the full name of the noise introduction model used.
- \(\text{param}\): a list of the argument values.
- \(\text{call}\): the function call.

Note

Noise model adapted from the papers in References.

References


See Also

unc_vgau_an, symd_rpix_an, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- symd_gimg_an(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)
```
# usage of the method for class formula
set.seed(9)
outfrm <- symd_gimg_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

**symd_rpix_an**  
*Symmetric/dependent random-pixel attribute noise*

## Description

Introduction of *Symmetric/dependent random-pixel attribute noise* into a classification dataset.

## Usage

```r
## Default S3 method:
symd_rpix_an(x, y, level, sortid = TRUE, ...

## S3 method for class 'formula'
symd_rpix_an(formula, data, ...)
```

## Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `sortid`: a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

## Details

*Symmetric/dependent random-pixel attribute noise* corrupts \((level \cdot 100)\%\) of the samples in the dataset. For each sample, its attribute values are shuffled using independent random permutations.

## Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per attribute.
idnoise an integer vector list with the indices of noisy samples per attribute.

tumclean an integer vector with the amount of clean samples per attribute.

ticlean an integer vector list with the indices of clean samples per attribute.

distr an integer vector with the samples per class in the original data.

model the full name of the noise introduction model used.

param a list of the argument values.

call the function call.

Note

Noise model adapted from the papers in References.

References


See Also

unc_fixw_an, sym_end_an, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- symd_rpix_an(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- symd_rpix_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
symd_uni_an Symmetric/dependent uniform attribute noise

Description

Introduction of Symmetric/dependent uniform attribute noise into a classification dataset.

Usage

## Default S3 method:
symd_uni_an(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
symd_uni_an(formula, data, ...)

Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `sortid`: a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

Details

Symmetric/dependent uniform attribute noise corrupts \((\text{level} \cdot 100)\%\) of the samples in the dataset. Their attribute values are replaced by random different ones between the minimum and maximum of the domain of each attribute following a uniform distribution (for numerical attributes) or choosing a random value (for nominal attributes).

Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per attribute.
- `idnoise`: an integer vector list with the indices of noisy samples per attribute.
- `numclean`: an integer vector with the amount of clean samples per attribute.
- `idclean`: an integer vector list with the indices of clean samples per attribute.
- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.
Note

Noise model adapted from the papers in References.

References


See Also

sym_uni_an, sym_cuni_an, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- symd_uni_an(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- symd_uni_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

sym_adj_ln

Symmetric adjacent label noise

Description

Introduction of Symmetric adjacent label noise into a classification dataset.

Usage

## Default S3 method:
sym_adj_ln(x, y, level, order = levels(y), sortid = TRUE, ...)

## S3 method for class 'formula'
sym_adj_ln(formula, data, ...)
Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in [0,1] with the noise level to be introduced.
- **order**: a character vector indicating the order of the classes (default: `levels(y)`).
- **sortid**: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

Details

*Symmetric adjacent label noise* randomly selects \((\text{level} \cdot 100)\)% of the samples in the dataset with independence of their class. Then, the labels of these samples are replaced by a random adjacent class label according to `order`.

Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

`sym_dran_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`
Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_adj_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)],
                     level = 0.1, order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_adj_ln(formula = Species ~ ., data = iris2D,
                     level = 0.1, order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

------------------------------------------------------

tabular

sym_cen_ln          Symmetric center-based label noise

Description

Introduction of Symmetric center-based label noise into a classification dataset.

Usage

## Default S3 method:
sym_cen_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_cen_ln(formula, data, ...)

Arguments

x       a data frame of input attributes.
y       a factor vector with the output class of each sample.
level   a double in [0,1] with the noise level to be introduced.
sortid  a logical indicating if the indices must be sorted at the output (default: TRUE).
...     other options to pass to the function.
formula a formula with the output class and, at least, one input attribute.
data     a data frame in which to interpret the variables in the formula.
**Details**

_Symmetric center-based label noise_ randomly selects \((\text{level} \cdot 100)\)% of the samples in the dataset with independence of their class. The probability for choosing the noisy label is determined based on the distance between class centers. Thus, the mislabeling probability between classes increases as the distance between their centers decreases. This model is consistent with the intuition that samples in similar classes are more likely to be mislabeled. Besides, the model also allows mislabeling data in dissimilar classes with a relatively small probability, which corresponds to label noise caused by random errors.

**Value**

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.

**Note**

Noise model adapted from the papers in References.

**References**


**See Also**

`glev_uni_ln`, `sym_hienc_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

**Examples**

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_cen_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)
```
```r
# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_cen_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**sym_con_ln**

Symmetric confusion label noise

Description

Introduction of Symmetric confusion label noise into a classification dataset.

Usage

```r
## Default S3 method:
sym_con_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_con_ln(formula, data, ...)
```

Arguments

- **x**  
  a data frame of input attributes.
- **y**  
  a factor vector with the output class of each sample.
- **level**  
  a double in [0,1] with the noise level to be introduced.
- **sortid**  
  a logical indicating if the indices must be sorted at the output (default: TRUE).
- **...**  
  other options to pass to the function.
- **formula**  
  a formula with the output class and, at least, one input attribute.
- **data**  
  a data frame in which to interpret the variables in the formula.

Details

Symmetric confusion label noise considers that the mislabeling probability for each class is level. It obtains the confusion matrix from the dataset, which is row-normalized to estimate the transition matrix and get the probability of selecting each class when noise occurs.
**Value**

An object of class `ndmodel` with elements:

- `xnoise` a data frame with the noisy input attributes.
- `ynoise` a factor vector with the noisy output class.
- `numnoise` an integer vector with the amount of noisy samples per class.
- `idnoise` an integer vector list with the indices of noisy samples.
- `numclean` an integer vector with the amount of clean samples per class.
- `idclean` an integer vector list with the indices of clean samples.
- `distr` an integer vector with the samples per class in the original data.
- `model` the full name of the noise introduction model used.
- `param` a list of the argument values.
- `call` the function call.

**Note**

Noise model adapted from the papers in References, considering C5.0 as classifier.

**References**


**See Also**

`sym_cen_ln`, `glev_uni_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

**Examples**

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_con_ln(x = iris2D[,ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_con_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```
sym_cuni_an  

Symmetric completely-uniform attribute noise

Description

Introduction of *Symmetric completely-uniform attribute noise* into a classification dataset.

Usage

