Package 'ROSE'

July 15, 2014

Type Package

Title ROSE: Random Over-Sampling Examples

Version 0.0-3

Date 2014-01-30

Author Nicola Lunardon, Giovanna Menardi, Nicola Torelli

Maintainer Nicola Lunardon <lunardon@stat.unipd.it>

Suggests MASS, nnet, rpart, tree

Description The package provides functions to deal with binary classification problems in the presence of imbalanced classes. Synthetic balanced samples are generated according to ROSE (Menardi and Torelli, 2013).
Functions that implement more traditional remedies to the class imbalance are also provided, as well as different metrics to evaluate a learner accuracy. These are estimated by holdout, bootstrap or cross-validation methods.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2014-07-15 18:41:03

R topics documented:

ROSE-package *ROSE: Random Over-Sampling Examples*

Description

The package provides functions to deal with binary classification problems in the presence of imbalanced classes. Synthetic balanced samples are generated according to ROSE (Menardi and Torelli, 2014). Functions that implement more traditional remedies to the class imbalance are also provided, as well as different metrics to evaluate a learner accuracy. These are estimated by holdout, bootrstrap or cross-validation methods.

Details

The package pivots on function `ROSE` which generates synthetic balanced samples and thus allows to strenghten the subsequent estimation of any binary classifier. ROSE (Random Over-Sampling Examples) is a bootstrap-based technique which aids the task of binary classification in the presence of rare classes. It handles both continuous and categorical data by generating synthetic examples from a conditional density estimate of the two classes. Different metrics to evaluate a learner accuracy are supplied by functions `roc.curve` and `accuracy.meas`. Holdout, bootstrap or cross-validation estimators of these accuracy metrics are computed by means of ROSE and provided by function `ROSE.eval`, to be used in conjuction with virtually any binary classifier. Additionally, function `ovun.sample` implements more traditional remedies to the class imbalance, such as over-sampling the minority class, under-sampling the majority class, or a combination of over- and under- sampling.

Author(s)

Nicola Lunardon, Giovanna Menardi, Nicola Torelli

Maintainer: Nicola Lunardon <lunardon@stat.unipd.it>

References

Lunardon, N., Menardi, G., and Torelli, N. (2014). ROSE: a Package for Binary Imbalanced Learning. *R Jorunal*, 6:82–92.

Menardi, G. and Torelli, N. (2014). Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery*, 28:92–122.

See Also

`DMwR-package`, `nnet`, `pROC-package`, `rpart`

Examples

```
# loading data
data(hacide)

# check imbalance
```

```
table(hacide.train$cls)

# train logistic regression on imbalanced data
log.reg.imb <- glm(cls ~ ., data=hacide.train, family=binomial)

# use the trained model to predict test data
pred.log.reg.imb <- predict(log.reg.imb, newdata=hacide.test,
                            type="response")

# generate new balanced data by ROSE
hacide.rose <- ROSE(cls ~ ., data=hacide.train, seed=123)$data

# check (im)balance of new data
table(hacide.rose$cls)

# train logistic regression on balanced data
log.reg.bal <- glm(cls ~ ., data=hacide.rose, family=binomial)

# use the trained model to predict test data
pred.log.reg.bal <- predict(log.reg.bal, newdata=hacide.test,
                            type="response")

# check accuracy of the two learners by measuring auc
roc.curve(hacide.test$cls, pred.log.reg.imb)
roc.curve(hacide.test$cls, pred.log.reg.bal, add.roc=TRUE, col=2)

# determine bootstrap distribution of the AUC of logit models
# trained on ROSE balanced samples
# B has been reduced from 100 to 10 for time saving solely
boot.auc.bal <- ROSE.eval(cls ~ ., data=hacide.train, learner= glm,
                          method.assess = "BOOT",
                          control.learner=list(family=binomial),
                          trace=TRUE, B=10)

summary(boot.auc.bal)
```

`accuracy.meas` *Metrics to evaluate a classifier accuracy in imbalanced learning*

Description

This function computes precision, recall and the F measure of a prediction.

