Package 'NoiseFiltersR'

August 29, 2016

Type Package

Title Label Noise Filters for Data Preprocessing in Classification

Version 0.1.0

Description An extensive implementation of state-of-the-art and classical algorithms to preprocess label noise in classification problems.

License GPL-3

LazyData TRUE

Imports RWeka, kknn, nnet, caret, e1071, rpart, randomForest, MASS, rJava, stats, utils

Depends R (>= 3.3.0)

RoxygenNote 5.0.1

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Pablo Morales [aut],
Julian Luengo [aut, cre],
Luis P.F. Garcia [aut],
Ana C. Lorena [aut],
Andre C.P.L.F. de Carvalho [aut],
Francisco Herrera [aut]

Maintainer Julian Luengo <julianlm@decsai.ugr.es>

Repository CRAN

Date/Publication 2016-06-24 12:34:50

R topics documented:

AENN *All-k Edited Nearest Neighbors*

Description

Similarity-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
AENN(x, k = 5, classColumn = ncol(x), ...)
```

Arguments

`formula`	A formula describing the classification variable and the attributes to be used.
`data, x`	Data frame containing the tranining dataset to be filtered.
`...`	Optional parameters to be passed to other methods.
`k`	Total number of nearest neighbors to be used.
`classColumn`	Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

AENN applies the Edited Nearest Neighbor algorithm ENN for all integers between 1 and k on the whole dataset. At the end, any instance considered noisy by some *ENN* is removed.

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab contains the new labels for repaired instances.
- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

References

Tomek I. (1976, June): An Experiment with the Edited Nearest-Neighbor Rule, in *Systems, Man and Cybernetics, IEEE Transactions on*, vol.SMC-6, no.6, pp. 448-452.

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
out <- AENN(Species~.-Petal.Length,iris)
print(out)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

BBNR *Blame Based Noise Reduction*

Description

Similarity-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
BBNR(x, k = 3, classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
k	Number of nearest neighbors to be used.
classColumn	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

BBNR removes an instance 'X' if: (i) it participates in the misclassification of other instance (i.e. 'X' is among the k nearest neighbors of a misclassified instance and has a different class); and (ii) its removal does not produce a misclassification in instances that, initially, were correctly classified by 'X' (i.e. 'X' was initially among the k nearest neighbors and had the same class).

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.
- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

References

Delany S. J., Cunningham P. (2004): An analysis of case-base editing in a spam filtering system. In *Advances in Case-Based Reasoning* (pp. 128-141). Springer Berlin Heidelberg.

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
out <- BBNR(iris, k = 5)
print(out)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

C45ensembles *Classical Filters based on C4.5*

Description

Ensembled-based filters that use C4.5 classifier to remove label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

```
## S3 method for class 'formula'
C45robustFilter(formula, data, ...)

## Default S3 method:
C45robustFilter(x, classColumn = ncol(x), ...)

## S3 method for class 'formula'
C45votingFilter(formula, data, ...)

## Default S3 method:
C45votingFilter(x, nfolds = 10, consensus = FALSE,
  classColumn = ncol(x), ...)

## S3 method for class 'formula'
C45iteratedVotingFilter(formula, data, ...)

## Default S3 method:
C45iteratedVotingFilter(x, nfolds = 10, consensus = FALSE,
  classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
classColumn	Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

nfolds	Number of folds in which the dataset is split.
consensus	Logical. If TRUE, consensus voting scheme is used. If FALSE, majority voting scheme is applied.

Details

Full description of the methods can be looked up in the provided reference. Notice that C4.5 is used as base classifier instead of TILDE, since a standard attribute-value classification framework is considered (instead of the ILP classification approach of the reference).

`C45robustFilter` builds a C4.5 decision tree from the training data, and then removes those instances misclassfied by this tree. The process is repeated until no instances are removed.

`C45votingFilter` splits the dataset into `nfolds` folds, building and testing a C4.5 tree on every combination of `nfolds`-1 folds. Thus `nfolds`-1 votes are gathered for each instance. Removal is carried out by majority or consensus voting schemes.

`C45iteratedVotingFilter` somehow combines the two previous filter, since it iterates `C45votingFilter` until no more noisy instances are removed.

Value

An object of class `filter`, which is a list with seven components:

- `cleanData` is a data frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- `repIdx` is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- `repLab` is a factor containing the new labels for repaired instances.
- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

Note

By means of a message, the number of noisy instances removed is displayed in the console.

References

Verbaeten S. (2002, December): Identifying mislabeled training examples in ILP classification problems, in *Proc. 12th Belgian-Dutch Conf. Mach. Learn.*, Utrecht, The Netherlands, pp. 71-78.