```r
## Default S3 method:
sym_cuni_an(x, y, level, sortid = TRUE, ...)
```

```r
## S3 method for class 'formula'
sym_cuni_an(formula, data, ...)
```

Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

Details

*Symmetric completely-uniform attribute noise* corrupts \((level \cdot 100)\)% of the values of each attribute in the dataset. In order to corrupt an attribute \(A\), \((level \cdot 100)\)% of the samples in the dataset are randomly chosen. Then, their values for \(A\) are replaced by random ones from the domain of the attribute. Note that the original attribute value of a sample can be chosen as noisy and the actual percentage of noise in the dataset can be lower than the theoretical noise level.

Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per attribute.
- `idnoise`: an integer vector list with the indices of noisy samples per attribute.
- `numclean`: an integer vector with the amount of clean samples per attribute.
- `idclean`: an integer vector list with the indices of clean samples per attribute.
- `distr`: an integer vector with the samples per class in the original data.
model the full name of the noise introduction model used.
param a list of the argument values.
call the function call.

Note

Noise model adapted from the papers in References, only considering attribute noise introduction.

References


See Also

sym_uni_an, sym_cuni_cn, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_cuni_an(x = iris2D[,ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_cuni_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**sym_cuni_cn**  
Symmetric completely-uniform combined noise

Description

Introduction of *Symmetric completely-uniform combined noise* into a classification dataset.
Usage

## Default S3 method:
sym_cuni_cn(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_cuni_cn(formula, data, ...)

Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in [0,1] with the noise level to be introduced.
- **sortid**: a logical indicating if the indices must be sorted at the output (default: TRUE).
- **...**: other options to pass to the function.
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

Details

**Symmetric completely-uniform combined noise** corrupts \( (\text{level} \cdot 100)\% \) of the values of each attribute in the dataset. In order to corrupt an attribute \( A \), \( (\text{level} \cdot 100)\% \) of the samples in the dataset are randomly chosen. Then, their values for \( A \) are replaced by random ones from the domain of the attribute.

Additionally, this noise model also selects \( (\text{level} \cdot 100)\% \) of the samples in the dataset with independence of their class. The labels of these samples are randomly replaced by other ones within the set of class labels.

Note that, for both attributes and class labels, the original value of a sample can be chosen as noisy and the actual percentage of noise in the dataset can be lower than the theoretical noise level.

Value

An object of class ndmodel with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per variable.
- **idnoise**: an integer vector list with the indices of noisy samples per variable.
- **numclean**: an integer vector with the amount of clean samples per variable.
- **idclean**: an integer vector list with the indices of clean samples per variable.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.
Note

Noise model adapted from the papers in References.

References


See Also

`uncs_guni_cn`, `sym_cuni_an`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_cuni_cn(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_cuni_cn(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

sym_cuni_ln

**Symmetric completely-uniform label noise**

Description

Introduction of *Symmetric completely-uniform label noise* into a classification dataset.

Usage

```r
## Default S3 method:
sym_cuni_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_cuni_ln(formula, data, ...)
```
**Arguments**

- `x` a data frame of input attributes.
- `y` a factor vector with the output class of each sample.
- `level` a double in [0,1] with the noise level to be introduced.
- `sortid` a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...` other options to pass to the function.
- `formula` a formula with the output class and, at least, one input attribute.
- `data` a data frame in which to interpret the variables in the formula.

**Details**

*Symmetric completely-uniform label noise* randomly selects \((level \cdot 100)\)% of the samples in the dataset with independence of their class. Then, the labels of these samples are randomly replaced by others within the set of class labels. Note that this model can choose the original label of a sample as noisy.

**Value**

An object of class `ndmodel` with elements:

- `xnoise` a data frame with the noisy input attributes.
- `ynoise` a factor vector with the noisy output class.
- `numnoise` an integer vector with the amount of noisy samples per class.
- `idnoise` an integer vector list with the indices of noisy samples.
- `numclean` an integer vector with the amount of clean samples per class.
- `idclean` an integer vector list with the indices of clean samples.
- `distr` an integer vector with the samples per class in the original data.
- `model` the full name of the noise introduction model used.
- `param` a list of the argument values.
- `call` the function call.

**Note**

Noise model adapted from the papers in References.

**References**


**See Also**

`symUniLn`, `symCuniAn`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`
sym_ddef_ln

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_cuni_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_cuni_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

sym_ddef_ln  Symmetric double-default label noise

Description

Introduction of *Symmetric double-default label noise* into a classification dataset.

Usage

```r
# Default S3 method:
sym_ddef_ln(
  x,
  y,
  level,
  def1 = 1,
  def2 = 2,
  order = levels(y),
  sortid = TRUE,
  ...
)
```

```r
# S3 method for class 'formula'
sym_ddef_ln(formula, data, ...)
```

Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
level  a double in [0,1] with the noise level to be introduced.
def1  an integer with the index of the first default class (default: 1).
def2  an integer with the index of the second default class (default: 2).
order a character vector indicating the order of the classes (default: levels(y)).
sortid a logical indicating if the indices must be sorted at the output (default: TRUE).
... other options to pass to the function.
formula a formula with the output class and, at least, one input attribute.
data a data frame in which to interpret the variables in the formula.

Details

Symmetric double-default label noise randomly selects (level·100)% of the samples in the dataset with independence of their class. Then, the labels of these samples are replaced by one of two fixed labels (def1 or def2) within the set of class labels. The indices def1 and def2 are taken according to the order given by order.

Value

An object of class ndmodel with elements:
xnoise   a data frame with the noisy input attributes.
ynoise   a factor vector with the noisy output class.
umnoise  an integer vector with the amount of noisy samples per class.
idnoise  an integer vector list with the indices of noisy samples.
umclean an integer vector with the amount of clean samples per class.
idclean an integer vector list with the indices of clean samples.
distr    an integer vector with the samples per class in the original data.
model    the full name of the noise introduction model used.
param    a list of the argument values.
call     the function call.

Note

Noise model adapted from the papers in References.

References


See Also

sym_exc_ln, sym_cuni_ln, print.ndmodel, summary.ndmodel, plot.ndmodel
Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_ddef_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)],
                      level = 0.1, order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_ddef_ln(formula = Species ~ ., data = iris2D,
                      level = 0.1, order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

sym_def_ln  
Symmetric default label noise

Description

Introduction of *Symmetric default label noise* into a classification dataset.

Usage

```r
## Default S3 method:
sym_def_ln(x, y, level, def = 1, order = levels(y), sortid = TRUE, ...)

## S3 method for class 'formula'
sym_def_ln(formula, data, ...)
```

Arguments

- **x** a data frame of input attributes.
- **y** a factor vector with the output class of each sample.
- **level** a double in [0,1] with the noise level to be introduced.
- **def** an integer with the index of the default class (default: 1).
- **order** a character vector indicating the order of the classes (default: levels(y)).
- **sortid** a logical indicating if the indices must be sorted at the output (default: TRUE).
- **formula** a formula with the output class and, at least, one input attribute.
- **data** a data frame in which to interpret the variables in the formula.
Details

Symmetric default label noise randomly selects \((\text{level} \cdot 100)\)% of the samples in the dataset with independence of their class. Then, the labels of these samples are replaced by a fixed label \((\text{def})\) within the set of class labels. The index \(\text{def}\) is taken according to the order given by \(\text{order}\).

Value

An object of class \text{ndmodel} with elements:

- \text{xnoise} a data frame with the noisy input attributes.
- \text{ynoise} a factor vector with the noisy output class.
- \text{numnoise} an integer vector with the amount of noisy samples per class.
- \text{idnoise} an integer vector list with the indices of noisy samples.
- \text{numclean} an integer vector with the amount of clean samples per class.
- \text{idclean} an integer vector list with the indices of clean samples.
- \text{distr} an integer vector with the samples per class in the original data.
- \text{model} the full name of the noise introduction model used.
- \text{param} a list of the argument values.
- \text{call} the function call.

Note

Noise model adapted from the papers in References.

References


See Also

- \text{sym_def_ln}, \text{sym_exc_ln}, \text{print.ndmodel}, \text{summary.ndmodel}, \text{plot.ndmodel}

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_def_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)],
                     level = 0.1, order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)
```
# usage of the method for class formula
set.seed(9)
outfrm <- sym_def_ln(formula = Species ~ ., data = iris2D,
                      level = 0.1, order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

**sym_dia_ln**

*Symmetric diametrical label noise*

**Description**

Introduction of *Symmetric diametrical label noise* into a classification dataset.