Usage

```
accuracy.meas(response, predicted, threshold = 0.5)
```

Arguments

response	A vector of responses containing two classes to be used to evaluate prediction accuracy. It can be of class "factor", "numeric" or "character".
predicted	A vector containing a prediction for each observation. This can be of class "factor" or "character" if the predicted label classes are provided or "numeric" for the probabilities of the rare class (or a monotonic function of them).
threshold	When predicted is of class numeric, it defines the probability threshold to classify an example as positive. Default value is meant for predicted probabilities and is set to 0.5. See further details below. Ignored if predicted is of class factor

Details

Prediction of positive or negative labels depends on the classification threshold, here defined as the value such that observations with predicted value greater than the threshold are assigned to the positive class. Some caution is due in setting the threshold as well as in using the default setting both because the default value is meant for predicted probabilities and because the default 0.5 is not necessarily the optimal choice for imbalanced learning. Smaller values set for the threshold correspond to assign a larger misclassification costs to the rare class, which is usually the case.

Precision is defined as follows:

$$\frac{\text{true positives}}{\text{true positives + false positives}}$$

Recall is defined as:

$$\frac{\text{true positives}}{\text{true positives + false negative}}$$

The F measure is the harmonic average between precision and recall:

$$2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision+recall}}$$

Value

The value is an object of class accuracy.meas which has components

Call	The matched call.
threshold	The selected threshold.
precision	A vector of length one giving the precision of the prediction
recall	A vector of length one giving the recall of the prediction
F	A vector of length one giving the F measure

References

Fawcet T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27 (8), 861–875.

See Also

roc.curve

Examples

```
# 2-dimensional example
# loading data
data(hacide)

# imbalance on training set
table(hacide.train$cls)

# model estimation using logistic regression
fit.hacide  <- glm(cls~., data=hacide.train, family="binomial")

# prediction on training set
pred.hacide.train <- predict(fit.hacide, newdata=hacide.train,
                             type="response")

# compute accuracy measures (training set)
accuracy.meas(hacide.train$cls, pred.hacide.train, threshold = 0.02)

# imbalance on test set
table(hacide.test$cls)

# prediction on test set
pred.hacide.test <- predict(fit.hacide, newdata=hacide.test,
                            type="response")

# compute accuracy measures (test set)
accuracy.meas(hacide.test$cls, pred.hacide.test, threshold = 0.02)
```

---

hacide *Half circle filled data*

Description

Simulated training and test set for imbalanced binary classification. The rare class may be described as a half circle depleted filled with the prevalent class, which is normally distributed and has elliptical contours.

Usage

```
data(hacide)
```

Format

Data represent 2 real features (denoted as `x1, x2`) and a binary label class (denoted as `cls`). Positive examples occur in about 2% of the data.

`hacide.train` Includes 1000 rows and 20 positive examples.

`hacide.test` Includes 250 rows and 5 positive examples.

Data have been simulated as follows:

- if `cls` = 0 then `(x1, x2)` $\sim \mathbf{N}_2\left(\mathbf{0}_2, (1/4, 1)\mathbf{I}_2\right)$
- if `cls` = 1 then `(x1, x2)` $\sim \mathbf{N}_2\left(\mathbf{0}_2, \mathbf{I}_2\right) \cap \|\mathbf{x}\|^2 > 4 \cap x_2 \leq 0$

References

Lunardon, N., Menardi, G., and Torelli, N. (2014). ROSE: a Package for Binary Imbalanced Learning. *R Jorunal*, 6:82–92.

Menardi, G. and Torelli, N. (2014). Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery*, 28:92–122.

Examples

```
data(hacide)
summary(hacide.train)
summary(hacide.test)
```

`ovun.sample` *Over-sampling, under-sampling, combination of over- and under-sampling.*

Description

Creates possibly balanced samples by random over-sampling minority examples, under-sampling majority examples or combination of over- and under-sampling.