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
out1 <- C45robustFilter(Species~.-Sepal.Length, iris)
# We fix a seed since next two functions create partitions of data for the ensemble
set.seed(1)
out2 <- C45votingFilter(iris, consensus = TRUE)
out3 <- C45iteratedVotingFilter(Species~., iris, nfolds = 5)
print(out1)
print(out2)
print(out3)
identical(out1$cleanData,iris[setdiff(1:nrow(iris),out1$remIdx),])
identical(out2$cleanData,iris[setdiff(1:nrow(iris),out2$remIdx),])
identical(out3$cleanData,iris[setdiff(1:nrow(iris),out3$remIdx),])

## End(Not run)
```

CNN *Condensed Nearest Neighbors*

Description

Similarity-based method designed to select the most relevant instances for subsequent classification with a *nearest neighbor* rule. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
CNN(x, classColumn = ncol(x), ...)
```

Arguments

`formula`	A formula describing the classification variable and the attributes to be used.
`data, x`	Data frame containing the tranining dataset to be filtered.
`...`	Optional parameters to be passed to other methods.
`classColumn`	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

CNN searches for a 'consistent subset' of the provided dataset, i.e. a subset that is enough for correctly classifying the rest of instances by means of 1-NN. To do so, CNN stores the first instance and goes for a first sweep over the dataset, adding to the stored bag those instances which are not correctly classified by 1-NN taking the stored bag as training set. Then, the process is iterated until all non-stored instances are correctly classified.

Although CNN is not strictly a label noise filter, it is included here for completeness, since the origins of noise filters are connected with instance selection algorithms.

Value

An object of class `filter`, which is a list with seven components:

- `cleanData` is a data frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- `repIdx` is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- `repLab` is a factor containing the new labels for repaired instances.
- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

References

Hart P. (May, 1968): The condensed nearest neighbor rule, *IEEE Trans. Inf. Theory*, vol. 14, no. 5, pp. 515-516.

See Also

RNN

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
out <- CNN(iris)
print(out)
length(out$remIdx)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

CVCF *Cross-Validated Committees Filter*

Description

Ensemble-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
CVCF(x, nfolds = 10, consensus = FALSE,
  classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
nfolds	number of folds in which the dataset is split.
consensus	logical. If TRUE, consensus voting scheme is used. If FALSE, majority voting scheme is applied.
classColumn	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

Full description of the method can be looked up in the provided references. Dataset is split in nfolds folds, a base classifiers (C4.5 in this implementation) is built over every combination of nfolds-1 folds, and then tested on the whole dataset. Finally, consensus or majority voting scheme is applied to remove noisy instances.

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.

- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

References

Verbaeten S., Van Assche A. (2003, June): Ensemble methods for noise elimination in classification problems. *Proc. 4th Int. Conf. Multiple Classifier Syst.*, Guildford, U.K., pp. 317-325.

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
# We fix a seed since there exists a random partition for the ensemble
set.seed(1)
out <- CVCF(Species~.-Sepal.Width, data = iris)
print(out)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

DROP *Decremental Reduction Optimization Procedures*

Description

Similarity-based filters for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

```
## S3 method for class 'formula'
DROP1(formula, data, ...)

## Default S3 method:
DROP1(x, k = 1, classColumn = ncol(x), ...)

## S3 method for class 'formula'
DROP2(formula, data, ...)

## Default S3 method:
DROP2(x, k = 1, classColumn = ncol(x), ...)

## S3 method for class 'formula'
DROP3(formula, data, ...)
```

```
## Default S3 method:
DROP3(x, k = 1, classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
k	Number of nearest neighbors to be used.
classColumn	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

DROP1 goes over the dataset in the provided order, and removes those instances whose removal does not decrease the accuracy of the 1-NN rule in the remaining dataset.

DROP2 introduces two modifications against DROP1. Regarding the order of processing instances, DROP2 starts with those which are furthest from their nearest "enemy" (two instances are said to be "enemies" if they belong to different classes). Moreover, DROP2 removes an instance if its removal does not decrease the accuracy of the 1-NN rule in the *original* dataset (rather than the *remaining* dataset as in DROP1).

DROP3 is identical to DROP2, but it includes a preprocessing step to clean the borders between classes. It consists of applying the ENN method: any instance misclassified by its k nearest neighbors is removed.

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.
- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

References

Wilson D. R., Martinez T. R. (2000): Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3), 257-286. Wilson D. R., Martinez T. R. (1997, July): Instance pruning techniques. In *ICML* (Vol. 97, pp. 403-411).

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
trainData <- iris[c(1:20,51:70,101:120),]
out1 <- DROP1(Species~ Petal.Length + Petal.Width, data = trainData)
summary(out1, explicit = TRUE)
identical(out1$cleanData, trainData[setdiff(1:nrow(trainData),out1$remIdx),])

## End(Not run)
```