**Usage**

```r
## Default S3 method:
sym_dia_ln(x, y, level, order = levels(y), sortid = TRUE, ...)

## S3 method for class 'formula'
sym_dia_ln(formula, data, ...)
```

**Arguments**

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `order`: a character vector indicating the order of the classes (default: `levels(y)`).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

**Details**

*Symmetric diametrical label noise* randomly selects (level·100)% of the samples in the dataset with independence of their class. In this model, diametrical (opposite) classes are more likely to have their labels mixed. The probability of mislabel a sample of class `i` as belonging to class `j` is computed as \(d_{ij}/S\), where \(d_{ij} = |i-j|\) and \(S\) is the sum of distances to class `i`. The order of the classes is determined by `order`. 
Value

An object of class ndmodel with elements:

- xnoise: a data frame with the noisy input attributes.
- ynoise: a factor vector with the noisy output class.
- numnoise: an integer vector with the amount of noisy samples per class.
- idnoise: an integer vector list with the indices of noisy samples.
- numclean: an integer vector with the amount of clean samples per class.
- idclean: an integer vector list with the indices of clean samples.
- distr: an integer vector with the samples per class in the original data.
- model: the full name of the noise introduction model used.
- param: a list of the argument values.
- call: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

sym_pes_ln, sym_opt_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_dia_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)],
                      level = 0.1, order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_dia_ln(formula = Species ~ ., data = iris2D,
                      level = 0.1, order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
sym_dran_ln

identical(outdef$idnoise, outfrm$idnoise)

---

**Symmetric double-random label noise**

Description

Introduction of *Symmetric double-random label noise* into a classification dataset.

Usage

```r
## Default S3 method:
sym_dran_ln(x, y, level, sortid = TRUE, ...)
```

```r
## S3 method for class 'formula'
sym_dran_ln(formula, data, ...)
```

Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in [0,1] with the noise level to be introduced.
- **sortid**: a logical indicating if the indices must be sorted at the output (default: TRUE).
- **...**: other options to pass to the function.
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

Details

*Symmetric double-random label noise* randomly selects ($level\cdot100$)% of the samples in the dataset with independence of their class. Then, each of the original class labels is flipped to one between two other random labels.

Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
distr  an integer vector with the samples per class in the original data.
model  the full name of the noise introduction model used.
param  a list of the argument values.
call   the function call.

Note

Noise model adapted from the papers in References.

References


See Also

sym_hie_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_dran_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_dran_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
Usage

```r
## Default S3 method:
sym_end_an(x, y, level, scale = 0.2, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_end_an(formula, data, ...)
```

Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `scale`: a double in (0,1) with the scale to be used (default: 0.2).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

Details

For each attribute $A$, Symmetric end-directed attribute noise computes a value $k = \text{scale} \cdot \text{max}(A)$. Then, it chooses $(\text{level}-100)\%$ of the values of that attribute. For each value, it applies the following procedure:

- If the value is less than the median of the attribute, the value transforms into adding $k$ to the maximum of the attribute $A$.
- If the value is greater than the median of the attribute, the value transforms into subtracting $k$ from the minimum of the attribute $A$.
- If the value matches the median, one of the two previous alternatives is chosen.

Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per attribute.
- `idnoise`: an integer vector list with the indices of noisy samples per attribute.
- `numclean`: an integer vector with the amount of clean samples per attribute.
- `idclean`: an integer vector list with the indices of clean samples per attribute.
- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.
**Note**

Noise model adapted from the papers in References.

**References**


**See Also**

sym_sgau_an, symd_gau_an, print.ndmodel, summary.ndmodel, plot.ndmodel

**Examples**

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_end_an(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_end_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**sym_exc_ln**

*Symmetric exchange label noise*

**Description**

Introduction of *Symmetric exchange label noise* into a classification dataset.

**Usage**

```r
## Default S3 method:
sym_exc_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_exc_ln(formula, data, ...)
```
Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in [0,1] with the noise level to be introduced.
- **sortid**: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

Details

Symmetric exchange label noise randomly selects `(level\cdot 100)\%` of the samples in the dataset with independence of their class. These samples are divided into two groups: `A` and `B`. Then, each sample of group `A` is labeled with the label of a sample of group `B` and vice versa.

Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

- `sym_cuni_ln`, `sym_cuni_an`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`
Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_exc_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_exc_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

sym_gau_an

Symmetric Gaussian attribute noise

Description

Introduction of Symmetric Gaussian attribute noise into a classification dataset.

Usage

## Default S3 method:
sym_gau_an(x, y, level, k = 0.2, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_gau_an(formula, data, ...)

Arguments

x a data frame of input attributes.
y a factor vector with the output class of each sample.
level a double in [0,1] with the noise level to be introduced.
k a double in [0,1] with the scale used for the standard deviation (default: 0.2).
sortid a logical indicating if the indices must be sorted at the output (default: TRUE).
... other options to pass to the function.
formula a formula with the output class and, at least, one input attribute.
data a data frame in which to interpret the variables in the formula.
Symmetric Gaussian attribute noise corrupts \((\text{level} \cdot 100)\)% of the values of each attribute in the dataset. In order to corrupt an attribute \(A\), \((\text{level} \cdot 100)\)% of the samples in the dataset are chosen. Then, their values for \(A\) are corrupted adding a random value that follows a Gaussian distribution of \(\text{mean} = 0\) and \(\text{standard deviation} = (\text{max-min}) \cdot k\), being \(\text{max}\) and \(\text{min}\) the limits of the attribute domain. For nominal attributes, a random value is chosen.

### Value

An object of class \(\text{ndmodel}\) with elements:

- \(\text{xnoise}\) a data frame with the noisy input attributes.
- \(\text{ynoise}\) a factor vector with the noisy output class.
- \(\text{numnoise}\) an integer vector with the amount of noisy samples per attribute.
- \(\text{idnoise}\) an integer vector list with the indices of noisy samples per attribute.
- \(\text{numclean}\) an integer vector with the amount of clean samples per attribute.
- \(\text{idclean}\) an integer vector list with the indices of clean samples per attribute.
- \(\text{distr}\) an integer vector with the samples per class in the original data.
- \(\text{model}\) the full name of the noise introduction model used.
- \(\text{param}\) a list of the argument values.
- \(\text{call}\) the function call.

### Note

Noise model adapted from the papers in References.