Usage

```
ovun.sample(formula, data, method="both", N, p=0.5,
            subset=options("subset")$subset,
            na.action=options("na.action")$na.action, seed)
```

Arguments

`formula`	An object of class `formula` (or one that can be coerced to that class). See `ROSE` for information about interaction among predictors or their transformations.
`data`	An optional data frame, list or environment (or object coercible to a data frame by `as.data.frame`) in which to preferentially interpret "formula". If not specified, the variables are taken from "environment(formula)".
`method`	One among `c("over", "under", "both")` to perform over-sampling minority examples, under-sampling majority examples or combination of over- and under-sampling, respectively.
`N`	The desired sample size of the resulting data set. If missing and `method` is either `"over"` or `"under"` the sample size is determined by oversampling or, respectively, undersampling examples so that the minority class occurs approximately in proportion `p`. When `method = "both"` the default value is given by the length of vectors specified in `formula`.

p	The probability of resampling from the rare class. If missing and method is either "over" or "under" this proportion is determined by oversampling or, respectively, undersampling examples so that the sample size is equal to N. When method ="both" the default value given by 0.5.
subset	An optional vector specifying a subset of observations to be used in the sampling process. The default is set by the subset setting of options.
na.action	A function which indicates what should happen when the data contain 'NA's. The default is set by the na.action setting of options.
seed	A single value, interpreted as an integer, recommended to specify seeds and keep trace of the sample.

Value

The value is an object of class ovun.sample which has components

Call	The matched call.
method	The method used to balance the sample. Possible choices are c("over", "under", "both").
data	The resulting new data set.

See Also

ROSE.

Examples

```
# 2-dimensional example
# loading data
data(hacide)

# imbalance on training set
table(hacide.train$cls)

# balanced data set with both over and under sampling
data.balanced.ou <- ovun.sample(cls~., data=hacide.train,
                                N=nrow(hacide.train), p=0.5,
                                seed=1, method="both")$data

table(data.balanced.ou$cls)

# balanced data set with over-sampling
data.balanced.over <- ovun.sample(cls~., data=hacide.train,
                                  p=0.5, seed=1,
                                  method="over")$data

table(data.balanced.over$cls)
```

`roc.curve` *ROC curve*

Description

This function returns the ROC curve and computes the area under the curve (AUC) for binary classifiers.

Usage

```
roc.curve(response, predicted, plotit = TRUE, add.roc = FALSE,
          n.thresholds=100, ...)
```

Arguments

`response`	A vector of responses containing two classes to be used to compute the ROC curve. It can be of class `"factor"`, `"numeric"` or `"character"`.
`predicted`	A vector containing a prediction for each observation. This can be of class `"factor"` or `"character"` if the predicted label classes are provided or `"numeric"` for the probabilities of the rare class (or a monotonic function of them).
`plotit`	Logical, if `TRUE` the ROC curve is plotted in a new window. Default value is set to `TRUE`.
`add.roc`	Logical, if `TRUE` the ROC curve is added to an existing window. Default value is set to `FALSE`.
`n.thresholds`	Number of `thresholds` at which the ROC curve is computed. Default value is the minimum between 100 and the number of elements in `response`. A value of `n.thresholds` greater than the length of `response` is ignored.
`...`	Further arguments to be passed either to `plot` or `lines`.

Value

The value is an object of class `roc.curve` which has components

`Call`	The matched call.
`auc`	The value of the area under the ROC curve.
`false positive rate`	The false positive rate (or equivalently the complement of sensitivity) of the classifier at the evaluated `thresholds`.
`true positive rate`	The true positive rate (or equivalently the specificity) of the classifier at the evaluated `thresholds`.
`thresholds`	Thresholds at which the ROC curve is evaluated.

References

Fawcet T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27 (8), 861–875.

See Also

accuracy.meas, roc.

Examples

```
# 2-dimensional example
# loading data
data(hacide)

# check imbalance on training set
table(hacide.train$cls)

# model estimation using logistic regression
fit.hacide  <- glm(cls~., data=hacide.train, family="binomial")

# prediction on training set
pred.hacide.train <- predict(fit.hacide, newdata=hacide.train)

# plot the ROC curve (training set)
roc.curve(hacide.train$cls, pred.hacide.train,
          main="ROC curve \n (Half circle depleted data)")

# check imbalance on test set
table(hacide.test$cls)

# prediction using test set
pred.hacide.test <- predict(fit.hacide, newdata=hacide.test)

# add the ROC curve (test set)
roc.curve(hacide.test$cls, pred.hacide.test, add=TRUE, col=2,
          lwd=2, lty=2)
legend("topleft", c("Resubstitution estimate", "Holdout estimate"),
        col=1:2, lty=1:2, lwd=2)
```

ROSE *Generation of synthetic data by Randomly Over Sampling Examples (ROSE)*

Description

Creates a sample of synthetic data by enlarging the features space of minority and majority class examples. Operationally, the new examples are drawn from a conditional kernel density estimate of the two classes, as described in Menardi and Torelli (2013).