`dynamicCF` *Dynamic Classification Filter*

Description

Ensemble-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
dynamicCF(x, nfolds = 10, consensus = FALSE, m = 3,
  classColumn = ncol(x), ...)
```

Arguments

`formula`	A formula describing the classification variable and the attributes to be used.
`data, x`	Data frame containing the tranining dataset to be filtered.
`...`	Optional parameters to be passed to other methods.
`nfolds`	Number of folds for the cross voting scheme.
`consensus`	If set to `TRUE`, consensus voting scheme is applied. Otherwise (default), majority scheme is used.
`m`	Number of classifiers to make up the ensemble. It must range between 1 and 9.
`classColumn`	Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

`dynamicCF` (Garcia et al., 2012) follows the same approach as `EF`, but the ensemble of classifiers is not fixed beforehand. Namely, `dynamicCF` trains 9 well-known classifiers in the dataset to be filtered, and selects for the ensemble those with the `m` best predictions. Then, a `nfolds`-folds cross voting scheme is applied, with consensus or majority strategies depending on parameter `consensus`.

The nine (standard) classifiers handled by `dynamicCF` are SVM, 3-KNN, 5-KNN, 9-KNN, CART, C4.5, Random Forest, Naive Bayes and Multilayer Perceptron Neural Network.

Value

An object of class `filter`, which is a list with seven components:

- `cleanData` is a data frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- `repIdx` is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- `repLab` is a factor containing the new labels for repaired instances.
- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

References

Garcia L. P. F., Lorena A. C., Carvalho A. C. (2012, October): A study on class noise detection and elimination. In *Brazilian Symposium on Neural Networks (SBRN)*, pp. 13-18, IEEE.

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
trainData <- iris[c(1:20,51:70,101:120),]
# We fix a seed since there exists a random partition for the ensemble
set.seed(1)
out <- dynamicCF(Species~Petal.Length + Sepal.Length, data = trainData, nfolds = 5, m = 3)
summary(out, explicit = TRUE)
identical(out$cleanData, trainData[setdiff(1:nrow(trainData),out$remIdx),])

## End(Not run)
```

---

edgeBoostFilter *Edge Boosting Filter*

Description

Ensemble-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
edgeBoostFilter(x, m = 15, percent = 0.05,
  threshold = 0, classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
m	Number of boosting iterations
percent	Real number between 0 and 1. It sets the percentage of instances to be removed (as long as their edge value exceeds the parameter threshold).
threshold	Real number between 0 and 1. It sets the minimum edge value required by an instance in order to be removed.
classColumn	Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

The full description of the method can be looked up in the provided reference.

An AdaBoost scheme (Freund & Schapire) is applied with a default C4.5 tree as weak classifier. After m iterations, those instances with larger (according to the constraints percent and threshold) edge values (Wheway, Freund & Schapire) are considered noisy and thus removed.

Notice that making use of extreme values (i.e. percent=1 or threshold=0) any 'removing constraints' can be ignored.

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.
- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

References

Freund Y., Schapire R. E. (1997): A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1), 119-139.

Wheway V. (2001, January): Using boosting to detect noisy data. In *Advances in Artificial Intelligence*. PRICAI 2000 Workshop Reader (pp. 123-130). Springer Berlin Heidelberg.

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
out <- edgeBoostFilter(Species~., data = iris, m = 10, percent = 0.05, threshold = 0)
print(out)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

EF *Ensemble Filter*

Description

Ensemble-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
EF(x, nfolds = 4, consensus = TRUE,
  classColumn = ncol(x), ...)
```

Arguments

`formula`	A formula describing the classification variable and the attributes to be used.
`data, x`	data frame containing the tranining dataset to be filtered.
`...`	Optional parameters to be passed to other methods.
`nfolds`	number of folds in which the dataset is split.
`consensus`	logical. If TRUE, consensus voting scheme is used. If FALSE, majority voting scheme is applied.
`classColumn`	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

Full description of the method can be looked up in the provided references. Dataset is split in `nfolds` folds, an ensemble of three different base classifiers (C4.5, 1-KNN, LDA) is built over every combination of `nfolds`-1 folds, and then tested on the other one. Finally, consensus or majority voting scheme is applied to remove noisy instances.

Value

An object of class `filter`, which is a list with seven components:

- `cleanData` is a data frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- `repIdx` is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- `repLab` is a factor containing the new labels for repaired instances.
- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

References

Brodley C. E., Friedl M. A. (1996, May): Improving automated land cover mapping by identifying and eliminating mislabeled observations from training data. In *Geoscience and Remote Sensing Symposium, 1996. IGARSS'96.'Remote Sensing for a Sustainable Future.', International* (Vol. 2, pp. 1379-1381). IEEE.

Brodley C. E., Friedl M. A. (1996, August): Identifying and eliminating mislabeled training instances. In *AAAI/IAAI*, Vol. 1 (pp. 799-805).

Brodley C. E., Friedl M. A. (1999): Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 131-167.

Examples

```
data(iris)
# We fix a seed since there exists a random partition for the ensemble
set.seed(1)
out <- EF(Species~., data = iris, consensus = FALSE)
summary(out, explicit = TRUE)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])
```

ENG *Editing with Neighbor Graphs*

Description

Similarity-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
ENG(x, graph = "RNG", classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
graph	Character indicating the type of graph to be constructed. It can be chosen between 'GG' (Gabriel Graph) and 'RNG' (Relative Neighborhood Graph). See 'References' for more details on both graphs.
classColumn	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

ENG builds a neighborhood graph which can be either *Gabriel Graph (GG)* or *Relative Neighborhood Graph (RNG)* [S\'anchez et al., 1997]. Then, an instance is considered as 'potentially noisy' if most of its neighbors have a different class. To decide whether such an instance 'X' is removed, let S be the subset given by 'X' together with its neighbors from the same class. Compute the majority class 'C' among the neighbors of examples in S, and remove 'X' if its class is not 'C'.

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.

- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

References

S\'anchez J. S., Pla F., Ferri F. J. (1997): Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters*, 18(6), 507-513.

Examples