### References


### See Also

- \(\text{sym_int_an}\), \(\text{symd_uni_an}\), \(\text{print.ndmodel}\), \(\text{summary.ndmodel}\), \(\text{plot.ndmodel}\)

### Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_gau_an(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
```
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_gau_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

**sym_hienc_ln**

*Symmetric hierarchical/next-class label noise*

**Description**

Introduction of *Symmetric hierarchical/next-class label noise* into a classification dataset.

**Usage**

```r
## Default S3 method:
sym_hiencLn(x, y, level, group, order = levels(y), sortid = TRUE, ...)
## S3 method for class 'formula'
sym_hiencLn(formula, data, ...)
```

**Arguments**

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `group`: a list of integer vectors with the indices of classes in each superclass.
- `order`: a character vector indicating the order of the classes (default: `levels(y)`).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

**Details**

*Symmetric hierarchical/next-class label noise* randomly selects (level·100)% of the samples in the dataset with independence of their class. Then, the labels of these samples are replaced by the next class within the set of class labels related to them (given by the argument `group`). The indices in `group` are taken according to the order given by `order`. Note that if a class does not belong to any superclass, it may be mislabeled as any other class.
**Value**

An object of class `ndmodel` with elements:

- `xnoise` a data frame with the noisy input attributes.
- `ynoise` a factor vector with the noisy output class.
- `numnoise` an integer vector with the amount of noisy samples per class.
- `idnoise` an integer vector list with the indices of noisy samples.
- `numclean` an integer vector with the amount of clean samples per class.
- `idclean` an integer vector list with the indices of clean samples.
- `distr` an integer vector with the samples per class in the original data.
- `model` the full name of the noise introduction model used.
- `param` a list of the argument values.
- `call` the function call.

**Note**

Noise model adapted from the papers in References.

**References**


**See Also**

- `sym_nexc_ln`, `sym_dia_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

**Examples**

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_hienc_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1,
                       group = list(c(1,2)), order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_hienc_ln(formula = Species ~ ., data = iris2D, level = 0.1,
                       group = list(c(1,2)), order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
```

Introduction of Symmetric hierarchical label noise into a classification dataset.

Usage

```r
# Default S3 method:
sym_hie_ln(x, y, level, group, order = levels(y), sortid = TRUE, ...)

# S3 method for class 'formula'
sym_hie_ln(formula, data, ...)
```

Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `group`: a list of integer vectors with the indices of classes in each superclass.
- `order`: a character vector indicating the order of the classes (default: `levels(y)`).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

Details

Symmetric hierarchical label noise randomly selects (level\cdot100)% of the samples in the dataset with independence of their class. Then, the labels of these samples are randomly replaced by other ones within the set of class labels related to them (given by the argument `group`). The indices in `group` are taken according to the order given by `order`. Note that if a class does not belong to any superclass, it may be mislabeled as any other class.

Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
`sym_hie_ln`  

- `idnoise`: an integer vector list with the indices of noisy samples.  
- `numclean`: an integer vector with the amount of clean samples per class.  
- `idclean`: an integer vector list with the indices of clean samples.  
- `distr`: an integer vector with the samples per class in the original data.  
- `model`: the full name of the noise introduction model used.  
- `param`: a list of the argument values.  
- `call`: the function call.  

**Note**  

Noise model adapted from the papers in References.

**References**  


**See Also**  

`sym_uni_ln`, `sym_def_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

**Examples**

```R  
# load the dataset  
data(iris2D)  

# usage of the default method: a superclass with labels of indices 1 and 2  
set.seed(9)  
outdef <- sym_hie_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1,  
group = list(c(1,2)), order = c("virginica", "setosa", "versicolor"))  

# show results  
summary(outdef, showid = TRUE)  
plot(outdef)  

# usage of the method for class formula  
set.seed(9)  
outfrm <- sym_hie_ln(formula = Species ~ ., data = iris2D, level = 0.1,  
group = list(c(1,2)), order = c("virginica", "setosa", "versicolor"))  

# check the match of noisy indices  
identical(outdef$idnoise, outfrm$idnoise)  
```
Symmetric interval-based attribute noise

Description

Introduction of Symmetric interval-based attribute noise into a classification dataset.

Usage

```r
# Default S3 method:
sym_int_an(x, y, level, nbins = 10, sortid = TRUE, ...)
```

```r
# S3 method for class 'formula'
sym_int_an(formula, data, ...)
```

Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `nbins`: an integer with the number of bins to create (default: 10).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

Details

Symmetric interval-based attribute noise corrupts (level·100)% of the values of each attribute in the dataset. In order to corrupt an attribute $A$, (level·100)% of the samples in the dataset are selected. To corrupt numeric attributes, the attribute is split into `nbins` equal-frequency intervals, one of its closest intervals is chosen and a random value within the interval is picked out as noisy. For nominal attributes, a random value within the domain is chosen.

Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per attribute.
- `idnoise`: an integer vector list with the indices of noisy samples per attribute.
- `numclean`: an integer vector with the amount of clean samples per attribute.
- `idclean`: an integer vector list with the indices of clean samples per attribute.
**Symmetric natural-distribution label noise**

**Description**

Introduction of *Symmetric natural-distribution label noise* into a classification dataset.

**distr**  
an integer vector with the samples per class in the original data.

**model**  
the full name of the noise introduction model used.

**param**  
a list of the argument values.

**call**  
the function call.

**Note**

Noise model adapted from the papers in References.

**References**


**See Also**

*symd_uni_an*, *sym_uni_an*, *print.ndmodel*, *summary.ndmodel*, *plot.ndmodel*

**Examples**

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_int_an(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_int_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```
Usage

## Default S3 method:
sym_natd_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_natd_ln(formula, data, ...)

Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in [0,1] with the noise level to be introduced.
- **sortid**: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- **...**: other options to pass to the function.
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

Details

`Symmetric natural-distribution label noise` randomly selects (level\cdot100)% of the samples in the dataset with independence of their class. Then, the labels of these samples are randomly replaced by other different ones within the set of class labels. When noise for a certain class occurs, another class with a probability proportional to the natural class distribution replaces it.

Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

Note

Noise model adapted from the papers in References.
References


See Also

sym_nuni ln, sym_adj ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_natd ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_natd ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

sym_nean ln

Symmetric nearest-neighbor label noise

Description

Introduction of Symmetric nearest-neighbor label noise into a classification dataset.

Usage

## Default S3 method:
sym_nean ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_nean ln(formula, data, ...)

Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

Details

*Symmetric nearest-neighbor label noise* randomly selects `(level-100)%` of the samples in the dataset with independence of their class. Then, the labels of these samples are replaced by the label of the nearest sample of a different class.

Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

`sym_con_ln`, `sym_cen_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`
sym_nexc_ln

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_nexc_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_nexc_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

sym_nexc_ln  Symmetric next-class label noise

Description

Introduction of Symmetric next-class label noise into a classification dataset.

Usage

## Default S3 method:
sym_nexc_ln(x, y, level, order = levels(y), sortid = TRUE, ...)

## S3 method for class 'formula'
sym_nexc_ln(formula, data, ...)

Arguments

x  a data frame of input attributes.
y  a factor vector with the output class of each sample.
level  a double in [0,1] with the noise level to be introduced.
order  a character vector indicating the order of the classes (default: levels(y)).
sortid  a logical indicating if the indices must be sorted at the output (default: TRUE).
...  other options to pass to the function.
formula  a formula with the output class and, at least, one input attribute.
data  a data frame in which to interpret the variables in the formula.
Details

The Symmetric next-class label noise introduction model randomly selects (level·100)% of the samples in the dataset with independence of their class. Then, the labels of these samples are replaced by the next class label according to order.

Value

An object of class ndmodel with elements:

- xnoise: a data frame with the noisy input attributes.
- ynoise: a factor vector with the noisy output class.
- numnoise: an integer vector with the amount of noisy samples per class.
- idnoise: an integer vector list with the indices of noisy samples.
- numclean: an integer vector with the amount of clean samples per class.
- idclean: an integer vector list with the indices of clean samples.
- distr: an integer vector with the samples per class in the original data.
- model: the full name of the noise introduction model used.
- param: a list of the argument values.
- call: the function call.