Usage

```
ROSE(formula, data, N, p=0.5, hmult.majo=1, hmult.mino=1,
     subset=options("subset")$subset,
     na.action=options("na.action")$na.action, seed)
```

Arguments

formula	An object of class formula (or one that can be coerced to that class). The left-hand-side (response) should be a vector specifying the class labels. The right-hand-side should be a series of vectors with the predictors. See "Warning" for information about interaction among predictors or their transformations.
data	An optional data frame, list or environment (or object coercible to a data frame by as.data.frame) in which to preferentially interpret "formula". If not specified, the variables are taken from "environment(formula)".
N	The desired sample size of the resulting data set generated by ROSE. If missing, it is set equal to the length of the response variable in formula.
p	The probability of the minority class examples in the resulting data set generated by ROSE.
hmult.majo	Optional shrink factor to be multiplied by the smoothing parameters to estimate the conditional kernel density of the majority class. See "References" and "Details".
hmult.mino	Optional shrink factor to be multiplied by the smoothing parameters to estimate the conditional kernel density of the minority class. See "References" and "Details".
subset	An optional vector specifying a subset of observations to be used in the sampling process. The default is set by the subset setting of options.
na.action	A function which indicates what should happen when the data contain 'NA's. The default is set by the na.action setting of options.
seed	A single value, interpreted as an integer, recommended to specify seeds and keep trace of the generated sample.

Details

ROSE (Random Over-Sampling Examples) aids the task of binary classification in the presence of rare classes. It produces a synthetic, possibly balanced, sample of data simulated according to a smoothed-bootstrap approach.

Denoted by y the binary response and by x a vector of numeric predictors observed on n subjects i, $(i = 1, \ldots, n)$, syntethic examples with class label $k, (k = 0, 1)$ are generated from a kernel estimate of the conditional density $f(x|y = k)$. The kernel is a Normal product function centered at each of the x_i with diagonal covariance matrix H_k. Here, H_k is the asymptotically optimal smoothing matrix under the assumption of multivariate normality. See "References" below and further references therein.

Essentially, ROSE selects an observation belonging to the class k and generates new examples in its neighbourhood, where the width of the neighbourhood is determined by H_k. The user is allowed

to shrink H_k by varying arguments `h.mult.majo` and `h.mult.mino`. Balancement is regulated by argument `p`, i.e. the probability of generating examples from class $k = 1$.

As they stand, kernel-based methods may be applied to continuous data only. However, as ROSE includes combination of over and under-sampling as a special case when H_k tend to zero, the assumption of continuity may be circumvented by using a degenerate kernel distribution to draw synthetic categorical examples. Basically, if the $j-$th component of x_i is categorical, a syntehic clone of x_i will have as $j-$th component the same value of the $j-$th component of x_i.

Value

The value is an object of class `ROSE` which has components

`Call`	The matched call.
`method`	The method used to balance the sample. The only possible choice is `ROSE`.
`data`	An object of class `data.frame` containing new examples generated by ROSE.

Warning

The purpose of `ROSE` is to generate new synthetic examples in the features space. The use of `formula` is intended solely to distinguish the response variable from the predictors. Hence, `formula` must not be confused with the one supplied to fit a classifier in which the specification of either tranformations or interactions among variables may be sensible/necessary. In the current version `ROSE` discards possible interactions and transformations of predictors specified in `formula` automatically. The automatic parsing of `formula` is able to manage virtually all cases on which it has been tested it but the user is warned to use caution in the specification of entangled functions of predictors. Any report about possible malfunctioning of the parsing mechanism is welcome.

References

Lunardon, N., Menardi, G., and Torelli, N. (2014). ROSE: a Package for Binary Imbalanced Learning. *R Jorunal*, 6:82–92.

Menardi, G. and Torelli, N. (2014). Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery*, 28:92–122.

See Also

`ovun.sample`, `ROSE.eval`.