```
# The example is not run because the graph construction is quite time-consuming.
## Not run:
   data(iris)
   trainData <- iris[c(1:20,51:70,101:120),]
   out <- ENG(Species~Petal.Length + Petal.Width, data = trainData, graph = "RNG")
   print(out)
   identical(out$cleanData,trainData[setdiff(1:nrow(trainData),out$remIdx),])

## End(Not run)
```

ENN *Edited Nearest Neighbors*

Description

Similarity-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
ENN(x, k = 3, classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
k	Number of nearest neighbors to be used.
classColumn	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

ENN finds the k nearest neighbors for each instance, which is removed if the majority class in this neighborhood is different from its class.

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.
- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

References

Wilson D. L. (1972): Asymptotic properties of nearest neighbor rules using edited data. *Systems, Man and Cybernetics, IEEE Transactions on*, (3), 408-421.

Examples

```
data(iris)
out <- ENN(Species~., data = iris, k = 5)
summary(out)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])
```

EWF *Edge Weight Filter*

Description

Similarity-based filter for removing or repairing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
EWF(x, threshold = 0.25, noiseAction = "remove",
  classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
threshold	Real number between 0 and 1. It sets the limit between good and suspicious instances. Its default value is 0.25.
noiseAction	Character being either 'remove' or 'hybrid'. It determines what to do with noisy instances. By default, it is set to 'remove'.
classColumn	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

EWF builds up a Relative Neighborhood Graph (RNG) from the dataset. Then, it identifies as 'suspicious' those instances with a significant value of its*local cut edge weight statistic*, which intuitively means that they are surrounded by examples from a different class.

Namely, the aforementioned statistic is the sum of the weights of edges joining the instance (in the RNG graph) with instances from a different class. Under the null hypothesis of the class label being independent of the event 'being neighbors in the RNG graph', the distribution of this statistic can be approximated by a gaussian one. Then, the p-value for the observed value is computed and contrasted with the provided threshold.

To handle 'suspicious' instances there are two approaches ('remove' or 'hybrid'), and the argument 'noiseAction' determines which one to use. With 'remove', every suspect is removed from the dataset. With the 'hybrid' approach, an instance is removed if it does not have *good* (i.e. non-suspicious) RNG-neighbors. Otherwise, it is relabelled with the majority class among its *good* RNG-neighbors.

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.
- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

References

Muhlenbach F., Lallich S., Zighed D. A. (2004): Identifying and handling mislabelled instances. *Journal of Intelligent Information Systems*, 22(1), 89-109.

Examples

```
# Next example is not run because EWF is time-consuming
## Not run:
   data(iris)
   trainData <- iris[c(1:20,51:70,101:120),]
  out <- EWF(Species~Petal.Length+Sepal.Length, data = trainData, noiseAction = "hybrid")
   print(out)

## End(Not run)
```

GE *Generalized Edition*

Description

Similarity-based filter for removing or repairing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
GE(x, k = 5, kk = ceiling(k/2), classColumn = ncol(x),
  ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
k	Number of nearest neighbors to be considered.
kk	Minimum size for local majority class in order to relabel an instance.
classColumn	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

GE is a generalization of ENN that integrates the possibility of 'repairing' or 'relabeling' instances rather than only 'removing'. For each instance, GE considers its k-1 neighbors and the instance itself. If there are at least kk examples from the same class, the instance is relabeled with that class (which could be its own). Otherwise, it is removed.

Value

An object of class `filter`, which is a list with seven components:

- `cleanData` is a data frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- `repIdx` is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- `repLab` is a factor containing the new labels for repaired instances.
- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

References

Koplowitz J., Brown T. A. (1981): On the relation of performance to editing in nearest neighbor rules. *Pattern Recognition*, 13(3), 251-255.

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
out <- GE(iris)
summary(out, explicit = TRUE)
# We check that the process was correct
irisCopy <- iris
irisCopy[out$repIdx,5] <- out$repLab
cleanData <- irisCopy[setdiff(1:nrow(iris),out$remIdx),]
identical(out$cleanData,cleanData)

## End(Not run)
```

HARF *High Agreement Random Forest*

Description

Ensemble-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
HARF(x, nfolds = 10, agreementLevel = 0.7, ntrees = 500,
  classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
nfolds	Number of folds for the cross voting scheme.
agreementLevel	Real number between 0.5 and 1. An instance is identified as noise when the classification confidences provided by the random forest to the classes that are not the actual class of the instance add up at least agreementLevel. Authors obtain the best performance in (Sluban et al., 2010) when setting it between 0.7 and 0.8.
ntrees	Number of trees for the random forest.
classColumn	Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

Making use of a nfolds-folds cross validation scheme, instances are identified as noise and removed when a random forest provides little confidence for the actual instance's label (namely, less than 1-agreementLevel). The value of agreementLevel allows to tune the precision and recall of the filter, getting the best trade-off when moving between 0.7 and 0.8 (Sluban et al., 2010).