Note

Noise model adapted from the papers in References

References


See Also

sym_dia_ln, sym_pes_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_nexc_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1, order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)
# usage of the method for class formula
set.seed(9)
outfrm <- sym_nuni_ln(formula = Species ~ ., data = iris2D,
  level = 0.1, order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

description

Symmetric non-uniform label noise

Introduction of Symmetric non-uniform label noise into a classification dataset.

Usage

## Default S3 method:
sym_nuni_ln(x, y, level, tramat, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_nuni_ln(formula, data, ...)

Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in [0,1] with the noise level to be introduced.
- **tramat**: a double matrix with the values of the transition matrix.
- **sortid**: a logical indicating if the indices must be sorted at the output (default: TRUE).
- **...**: other options to pass to the function.
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

Details

Symmetric non-uniform label noise randomly selects (level·100)% of the samples in the dataset with independence of their class. Then, the labels of these samples are randomly replaced by other different ones according to the probabilities given in the transition matrix tramat. For details about the structure of the transition matrix, see Kang et al. (2021).
Value

An object of class `ndmodel` with elements:

- `xnoise` a data frame with the noisy input attributes.
- `ynoise` a factor vector with the noisy output class.
- `numnoise` an integer vector with the amount of noisy samples per class.
- `idnoise` an integer vector list with the indices of noisy samples.
- `numclean` an integer vector with the amount of clean samples per class.
- `idclean` an integer vector list with the indices of clean samples.
- `distr` an integer vector with the samples per class in the original data.
- `model` the full name of the noise introduction model used.
- `param` a list of the argument values.
- `call` the function call.

Note

Noise model adapted from the papers in References.

References


See Also

- `sym_adj_ln`, `sym_dran_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
tramat <- matrix(data = c(0.9, 0.03, 0.07, 0.03, 0.9, 0.07, 0.03, 0.07, 0.9),
nrow = 3, ncol = 3, byrow = TRUE)
outdef <- sym_nuni_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)],
                      level = 0.1, tramat = tramat)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_nuni_ln(formula = Species ~ ., data = iris2D, level = 0.1, tramat = tramat)
```
# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

**sym_opt_ln**

**Symmetric optimistic label noise**

**Description**

Introduction of *Symmetric optimistic label noise* into a classification dataset.

**Usage**

```r
## Default S3 method:
sym_opt_ln(x, y, level, levelH = 0.9, order = levels(y), sortid = TRUE, ...)
```

```r
## S3 method for class 'formula'
sym_opt_ln(formula, data, ...)
```

**Arguments**

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in [0,1] with the noise level to be introduced.
- `levelH`: a double in (0.5, 1] with the noise level for higher classes (default: 0.9).
- `order`: a character vector indicating the order of the classes (default: `levels(y)`).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

**Details**

*Symmetric optimistic label noise* randomly selects $(\text{level} \cdot 100\%)$ of the samples in the dataset with independence of their class. In the optimistic case, the probability of a class $i$ of being mislabeled as class $j$ is higher for $j > i$ in comparison to $j < i$. Thus, when noise for a certain class occurs, it is assigned to a random higher class with probability $\text{levelH}$ and to a random lower class with probability $1-\text{levelH}$. The order of the classes is determined by `order`.
### Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.

### Note

Noise model adapted from the papers in References.

### References


### See Also

`sym_usim_ln`, `sym_natd_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

### Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_opt_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)],
                      level = 0.1, order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_opt_ln(formula = Species ~ ., data = iris2D,
                      level = 0.1, order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
```
sym_pes_ln

identical(outdef$idnoise, outfrm$idnoise)

---

**Symmetric pessimistic label noise**

### Description

Introduction of *Symmetric pessimistic label noise* into a classification dataset.

### Usage

```r
## Default S3 method:
sym_pes_ln(x, y, level, levelL = 0.9, order = levels(y), sortid = TRUE, ...)

## S3 method for class 'formula'
sym_pes_ln(formula, data, ...)
```

### Arguments

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in \([0, 1]\) with the noise level to be introduced.
- `levelL`: a double in \((0.5, 1]\) with the noise level for lower classes (default: 0.9).
- `order`: a character vector indicating the order of the classes (default: `levels(y)`).
- `sortid`: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

### Details

*Symmetric pessimistic label noise* randomly selects \((\text{level} \cdot 100)\)% of the samples in the dataset with independence of their class. In the pessimistic case, the probability of a class \(i\) of being mislabeled as class \(j\) is higher for \(j < i\) in comparison to \(j > i\). Thus, when noise for a certain class occurs, it is assigned to a random lower class with probability \(\text{levelL}\) and to a random higher class with probability \(1 - \text{levelL}\). The order of the classes is determined by `order`.

### Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
idnoise       an integer vector list with the indices of noisy samples.
numclean      an integer vector with the amount of clean samples per class.
idclean       an integer vector list with the indices of clean samples.
distr         an integer vector with the samples per class in the original data.
model         the full name of the noise introduction model used.
param         a list of the argument values.
call          the function call.

Note

Noise model adapted from the papers in References.

References

R. C. Prati, J. Luengo, and F. Herrera. Emerging topics and challenges of learning from noisy

See Also

sym_opt_ln, sym_usim_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_pes_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)],
                      level = 0.1, order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_pes_ln(formula = Species ~ ., data = iris2D,
                      level = 0.1, order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
Symmetric scaled-Gaussian attribute noise

Description

Introduction of Symmetric scaled-Gaussian attribute noise into a classification dataset.

Usage

## Default S3 method:
sym_sgau_an(x, y, level, k = 0.2, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_sgau_an(formula, data, ...)

Arguments

x a data frame of input attributes.
y a factor vector with the output class of each sample.
level a double in [0,1] with the noise level to be introduced.
k a double in [0,1] with the scale used for the standard deviation (default: 0.2).
sortid a logical indicating if the indices must be sorted at the output (default: TRUE).
... other options to pass to the function.
formula a formula with the output class and, at least, one input attribute.
data a data frame in which to interpret the variables in the formula.

Details

Symmetric scaled-Gaussian attribute noise corrupts (level·100)% of the values of each attribute in the dataset. In order to corrupt an attribute $A$, (level·100)% of the samples in the dataset are chosen. Then, their values for $A$ are modified adding a random value that follows a Gaussian distribution of mean = 0 and standard deviation = (max-min)·k·level, being max and min the limits of the attribute domain. For nominal attributes, a random value is chosen.

Value

An object of class ndmodel with elements:
xnoise a data frame with the noisy input attributes.
ynoise a factor vector with the noisy output class.
numnoise an integer vector with the amount of noisy samples per attribute.
idnoise an integer vector list with the indices of noisy samples per attribute.
numclean an integer vector with the amount of clean samples per attribute.
idclean an integer vector list with the indices of clean samples per attribute.
sym_uni_an

<table>
<thead>
<tr>
<th>distr</th>
<th>an integer vector with the samples per class in the original data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>the full name of the noise introduction model used.</td>
</tr>
<tr>
<td>param</td>
<td>a list of the argument values.</td>
</tr>
<tr>
<td>call</td>
<td>the function call.</td>
</tr>
</tbody>
</table>

**Note**

Noise model adapted from the papers in References.