Examples

```
# 2-dimensional example
# loading data
data(hacide)

# imbalance on training set
table(hacide.train$cls)
#imbalance on test set
table(hacide.test$cls)
```

```
# plot unbalanced data highlighting the majority and
# minority class examples.
par(mfrow=c(1,2))
plot(hacide.train[, 2:3], main="Unbalanced data", xlim=c(-4,4),
     ylim=c(-4,4), col=as.numeric(hacide.train$cls), pch=20)
legend("topleft", c("Majority class","Minority class"), pch=20, col=1:2)

# model estimation using logistic regression
fit <- glm(cls~., data=hacide.train, family="binomial")
# prediction using test set
pred <- predict(fit, newdata=hacide.test)
roc.curve(hacide.test$cls, pred,
          main="ROC curve \n (Half circle depleted data)")

# generating data according to ROSE: p=0.5 as default
data.rose <- ROSE(cls~., data=hacide.train, seed=3)$data
table(data.rose$cls)

par(mfrow=c(1,2))
# plot new data generated by ROSE highlighting the
# majority and minority class examples.
plot(data.rose[, 2:3], main="Balanced data by ROSE",
     xlim=c(-6,6), ylim=c(-6,6), col=as.numeric(data.rose$cls), pch=20)
legend("topleft", c("Majority class","Minority class"), pch=20, col=1:2)

fit.rose <- glm(cls~., data=data.rose, family="binomial")
pred.rose <- predict(fit.rose, data=data.rose, type="response")
roc.curve(data.rose$cls, pred.rose,
          main="ROC curve \n (Half circle depleted data balanced by ROSE)")
par(mfrow=c(1,1))
```

ROSE.eval *Evaluation of learner accuracy by ROSE*

Description

Given a classifier and a set of data, this function exploits ROSE generation of synthetic samples to provide holdout, bootstrap or leave-K-out cross-validation estimates of a specified accuracy measure.

Usage

```
ROSE.eval(formula, data, learner, acc.measure="auc", extr.pred=NULL,
          method.assess="holdout", K=1, B=100, control.rose=list(),
          control.learner=list(), control.predict=list(),
          control.accuracy=list(), trace=FALSE,
          subset=options("subset")$subset,
          na.action=options("na.action")$na.action, seed)
```

Arguments

formula	An object of class formula (or one that can be coerced to that class). The specification of the formula must be suited for the selected classifier. See ROSE and the "Note" below for information about interaction among predictors or their transformations.
data	An optional data frame, list or environment (or object coercible to a data frame by as.data.frame) in which to preferentially interpret "formula". If not specified, the variables are taken from "environment(formula)".
learner	Either a built-in **R** or an user defined function that fits a classifier and that returns a vector of predicted values. See "Details" below.
acc.measure	One among c("auc", "precision", "recall", "F"), it defines the accuracy measure to be estimated. Function roc.curve is internally called when auc="auc" while the other options entail an internal call of function accuracy.meas. Default value is "auc".
extr.pred	An optional function that extracts from the output of a predict function the vector of predicted values. If not specified, the value returned by "predict" is used. See Examples below.
method.assess	One among c("holdout", "LKOCV", "BOOT"), it is the method used for model assessment. When "holdout" is chosen, the learner is fitted on one ROSE sample and tested on the data provided in formula. "LKOCV" stands for "leave-K-out cross validation": the original data set is divided into Q subsets of K observations; at each round, the specified learner is estimated on a ROSE sample built on the provided data but one of these groups and then a prediction on the excluded set of observations is made. At the end of the process, the Q distinct predictions are deployed to compute the selected accuracy measure. "BOOT" estimates the accuracy measure by fitting a learner on B ROSE samples and testing each of them on the provided data.
K	An integer value indicating the size of the subsets created when method.assess="LKOCV". If K is not a multiple of the sample size n, then $Q-1$ sets of size K are created and the remaining $n-(Q-1)K$ observations are used to form the last subset. Default value is 1, i.e. leave-1-out cross validation is performed.
B	The number of bootstrap replications to set when method.assess="BOOT". Ignored otherwise. Default value is 100.
control.learner	Further arguments to be passed to learner
control.rose	Optional arguments to be passed to ROSE.
control.predict	Further arguments to be passed to predict.
control.accuracy	Optional arguments to be passed to either roc.curve or accuracy.meas depending on the selected accuracy measure.
trace	logical, if TRUE traces information on the progress of model assessment (number of bootstrap or cross validation iterations performed).

subset	An optional vector specifying a subset of observations to be used in the sampling and learning process. The default is set by the subset setting of options.
na.action	A function which indicates what should happen when the data contain 'NA's. The default is set by the na.action setting of options.
seed	A single value, interpreted as an integer, recommended to specify seeds and keep trace of the generated ROSE sample/es.