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.
- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

References

Sluban B., Gamberger D., Lavrac N. (2010, August): Advances in Class Noise Detection. In *ECAI* (pp. 1105-1106).

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
# We fix a seed since there exists a random partition for the ensemble
set.seed(1)
out <- HARF(Species~., data = iris, ntrees = 100)
print(out)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

---

hybridRepairFilter *Hybrid Repair-Remove Filter*

Description

Ensemble-based filter for removing or repairing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
hybridRepairFilter(x, consensus = FALSE,
  noiseAction = "remove", classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be processed.
...	Optional parameters to be passed to other methods.
consensus	If set to TRUE, consensus voting scheme is applied to identify noisy instances. Otherwise (default), majority approach is used.
noiseAction	Character which can be set to "remove", "repair" or "hybrid". The filter accordingly decides what to do with the identified noise (see Details).
classColumn	Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

As presented in (Miranda et al., 2009), `hybridRepairFilter` builds on the dataset an ensemble of four classifiers: SVM, Neural Network, CART, KNN (combining k=1,3,5). According to their predictions and majority or consensus voting schemes, a subset of instances are labeled as noise. These are removed if `noiseAction` equals "remove", their class is changed into the most voted among the ensemble if `noiseAction` equals "repair", and when the latter is set to "hybrid", the vote of KNN decides whether remove or repair.

All this procedure is repeated while the accuracy (over the original dataset) of the ensemble trained with the processed dataset increases.

Value

An object of class `filter`, which is a list with seven components:

- `cleanData` is a data frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- `repIdx` is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- `repLab` is a factor containing the new labels for repaired instances.
- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

References

Miranda A. L., Garcia L. P. F., Carvalho A. C., Lorena A. C. (2009): Use of classification algorithms in noise detection and elimination. In *Hybrid Artificial Intelligence Systems* (pp. 417-424). Springer Berlin Heidelberg.

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
out <- hybridRepairFilter(iris, noiseAction = "hybrid")
summary(out, explicit = TRUE)

## End(Not run)
```

INFFC *Iterative Noise Filter based on the Fusion of Classifiers*

Description

Ensemble-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
INFFC(x, consensus = FALSE, p = 0.01, s = 3, k = 5,
  threshold = 0, classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
consensus	Logical. If FALSE, majority voting scheme is used for 'preliminary filtering' and 'noise free filtering' (see 'Details' and References' section). If TRUE, consensus voting scheme is applied.
p	Real number between 0 and 1. It sets the minimum proportion of original instances which must be tagged as noisy in order to go for another iteration.
s	Positive integer setting the stop criterion together with p. The filter stops after s iterations with not enough noisy instances removed (according to the proportion p).
k	Parameter for the k-nearest neighbors algorithm used for the 'noise score' stage (see 'Details' and 'References').
threshold	Real number between -1 and 1. It sets the noise score value above which an instance is removed.
classColumn	Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

The full description of the method can be looked up in the provided reference. A 'preliminary filtering' is carried out with a fusion of classifiers (FC), including C4.5, 3NN, and logistic regression. Then, potentially noisy instances are identified in a 'noise free filtering' process building the FC on the (preliminary) filtered instances. Finally, a 'noise score' is computed on these potentially noisy instances, removing those exceeding the threshold value. The process stops after s iterations with not enough (according to the proportion p) noisy instances removed.

Value

An object of class `filter`, which is a list with seven components:

- `cleanData` is a data frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- `repIdx` is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- `repLab` is a factor containing the new labels for repaired instances.
- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

Note

By means of a message, the number of noisy instances removed in each iteration is displayed in the console.

References

S\'aez J. A., Galar M., Luengo J., Herrera F. (2016): INFFC: An iterative class noise filter based on the fusion of classifiers with noise sensitivity control. *Information Fusion*, 27, 19-32.

Examples

```
# Next example is not run because it might be time-consuming
## Not run:
data(iris)
out <- INFFC(Species~., data = iris)
summary(out)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

IPF *Iterative Partitioning Filter*

Description

Ensemble-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
IPF(x, nfolds = 5, consensus = FALSE, p = 0.01, s = 3,
  y = 0.5, classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
nfolds	Number of partitions in each iteration.
consensus	Logical. If FALSE, majority voting scheme is used. If TRUE, consensus voting scheme is applied.
p	Real number between 0 and 1. It sets the minimum proportion of original instances which must be tagged as noisy in order to go for another iteration.
s	Positive integer setting the stop criterion together with p. The filter stops after s iterations with not enough noisy instances removed (according to the proportion p, see the 'Details').
y	Real number between 0 and 1. It sets the proportion of good instances which must be stored in each iteration.
classColumn	Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

The full description of the method can be looked up in the provided references. A base classifier is built in each of the nfolds partitions of data. Then, they are tested in the whole dataset, and the removal of noisy instances is decided via consensus or majority voting schemes. Finally, a proportion of good instances (i.e. those whose label agrees with all the base classifiers) is stored and removed for the next iteration. The process stops after s iterations with not enough (according to the proportion p) noisy instances removed. In this implementation, the base classifier used is C4.5.