**References**


**See Also**

`sym_sgau_an`, `sym_gau_an`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

**Examples**

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_sgau_an(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_sgau_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**sym_uni_an**

**Symmetric uniform attribute noise**

**Description**

Introduction of *Symmetric uniform attribute noise* into a classification dataset.
sym_uni_an

Usage

## Default S3 method:
sym_uni_an(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_uni_an(formula, data, ...)

Arguments

x a data frame of input attributes.
y a factor vector with the output class of each sample.
level a double in [0,1] with the noise level to be introduced.
sortid a logical indicating if the indices must be sorted at the output (default: TRUE).
... other options to pass to the function.
formula a formula with the output class and, at least, one input attribute.
data a data frame in which to interpret the variables in the formula.

Details

Symmetric uniform attribute noise corrupts (level·100)% of the values of each attribute in the dataset. In order to corrupt an attribute $A$, $(\text{level} \cdot 100)%$ of the samples in the dataset are randomly chosen. Then, their values for $A$ are replaced by random different ones from the domain of the attribute.

Value

An object of class ndmodel with elements:

xnoise a data frame with the noisy input attributes.
ynoise a factor vector with the noisy output class.
numnoise an integer vector with the amount of noisy samples per attribute.
idnoise an integer vector list with the indices of noisy samples per attribute.
numclean an integer vector with the amount of clean samples per attribute.
idclean an integer vector list with the indices of clean samples per attribute.
distr an integer vector with the samples per class in the original data.
model the full name of the noise introduction model used.
param a list of the argument values.
call the function call.

Note

Noise model adapted from the papers in References.
References


See Also

`sym_cuni_an`, `sym_cuni_cn`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_uni_an(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_uni_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**sym_uni_ln**

*Symmetric uniform label noise*

Description

Introduction of *Symmetric uniform label noise* into a classification dataset.

Usage

```r
## Default S3 method:
sym_uni_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_uni_ln(formula, data, ...)
```
Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in \([0,1]\) with the noise level to be introduced.
- **sortid**: a logical indicating if the indices must be sorted at the output (default: `TRUE`).
- **...**: other options to pass to the function.
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.

Details

*Symmetric uniform label noise* randomly selects \((\text{level}\cdot100)\%\) of the samples in the dataset with independence of their class. Then, the labels of these samples are randomly replaced by other different ones within the set of class labels.

Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per class.
- **idnoise**: an integer vector list with the indices of noisy samples.
- **numclean**: an integer vector with the amount of clean samples per class.
- **idclean**: an integer vector list with the indices of clean samples.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

`sym_def_ln`, `sym_ddef_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`
Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_uni_ln(x = iris2D[,ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- sym_uni_ln(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**sym_uni_ln**  
*Symmetric unit-simplex label noise*

Description

Introduction of *Symmetric unit-simplex label noise* into a classification dataset.

Usage

```
## Default S3 method:
sym_uni_ln(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
sym_uni_ln(formula, data, ...)
```

Arguments

- **x** a data frame of input attributes.
- **y** a factor vector with the output class of each sample.
- **level** a double in [0,1] with the noise level to be introduced.
- **sortid** a logical indicating if the indices must be sorted at the output (default: TRUE).
- **...** other options to pass to the function.
- **formula** a formula with the output class and, at least, one input attribute.
- **data** a data frame in which to interpret the variables in the formula.
Details

*Symmetric unit-simplex label noise* randomly selects ($\text{level} \cdot 100\%$) of the samples in the dataset with independence of their class. Then, the labels of these samples are randomly replaced by other different ones within the set of class labels. The probability for each noisy class is drawn uniformly and independently from the $M-1$-dimensional unit simplex (with $M$ the number of classes).

Value

An object of class `ndmodel` with elements:

- `xnoise` a data frame with the noisy input attributes.
- `ynoise` a factor vector with the noisy output class.
- `numnoise` an integer vector with the amount of noisy samples per class.
- `idnoise` an integer vector list with the indices of noisy samples.
- `numclean` an integer vector with the amount of clean samples per class.
- `idclean` an integer vector list with the indices of clean samples.
- `distr` an integer vector with the samples per class in the original data.
- `model` the full name of the noise introduction model used.
- `param` a list of the argument values.
- `call` the function call.

Note

Noise model adapted from the papers in References.

References


See Also

`sym_natd_ln`, `sym_nuni_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- sym_usim_ln(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
```
ugau_bor_ln

Uneven-Gaussian borderline label noise

Description

Introduction of Uneven-Gaussian borderline label noise into a classification dataset.

Usage