Details

This function estimates a measure of accuracy of a classifier specified by the user by using either holdout, cross-validation, or bootstrap estimators. Operationally, the classifier is trained over synthetic data generated by ROSE and then evaluated on the original data.

Whatever accuracy measure and estimator are chosen, the *true* accuracy depends on the probability distribution underlying the training data. This is clearly affected by the imbalance and its estimation is then regulated by argument control.rose. A default setting of the arguments (that is, p=0.5) entails the estimation of the learner accuracy conditional to a balanced training set. In order to estimate the accuracy of a learner fitted on unbalanced data, the user may set argument p of control.rose to the proportion of positive examples in the observed sample. See Example 2 below and, for further details, Menardi and Torelli (2014).

To the aim of a grater flexibility, ROSE.eval is not linked to the use of a specific learner and works virtually with any classifier. The actual implementation supports the following two type of learner.

In the first case, learner has a 'standard' behavior in the sense that it is a function having formula as a mandatory argument and retrieves an object whose class is associated to a predict method. The user that is willing to define her/his own learner must follow the implicit convention that when a classed object is created, then the function name and the class should match (such as lm, glm, rpart, tree, nnet, lda, etc). Furthermore, since predict returns are very heterogeneous, the user is allowed to define some function extr.pred which extracts from the output of predict the desired vector of predicted values.

In the second case, learner is a wrapper that allows to embed functions that do not meet the aforementioned requirements. The wrapper must have the following mandatory arguments: data and newdata, and must return a vector of predicted values. Optional arguments can be passed as well into the wrapper including the ... and by specifiyng them through control.learner. When argument data in ROSE.eval is not missing, data in learner receives a data frame structured as the one in input, otherwise it is constructed according to the template provided by formula. The same rule applies for argument newdata with the exception that the class label variable is dropped. See "Examples" below.

Value

The value is an object of class ROSE.eval which has components

Call	The matched call.
method	The selected method for model assessment.
measure	The selected measure to evaluate accuracy.
acc	The vector of the estimated measure of accuracy. It has length 1 if method.assess="holdout", or method.assess="LKOCV" and length B if method.assess="BOOT", corresponding to the bootstrap distribution of the accuracy estimator.

Note

The function allows the user to include in the formula transformations of predictors or interactions among them. ROSE samples are generated on the original data and transformations or interactions are ignored. These are then retrieved in fitting the classifier, provided that the selected learner function can handle them. See also "Warning" in ROSE.

References

Lunardon, N., Menardi, G., and Torelli, N. (2014). ROSE: a Package for Binary Imbalanced Learning. *R Jorunal*, 6:82–92.

Menardi, G. and Torelli, N. (2014). Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery*, 28:92–122.

See Also

ROSE, roc.curve, accuracy.meas.