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.

- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

Note

By means of a message, the number of noisy instances removed in each iteration is displayed in the console.

References

Khoshgoftaar T. M., Rebours P. (2007): Improving software quality prediction by noise filtering techniques. *Journal of Computer Science and Technology*, 22(3), 387-396.

Zhu X., Wu X., Chen Q. (2003, August): Eliminating class noise in large datasets. *International Conference in Machine Learning* (Vol. 3, pp. 920-927).

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
# We fix a seed since there exists a random folds partition for the ensemble
set.seed(1)
out <- IPF(Species~., data = iris, s = 2)
summary(out, explicit = TRUE)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

`ModeFilter` *Mode Filter*

Description

Similarity-based filter for removing or repairing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
ModeFilter(x, type = "classical", noiseAction = "repair",
  epsilon = 0.05, maxIter = 100, alpha = 1, beta = 1,
  classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
type	Character indicating the scheme to be used. It can be 'classical', 'iterative' or 'weighted'.
noiseAction	Character indicating what to do with noisy instances. It can be either 'remove' or 'repair'.
epsilon	If 'iterative' type is used, the loop will be stopped if the proportion of modified instances is less or equal than this threshold.
maxIter	Maximum number of iterations in 'iterative' type.
alpha	Parameter used in the computation of the similarity between two instances.
beta	It regulates the influence of the similarity metric in the estimation of a new label for an instance.
classColumn	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

ModeFilter estimates the most appropriate class for each instance based on the similarity metric and the provided label. This can be addressed in three different ways (argument 'type'):

In the classical approach, all labels are tried for all instances, and the one maximizing a metric based on similarity is chosen. In the iterative approach, the same scheme is repeated until the proportion of modified instances is less than *epsilon* or the maximum number of iterations *maxIter* is reached. The weighted approach extends the classical one by assigning a weight for each instance, which quantifies the reliability on its label. This weights is utilized in the computation of the metric to be maximized.

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.
- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

References

Du W., Urahama K. (2010, November): Error-correcting semi-supervised pattern recognition with mode filter on graphs. In *Aware Computing (ISAC), 2010 2nd International Symposium on* (pp. 6-11). IEEE.

Examples

```
# Next example is not run because in some cases it can be rather slow
## Not run:
   data(iris)
   out <- ModeFilter(Species~., data = iris, type = "classical", noiseAction = "remove")
   print(out)
   identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

`ORBoostFilter` *Outlier Removal Boosting Filter*

Description

Ensemble-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
ORBoostFilter(x, N = 20, d = 11, Naux = max(20, N),
  useDecisionStump = FALSE, classColumn = ncol(x), ...)
```

Arguments

`formula`	A formula describing the classification variable and the attributes to be used.
`data, x`	Data frame containing the tranining dataset to be filtered.
`...`	Optional parameters to be passed to other methods.
`N`	Number of boosting iterations.
`d`	Threshold for removing noisy instances. Authors recommend to set it between 3 and 20. If it is set to `NULL`, the optimal threshold is chosen according to the procedure described in Karmaker & Kwek. However, this can be very time-consuming, and in most cases is little relevant for the final result.
`Naux`	Number of boosting iterations for AdaBoost when computing the optimal threshold 'd'.

useDecisionStump
: If TRUE, a decision stump is used as weak classifier. Otherwise (default), naive-Bayes is applied. Recall decision stumps are not appropriate for multi-class problems.

classColumn
: Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

The full description of ORBoostFilter method can be looked up in Karmaker & Kwek. In general terms, a weak classifier is built in each iteration, and misclassified instances have their weight increased for the next round. Instances are removed when their weight exceeds the threshold d, i.e. they have been misclassified in consecutive rounds.

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.
- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

Note

By means of a message, the number of noisy instances removed in each iteration is displayed in the console.

References

Karmaker A., Kwek S. (2005, November): A boosting approach to remove class label noise. In *Hybrid Intelligent Systems*, 2005. HIS'05. Fifth International Conference on (pp. 6-pp). IEEE.

Freund Y., Schapire R. E. (1997): A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1), 119-139.

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
out <- ORBoostFilter(Species~., data = iris, N = 10)
summary(out)
```

```
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

PF *Partitioning Filter*

Description

Ensemble-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
PF(x, nfolds = 5, consensus = FALSE, p = 0.01, s = 3,
  y = 0.5, theta = 0.7, classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
nfolds	Number of partitions in each iteration.
consensus	Logical. If FALSE, majority voting scheme is used. If TRUE, consensus voting scheme is applied.
p	Real number between 0 and 1. It sets the minimum proportion of original instances which must be tagged as noisy in order to go for another iteration.
s	Positive integer setting the stop criterion together with p. The filter stops after s iterations with not enough noisy instances removed (according to the proportion p).
y	Real number between 0 and 1. It sets the proportion of good instances which must be stored in each iteration.
theta	Real number between 0 and 1. It sets the proportion of 'good rules' to be selected (see also 'Details' section).
classColumn	Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

The full description of the method can be looked up in the provided references. A PART rules set (from RWeka) is built in each of the `nfolds` partitions of `data`. After a 'good rules selection' process based on the accuracy of each rule, the subsequent good rules sets are tested in the whole dataset, and the removal of noisy instances is decided via consensus or majority voting schemes. Finally, a proportion of good instances (i.e. those whose label agrees with all the base classifiers) is stored and not considered in subsequent iterations. The process stops after `s` iterations with not enough (according to the proportion `p`) noisy instances removed.