```r
## Default S3 method:
ugau_bor_ln(
x, y, level, mean = 0, sd = 1, k = 1, order = levels(y), sortid = TRUE, ...
)
## S3 method for class 'formula'
ugau_bor_ln(formula, data, ...)
```

Arguments

- `x` a data frame of input attributes.
- `y` a factor vector with the output class of each sample.
- `level` a double vector with the noise levels in [0,1] to be introduced into each class.
- `mean` a double with the mean for the Gaussian distribution (default: 0).
- `sd` a double with the standard deviation for the Gaussian distribution (default: 1).
- `k` an integer with the number of nearest neighbors to be used (default: 1).
- `order` a character vector indicating the order of the classes (default: `levels(y)`).
- `sortid` a logical indicating if the indices must be sorted at the output (default: TRUE).
- `formula` a formula with the output class and, at least, one input attribute.
- `data` a data frame in which to interpret the variables in the formula.
Details

*Uneven-Gaussian borderline label noise* uses an SVM to induce the decision border in the dataset. For each sample, its distance to the decision border is computed. Then, a Gaussian distribution with parameters (mean, sd) is used to compute the value for the probability density function associated to each distance. For each class \( c[i] \), it randomly selects \((level[i]-100)\%\) of the samples in the dataset based on their values of the probability density function -the order of the class labels is determined by order. For each noisy sample, the majority class among its \( k \)-nearest neighbors of a different class is chosen as the new label.

Value

An object of class `ndmodel` with elements:

- `xnoise`: a data frame with the noisy input attributes.
- `ynoise`: a factor vector with the noisy output class.
- `numnoise`: an integer vector with the amount of noisy samples per class.
- `idnoise`: an integer vector list with the indices of noisy samples.
- `numclean`: an integer vector with the amount of clean samples per class.
- `idclean`: an integer vector list with the indices of clean samples.
- `distr`: an integer vector with the samples per class in the original data.
- `model`: the full name of the noise introduction model used.
- `param`: a list of the argument values.
- `call`: the function call.

Note

Noise model adapted from the papers in References to multiclass data, considering SVM with linear kernel as classifier, a mislabeling process using the neighborhood of noisy samples and a noise level to control the number of errors in the data.

References


See Also

- `gaum_bor_ln`, `gau_bor_ln`, `print.ndmodel`, `summary.ndmodel`, `plot.ndmodel`

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- ugap_bor_ln(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)],
level = c(0.1, 0.2, 0.3), order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- ugap_bor_ln(formula = Species ~ ., data = iris2D,
  level = c(0.1, 0.2, 0.3), order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

ulap_bor_ln  Uneven-Laplace borderline noise

Description

Introduction of Uneven-Laplace borderline noise into a classification dataset.

Usage

## Default S3 method:
ulap_bor_ln(
  x,
  y,
  level,
  mu = 0,
  b = 1,
  k = 1,
  order = levels(y),
  sortid = TRUE,
  ...
)

## S3 method for class 'formula'
ulap_bor_ln(formula, data, ...)

Arguments

  x  a data frame of input attributes.
  y  a factor vector with the output class of each sample.
  level  a double vector with the noise levels in [0,1] to be introduced into each class.
  mu  a double with the location for the Laplace distribution (default: 0).
uneven-Laplace borderline noise uses an SVM to induce the decision border in the dataset. For each sample, its distance to the decision border is computed. Then, a Laplace distribution with parameters \((\mu, b)\) is used to compute the value for the probability density function associated to each distance. For each class \(c[i]\), it randomly selects \((\text{levels}[i] \cdot 100)\%\) of the samples in the dataset based on their values of the probability density function -the order of the class labels is determined by \(\text{order}\). For each noisy sample, the majority class among its \(k\)-nearest neighbors of a different class is chosen as the new label.

Value
An object of class \text{ndmodel} with elements:

- \text{xnoise} a data frame with the noisy input attributes.
- \text{ynoise} a factor vector with the noisy output class.
- \text{numnoise} an integer vector with the amount of noisy samples per class.
- \text{idnoise} an integer vector list with the indices of noisy samples.
- \text{numclean} an integer vector with the amount of clean samples per class.
- \text{idclean} an integer vector list with the indices of clean samples.
- \text{distr} an integer vector with the samples per class in the original data.
- \text{model} the full name of the noise introduction model used.
- \text{param} a list of the argument values.
- \text{call} the function call.

Note
Noise model adapted from the papers in References to multiclass data, considering SVM with linear kernel as classifier, a mislabeling process using the neighborhood of noisy samples and a noise level to control the number of errors in the data.

References
See Also

lap_bor_ln, ugau_bor_ln, print.ndmodel, summary.ndmodel, plot.ndmodel

Examples

# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- ulap_bor_ln(x = iris2D[-ncol(iris2D)], y = iris2D[,ncol(iris2D)],
                      level = c(0.1, 0.2, 0.3), order = c("virginica", "setosa", "versicolor"))

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- ulap_bor_ln(formula = Species ~ ., data = iris2D,
                      level = c(0.1, 0.2, 0.3), order = c("virginica", "setosa", "versicolor"))

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

uncs_guni_cn

Unconditional/symmetric Gaussian/uniform combined noise

Description

Introduction of Unconditional/symmetric Gaussian/uniform combined noise into a classification dataset.

Usage

## Default S3 method:
uncs_guni_cn(x, y, level, k = 0.2, sortid = TRUE, ...)

## S3 method for class "formula"
uncs_guni_cn(formula, data, ...)

Arguments

x a data frame of input attributes.
y a factor vector with the output class of each sample.
level a double in [0,1] with the noise level to be introduced.
k a double in [0,1] with the scale used for the standard deviation (default: 0.2).
sortid   a logical indicating if the indices must be sorted at the output (default: TRUE).

...   other options to pass to the function.

formula   a formula with the output class and, at least, one input attribute.

data   a data frame in which to interpret the variables in the formula.

Details

*Unconditional/symmetric Gaussian/uniform combined noise* corrupts all the samples for each attribute in the dataset. Their values are corrupted by adding a random value following a Gaussian distribution of \( \text{mean} = 0 \) and \( \text{standard deviation} = (\text{max}-\text{min}) \cdot k \), being \( \text{max} \) and \( \text{min} \) the limits of the attribute domain. For nominal attributes, a random value is chosen. Additionally, this noise model also selects \((\text{level} \cdot 100)\)% of the samples in the dataset with independence of their class. The labels of these samples are randomly replaced by different ones within the set of class labels.

Value

An object of class `ndmodel` with elements:

- **xnoise**: a data frame with the noisy input attributes.
- **ynoise**: a factor vector with the noisy output class.
- **numnoise**: an integer vector with the amount of noisy samples per variable.
- **idnoise**: an integer vector list with the indices of noisy samples per variable.
- **numclean**: an integer vector with the amount of clean samples per variable.
- **idclean**: an integer vector list with the indices of clean samples per variable.
- **distr**: an integer vector with the samples per class in the original data.
- **model**: the full name of the noise introduction model used.
- **param**: a list of the argument values.
- **call**: the function call.

Note

Noise model adapted from the papers in References.

References


See Also

- `sym_cuni_cn`
- `sym_cuni_an`
- `print.ndmodel`
- `summary.ndmodel`
- `plot.ndmodel`
Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- uncs_guni_cn(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- uncs_guni_cn(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```

---

**unc_fixw_an**

Unconditional fixed-width attribute noise

Description

Introduction of Unconditional fixed-width attribute noise into a classification dataset.

Usage

```r
## Default S3 method:
unc_fixw_an(x, y, level, k = 0.1, sortid = TRUE, ...)

## S3 method for class 'formula'
unc_fixw_an(formula, data, ...)
```

Arguments

- **x**: a data frame of input attributes.
- **y**: a factor vector with the output class of each sample.
- **level**: a double in [0,1] with the noise level to be introduced in nominal attributes.
- **k**: a double in [0,1] with the domain proportion of the noise width (default: 0.1).
- **sortid**: a logical indicating if the indices must be sorted at the output (default: TRUE).
- **formula**: a formula with the output class and, at least, one input attribute.
- **data**: a data frame in which to interpret the variables in the formula.
Details

Unconditional fixed-width attribute noise corrupts all the samples in the dataset. For each attribute \( A \), all the original values are corrupted by adding a random number in the interval \([-\text{width}, \text{width}]\), being \( \text{width} = (\text{max}(A) - \text{min}(A))\cdot k \). For nominal attributes, \((\text{level}\cdot100)\%\) of the samples in the dataset are chosen and a random value is selected as noisy.

Value

An object of class \texttt{ndmodel} with elements:

- \texttt{xnoise}\ a data frame with the noisy input attributes.
- \texttt{ynoise}\ a factor vector with the noisy output class.
- \texttt{numnoise}\ an integer vector with the amount of noisy samples per attribute.
- \texttt{idnoise}\ an integer vector list with the indices of noisy samples per attribute.
- \texttt{numclean}\ an integer vector with the amount of clean samples per attribute.
- \texttt{idclean}\ an integer vector list with the indices of clean samples per attribute.
- \texttt{distr}\ an integer vector with the samples per class in the original data.
- \texttt{model}\ the full name of the noise introduction model used.
- \texttt{param}\ a list of the argument values.
- \texttt{call}\ the function call.

Note

Noise model adapted from the papers in References, corrupting all samples and allowing nominal attributes.

References


See Also

\texttt{sym_end_an}, \texttt{sym_sgau_an}, \texttt{print.ndmodel}, \texttt{summary.ndmodel}, \texttt{plot.ndmodel}

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- unc_fixw_an(x = iris2D[,-ncol(iris2D)], y = iris2D[,ncol(iris2D)], level = 0.1)

# show results
```
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- unc_fixw_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)

---

**unc_vgau_an**  
*Unconditional vp-Gaussian attribute noise*

**Description**

Introduction of *Unconditional vp-Gaussian attribute noise* into a classification dataset.

**Usage**

```r
## Default S3 method:
unc_vgau_an(x, y, level, sortid = TRUE, ...)

## S3 method for class 'formula'
unc_vgau_an(formula, data, ...)
```

**Arguments**

- `x`: a data frame of input attributes.
- `y`: a factor vector with the output class of each sample.
- `level`: a double in $[0,1]$ with the noise level to be introduced.
- `sortid`: a logical indicating if the indices must be sorted at the output (default: TRUE).
- `...`: other options to pass to the function.
- `formula`: a formula with the output class and, at least, one input attribute.
- `data`: a data frame in which to interpret the variables in the formula.