Examples

```
# 2-dimensional data
# loading data
data(hacide)

# in the following examples
# use of a small subset of observations only --> argument subset

dat <- hacide.train

table(dat$cls)

##Example 1
# classification with logit model
# arguments to glm are passed through control.learner
# leave-one-out cross-validation estimate of auc of classifier
# trained on balanced data
ROSE.eval(cls~., data=dat, glm, subset=c(1:50, 981:1000),
          method.assess="LKOCV", K=5,
          control.learner=list(family=binomial), seed=1)

## Not run:
##Example 2
# classification with decision tree
# require package rpart
library(rpart)

# function is needed to extract predicted probability of cls 1
f.pred.rpart <- function(x) x[,2]

# holdout estimate of auc of two classifiers
```

```
# first classifier trained on ROSE unbalanced sample
# proportion of rare events in original data
p <- (table(dat$cls)/sum(table(dat$cls)))[2]
ROSE.eval(cls~., data=dat, rpart, subset=c(1:50, 981:1000),
          control.rose=list(p = p), extr.pred=f.pred.rpart, seed=1)

# second classifier trained on ROSE balanced sample
# optional arguments to plot the roc.curve are passed through
# control.accuracy
ROSE.eval(cls~., data=dat, rpart, subset=c(1:50, 981:1000),
          control.rose=list(p = 0.5), control.accuracy = list(add.roc = TRUE,
          col = 2), extr.pred=f.pred.rpart, seed=1)

##Example 3
# classification with linear discriminant analysis
library(MASS)

# function is needed to extract the predicted values from predict.lda
f.pred.lda <- function(z) z$posterior[,2]

# bootstrap estimate of precision of learner trained on balanced data
prec.distr <- ROSE.eval(cls~., data=dat, lda, subset=c(1:50, 981:1000),
                        extr.pred=f.pred.lda, acc.measure="precision",
                        method.assess="BOOT", B=100, trace=TRUE)

summary(prec.distr)

##Example 4
# compare auc of classification with neural network
# with auc of classification with tree
# require package nnet
# require package tree

library(nnet)
library(tree)

# optional arguments to nnet are passed through control.learner
ROSE.eval(cls~., data=dat, nnet, subset=c(1:50, 981:1000),
          method.assess="holdout", control.learn=list(size=1), seed=1)

# optional arguments to plot the roc.curve are passed through
# control.accuracy
# a function is needed to extract predicted probability of class 1
f.pred.rpart <- function(x) x[,2]
f.pred.tree  <- function(x) x[,2]
ROSE.eval(cls~., data=dat, tree, subset=c(1:50, 981:1000),
          method.assess="holdout", extr.pred=f.pred.tree,
          control.acc=list(add=TRUE, col=2), seed=1)

##Example 5
# An user defined learner with a standard behavior
# Consider a dummy example for illustrative purposes only
# Note that function name and the name of the class returned match
```

```
DummyStump <- function(formula, ...)
{
   mc <- match.call()
   m <- match(c("formula", "data", "na.action", "subset"), names(mc), 0L)
   mf <- mc[c(1L, m)]
   mf[[1L]] <- as.name("model.frame")
   mf <- eval(mf, parent.frame())
   data.st <- data.frame(mf)
   out <- list(colname=colnames(data.st)[2], threshold=1)
   class(out) <- "DummyStump"
   out
}

# Associate to DummyStump a predict method
# Usual S3 definition: predic.classname
predict.DummyStump <- function(object, newdata)
{
   out <- newdata[,object$colname]>object$threshold
   out
}

ROSE.eval(formula=cls~., data=dat, learner=DummyStump,
          subset=c(1:50, 981:1000), method.assess="holdout", seed=3)


##Example 6
# The use of the wrapper for a function with non standard behaviour
# Consider knn in package class
# require package class

library(class)

# the wrapper require two mandatory arguments: data, newdata.
# optional arguments can be passed by including the object '...'
# note that we are going to specify data=data in ROSE.eval
# therefore data in knn.wrap will receive a data set structured
# as dat as well as newdata but with the class label variable dropped
# note that inside the wrapper we dispense to knn
# the needed quantities accordingly

knn.wrap <- function(data, newdata, ...)
{
   knn(train=data[,-1], test=newdata, cl=data[,1], ...)
}

# optional arguments to knn.wrap may be specified in control.learner
ROSE.eval(formula=cls~., data=dat, learner=knn.wrap,
          subset=c(1:50, 981:1000), method.assess="holdout",
          control.learner=list(k=2, prob=T), seed=1)

# if we swap the columns of dat we have to change the wrapper accordingly
dat <- dat[,c("x1","x2","cls")]
```

```
# now class label variable is the last one
knn.wrap <- function(data, newdata, ...)
{
   knn(train=data[,-3], test=newdata, cl=data[,3], ...)
}

ROSE.eval(formula=cls~., data=dat, learner=knn.wrap,
          subset=c(1:50, 981:1000), method.assess="holdout",
          control.learner=list(k=2, prob=T), seed=1)


## End(Not run)
```

Index