Value

An object of class `filter`, which is a list with seven components:

- `cleanData` is a data frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- `repIdx` is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- `repLab` is a factor containing the new labels for repaired instances.
- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

Note

The base rule classifier used is PART instead of C4.5rules used in the references.

For the 'good rules selection' step, we implement the 'Best-L rules' scheme since, according to the authors, it usually outperforms the others 'Adaptive Threshold' and 'Fixed Threshold' schemes.

By means of a message, the number of noisy instances removed in each iteration is displayed in the console.

References

Zhu X., Wu X., Chen Q. (2003, August): Eliminating class noise in large datasets. *International Conference in Machine Learning* (Vol. 3, pp. 920-927).

Zhu X., Wu X., Chen Q. (2006): Bridging local and global data cleansing: Identifying class noise in large, distributed data datasets. *Data mining and Knowledge discovery*, 12(2-3), 275-308.

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
# We fix a seed since there exists a random partition for the ensemble
set.seed(1)
out <- PF(Species~., data = iris, s = 1, nfolds = 3)
```

```
print(out)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

PRISM *PReprocessing Instances that Should be Misclassified*

Description

Similarity-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
PRISM(x, classColumn = ncol(x), ...)
```

Arguments

`formula`	A formula describing the classification variable and the attributes to be used.
`data, x`	Data frame containing the tranining dataset to be filtered.
`...`	Optional parameters to be passed to other methods.
`classColumn`	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

`PRISM` identifies *ISMs* (Instances that Should be Misclassified) and removes them from the dataset. In order to do so, it combines five heuristics based on varied approaches by means of a formula. One heuristic relies on class distribution among nearest neighbors, two heuristics are based on the class distribution in a leaf node of a C4.5 tree (either pruned or unpruned), and the other two are based on the class likelihood for an instance, assuming gaussian distribution for continuous variables when necessary.

Value

An object of class `filter`, which is a list with seven components:

- `cleanData` is a data frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- `repIdx` is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).

- repLab is a factor containing the new labels for repaired instances.
- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

References

Smith M. R., Martinez T. (2011, July): Improving classification accuracy by identifying and removing instances that should be misclassified. In *Neural Networks (IJCNN), The 2011 International Joint Conference on* (pp. 2690-2697). IEEE.

Examples

```
data(iris)
out <- PRISM(Species~., data = iris)
print(out)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])
```

RNN *Reduced Nearest Neighbors*

Description

Similarity-based method designed to select the most relevant instances for subsequent classification with a *nearest neighbor* rule. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
RNN(x, classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
classColumn	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

RNN is an extension of CNN. The latter provides a 'consistent subset', i.e. it is enough for correctly classifying the rest of instances by means of 1-NN. Then, in the given order, RNN removes instances as long as the remaining do not loss the property of being a 'consistent subset'.

Although RNN is not strictly a class noise filter, it is included here for completeness, since the origins of noise filters are connected with instance selection algorithms.

Value

An object of class `filter`, which is a list with seven components:

- `cleanData` is a data frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- `repIdx` is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- `repLab` is a factor containing the new labels for repaired instances.
- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

References

Gates G.W. (1972): The Reduced Nearest Neighbour Rule. *IEEE Transactions on Information Theory*, 18:3 431-433.

See Also

CNN

Examples

```
# Next example is not run in order to save time
## Not run:
data(iris)
out <- RNN(Species~., data = iris)
print(out)
identical(out$cleanData, iris[setdiff(1:nrow(iris),out$remIdx),])