**Details**

In *Unconditional vp-Gaussian attribute noise*, the noise level for numeric attributes indicates the magnitude of the errors introduced. For each attribute $A$, all the original values are corrupted by adding a random number that follows a Gaussian distribution with $\text{mean} = 0$ and $\text{variance} = \text{level}\%$ of the variance of $A$. For nominal attributes, $(\text{level}\times100)\%$ of the samples in the dataset are chosen and a random value is selected as noisy.
unc_vgau_an

Value

An object of class ndmodel with elements:

- \texttt{xnoise} a data frame with the noisy input attributes.
- \texttt{ynoise} a factor vector with the noisy output class.
- \texttt{numnoise} an integer vector with the amount of noisy samples per attribute.
- \texttt{idnoise} an integer vector list with the indices of noisy samples per attribute.
- \texttt{numclean} an integer vector with the amount of clean samples per attribute.
- \texttt{idclean} an integer vector list with the indices of clean samples per attribute.
- \texttt{distr} an integer vector with the samples per class in the original data.
- \texttt{model} the full name of the noise introduction model used.
- \texttt{param} a list of the argument values.
- \texttt{call} the function call.

Note

Noise model adapted from the papers in References, corrupting all samples and allowing nominal attributes.

References


See Also

\texttt{symd_rpix_an, unc_fixw_an, print.ndmodel, summary.ndmodel, plot.ndmodel}

Examples

```r
# load the dataset
data(iris2D)

# usage of the default method
set.seed(9)
outdef <- unc_vgau_an(x = iris2D[, -ncol(iris2D)], y = iris2D[, ncol(iris2D)], level = 0.1)

# show results
summary(outdef, showid = TRUE)
plot(outdef)

# usage of the method for class formula
set.seed(9)
outfrm <- unc_vgau_an(formula = Species ~ ., data = iris2D, level = 0.1)

# check the match of noisy indices
identical(outdef$idnoise, outfrm$idnoise)
```
Index

* datasets
  diris2D, 18
  iris2D, 35
  asy_depln, 3, 12, 43
  asy_int_an, 5, 15, 35
  asy_spa_ln, 7, 60
  asy_uni_an, 6, 9, 35
  asy_uni_ln, 11, 47
  attm_uni_ln, 12, 56, 75
  boud_gau_an
  clu_vot_ln, 16, 22, 65
  diris2D, 18
  exp_bor_ln, 20, 25
  exps_cuni_ln, 13, 19, 68
  fra_bdir_ln, 8, 22, 45
  gam_bor_ln, 24
  gau_bor_ln, 27, 28, 133
  gau_m_bor_ln, 26, 40, 133
  glev_uni_ln, 30, 87, 89
  hubp_uni_ln, 32, 41, 72
  imp_int_an, 15, 33
  iris, 35
  iris2D, 18, 35
  irs_bdir_ln, 23, 36, 45
  lap_bor_ln, 38, 52, 136
  larm_uni_ln, 29, 40, 72
  maj_udir_ln, 12, 42, 47
  mind_bdir_ln, 8, 44, 60
  mis_pre_ln, 17, 47, 70
  mulc_udir_ln, 49
  nei_bor_ln, 51, 54, 74
  nlin_bor_ln, 49, 53, 74
  oned_uni_ln, 33, 54, 75
  opes_idnn_ln, 57
  opes_idu_ln, 20, 58, 59
  pai_bdir_ln, 23, 38, 61
  plot.ndmodel, 4, 6, 8, 10, 12, 13, 15, 17, 18,
  20, 22, 23, 25, 27, 29, 31, 33, 35, 36,
  38, 40, 41, 43, 45, 47, 49, 50, 52, 54,
  56, 58, 60, 62, 63, 65, 66, 68, 70, 72,
  74, 75, 77, 78, 80, 82, 84, 85, 87, 89,
  91, 93, 94, 96, 98, 100, 102, 104,
  105, 107, 109, 111, 113, 115, 116,
  118, 120, 122, 124, 126, 128, 129,
  131, 133, 136, 137, 139, 141
  pmd_con_ln, 22, 25, 64
  print.ndmodel, 4, 6, 8, 10, 12, 13, 15, 17, 18,
  20, 22, 23, 25, 27, 29, 31, 33, 35, 36,
  38, 40, 41, 43, 45, 47, 49, 50, 52, 54,
  56, 58, 60, 62, 64, 65, 66, 68, 70, 72,
  74, 75, 77, 78, 80, 82, 84, 85, 87, 89,
  91, 93, 94, 96, 98, 100, 102, 104,
  105, 107, 109, 111, 113, 115, 116,
  118, 120, 122, 124, 126, 128, 129,
  131, 133, 136, 137, 139, 141
  qua_uni_ln, 13, 56, 67
  sco_con_ln, 17, 65, 69
  sigb_uni_ln, 27, 29, 71
  smam_bor_ln, 49, 70, 72
  smu_cuni_ln, 33, 41, 74
  summary.ndmodel, 4, 6, 8, 10, 12, 13, 15, 17,
  18, 20, 22, 23, 25, 27, 29, 31, 33, 35,
  36, 38, 40, 41, 43, 45, 47, 49, 50, 52,
  54, 56, 58, 60, 62, 64–66, 68, 70, 72,
  74, 75, 76, 78, 80, 82, 84, 85, 87, 89,
  91, 93, 94, 96, 98, 100, 102, 104,
INDEX

105, 107, 109, 111, 113, 115, 116, 118, 120, 122, 124, 126, 128, 129, 131, 133, 136, 137, 139, 141

sym_adj_ln, 84, 115, 120
sym_cen_ln, 86, 89, 116
sym_con_ln, 88, 116
sym_cuni_an, 84, 90, 93, 94, 105, 128, 137
sym_cuni_cn, 91, 91, 128, 137
sym_cuni_ln, 64, 66, 77, 93, 96, 105
sym_ddef_ln, 95, 98, 129
sym_def_ln, 97, 111, 129
sym_dia_ln, 99, 109, 118
sym_dran_ln, 85, 101, 120
sym_end_an, 82, 102, 139
sym_exc_ln, 96, 98, 104
sym_gau_an, 78, 106, 126
sym_hie_ln, 102, 110
sym_hienc_ln, 31, 87, 108
sym_int_an, 78, 107, 112
sym_natd_ln, 113, 122, 131
sym_nean_ln, 4, 115
sym_nexc_ln, 31, 109, 117
sym_nuni_ln, 115, 119, 131
sym_opt_ln, 100, 121, 124
sym_pes_ln, 100, 118, 123
sym_sgau_an, 104, 125, 126, 139
sym_uni_an, 36, 64, 66, 77, 84, 91, 113, 126
sym_uni_ln, 36, 64, 66, 77, 94, 111, 128
sym_usim_ln, 122, 124, 130
symd_gau_an, 77, 104
symd_gimg_an, 6, 10, 79
symd_rpix_an, 80, 81, 141
symd_uni_an, 83, 107, 113

ugau_bor_ln, 40, 132, 136
ulap_bor_ln, 52, 54, 134
unc_fixw_an, 82, 138, 141
unc_vgau_an, 10, 80, 140
uncs_guni_cn, 93, 136