## End(Not run)
```

saturationFilter *Saturation Filters*

Description

Data complexity based filters for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
saturationFilter(x, noiseThreshold = NULL,
  classColumn = ncol(x), ...)

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

## Default S3 method:
consensusSF(x, nfolds = 10, consensusLevel = nfolds - 1,
  noiseThreshold = NULL, classColumn = ncol(x), ...)

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

## Default S3 method:
classifSF(x, nfolds = 10, noiseThreshold = NULL,
  classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
noiseThreshold	The threshold for removing noisy instances in the saturation filter. Authors recommend values between 0.25 and 2. If it is set to NULL, the threshold is appropriately chosen according to the number of training instances.
classColumn	Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.
nfolds	For consensusSF and classifSF, number of folds in which the dataset is split.
consensusLevel	For consensusSF, it sets the (minimum) number of 'noisy votes' an instance must get in order to be removed. By default, the nfolds-1 filters built over each instance must label it as noise.

Details

Based on theoretical studies about data complexity (Gamberger & Lavrac, 1997), `saturationFilter` removes those instances which most enable to reduce the CLCH (Complexity of the Least Complex Hypotheses) of the training dataset. The full method can be looked up in (Gamberger et al., 1999), and the previous step of *literals* extraction is detailed in (Gamberger et al., 1996).

`consensusSF` splits the dataset in `nfolds` folds, and applies `saturationFilter` to every combination of `nfolds-1` folds. Those instances with (at least) `consensusLevel` 'noisy votes' are removed.

`classifSF` combines `saturationFilter` with a `nfolds`-folds cross validation scheme (the latter in the spirit of filters such as `EF`, `CVCF`). Namely, the dataset is split in `nfolds` folds and, for every combination of `nfolds-1` folds, `saturationFilter` is applied and a classifier (we implement a standard C4.5 tree) is built. Instances from the excluded fold are removed according to this classifier.

Value

An object of class `filter`, which is a list with seven components:

- `cleanData` is a data frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- `repIdx` is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- `repLab` is a factor containing the new labels for repaired instances.
- `parameters` is a list containing the argument values.
- `call` contains the original call to the filter.
- `extraInf` is a character that includes additional interesting information not covered by previous items.

References

Gamberger D., Lavrac N., Groselj C. (1999, June): Experiments with noise filtering in a medical domain. In *ICML* (pp. 143-151).

Gamberger D., Lavrac N., Dzeroski S. (1996, January): Noise elimination in inductive concept learning: A case study in medical diagnosis. In *Algorithmic Learning Theory* (pp. 199-212). Springer Berlin Heidelberg.

Gamberger D., Lavrac N. (1997): Conditions for Occam's razor applicability and noise elimination (pp. 108-123). Springer Berlin Heidelberg.

Examples

```
# Next example is not run because saturation procedure is time-consuming.
## Not run:
data(iris)
out1 <- saturationFilter(Species~., data = iris)
out2 <- consensusSF(Species~., data = iris)
out3 <- classifSF(Species~., data = iris)
print(out1)
```

```
print(out2)
print(out3)

## End(Not run)
```

`summary.filter` *Summary method for class filter*

Description

This methods allows for appropriately displaying the most important information about a filtered dataset, contained in the S3 class `filter`.

Usage

```
## S3 method for class 'filter'
summary(object, ..., explicit = FALSE)
```

Arguments

`object`	Object of class `filter`.
`...`	Additional arguments affecting the summary produced.
`explicit`	If set to `TRUE`, the indexes for removed and repaired instances (as well as new labels for the latters) are displayed. It defaults to `FALSE`.

Details

The information offered is the following:

- Names of the dataset and the filter.
- Original call to the filter.
- Specific parameters used for the filter.
- Results: number of removed and repaired instances (absolute number and percentage).
- Additional information (if available, it depends on the filter).
- Optionally, if `explicit=TRUE`, the indexes for removed and repaired instances, as well as the new labels.

Examples

```
# Next example is not run in order to save time
## Not run:
# Example of filter with additional information available.
data(iris)
out <- edgeBoostFilter(Species~., data = iris)
class(out)
summary(out)
summary(out, explicit = TRUE)

## End(Not run)
```

TomekLinks *TomekLinks*

Description

Similarity-based filter for removing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Usage

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

## Default S3 method:
TomekLinks(x, classColumn = ncol(x), ...)
```

Arguments

formula	A formula describing the classification variable and the attributes to be used.
data, x	Data frame containing the tranining dataset to be filtered.
...	Optional parameters to be passed to other methods.
classColumn	positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Details

The function TomekLinks removes "TomekLink points" from the dataset. These are introduced in [Tomek, 1976], and are expected to lie on the border between classes. Removing such points is a typical procedure for cleaning noise [Lorena, 2002].

Since the computation of mean points is necessary for TomekLinks, only numeric attributes are allowed. Moreover, only two different classes are allowed to detect TomekLinks.

Value

An object of class filter, which is a list with seven components:

- cleanData is a data frame containing the filtered dataset.
- remIdx is a vector of integers indicating the indexes for removed instances (i.e. their row number with respect to the original data frame).
- repIdx is a vector of integers indicating the indexes for repaired/relabelled instances (i.e. their row number with respect to the original data frame).
- repLab is a factor containing the new labels for repaired instances.
- parameters is a list containing the argument values.
- call contains the original call to the filter.
- extraInf is a character that includes additional interesting information not covered by previous items.

References

Tomek I. (Nov. 1976): Two modifications of CNN, *IEEE Trans. Syst., Man, Cybern.*, vol. 6, no. 11, pp. 769-772.

Lorena A. C., Batista G. E. A. P. A., de Carvalho A. C. P. L. F., Monard M. C. (Nov. 2002): The influence of noisy patterns in the performance of learning methods in the splice junction recognition problem, in *Proc. 7th Brazilian Symp. Neural Netw.*, Recife, Brazil, pp. 31-37.

Examples

```
# Next code fails since TomekLinks method is designed for two-class problems.
# Some decomposition strategy like OVO or OVA could be used to overcome this.
## Not run:
data(iris)
out <- TomekLinks(Species~., data = iris)

## End(Not run)
```

Index