Package: imbalanceKit (via r-universe)

November 14, 2026

Type Package

Title Resampling Methods and Tools for Imbalanced Classification

Version 0.2.0

Description A collection of resampling methods for class imbalance in classification problems. Includes the oversampling methods SMOTE with boosting (SMOTEWB) by F. Sağlam and M. A. Cengiz (2022) <doi:10.1016/j.eswa.2022.117023>, Markov chain SMOTE (MCSMOTE) by F. Sağlam and M. A. Cengiz (2025) <doi:10.1016/j.asoc.2025.113906>, uncertainty guided oversampling (UGO) by F. Sağlam and M. A. Cengiz (2026) <doi:10.1016/j.knosys.2026.115886>, and bi-directional SMOTE (BiSMOTE) by F. Sağlam and U. Yamak (2027) <doi:10.1016/j.eswa.2026.134648>, and the data relocation approach DatRel by F. Sağlam (2025) <doi:10.1007/s10994-025-06755-8>, which can also be applied to synthetic samples from any other source, together with commonly used alternatives such as SMOTE, ADASYN, Borderline-SMOTE, Safe-Level-SMOTE, Radius-SMOTE, VS-SMOTE, geometric SMOTE, random oversampling and undersampling. The methods share the same interface: they take a feature matrix and a factor class variable and return the resampled data together with the synthetic samples. The package also provides performance metrics for imbalanced classification and data complexity measures of class imbalance, class overlap and noise.

Depends R (>= 4.0.0)

Imports stats, utils, FNN, rpart, Rfast, Rcpp

LinkingTo Rcpp

Suggests randomForest, nnet, testthat (>= 3.0.0), knitr, rmarkdown

License MIT + file LICENSE

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Config/testthat/edition 3

URL https://fatihsaglam.github.io/imbalanceKit/

Config/roxygen2/version 8.1.1

Config/pak/sysreqs cmake

Repository https://fatihsaglam.r-universe.dev

Date/Publication 2026-11-13 19:46:10 UTC

RemoteUrl https://github.com/fatihsaglam/imbalancekit

RemoteRef HEAD

RemoteSha b3a71b9710cde3e3938f354c5d8652336ade5b09

Contents

ADASYN . . . 3
banana . . . 4
BiSMOTE . . . 5
BLSMOTE . . . 6
complexity . . . 8
complexity_measures . . . 9
DatRel . . . 12
EBSMOTE . . . 14
evaluate . . . 16
GQEO . . . 17
GSMOTE . . . 18
LEAO . . . 20
MCSMOTE . . . 22
metrics . . . 23
oversampleDatRel . . . 26
RadiusSMOTE . . . 28
ROS . . . 29
ROSE . . . 31
RSLSMOTE . . . 32
RUS . . . 34
RWO . . . 35
SLSMOTE . . . 36
SMOTE . . . 38
SMOTEWB . . . 39
SyMProD . . . 41
UGO . . . 42
VSSMOTE . . . 45

Index **48**

ADASYN *Adaptive Synthetic Sampling (ADASYN, 2008)*

Description

Generates synthetic data for minority class to balance imbalanced datasets using ADASYN.

Usage

```
ADASYN(x, y, k = 5, ovRate = 1)
```

Arguments

x	feature matrix or data.frame.
y	a factor class variable. Can have more than two classes.
k	number of neighbors. Default is 5.
ovRate	Oversampling rate. Class j receives `round(ovRate * (n_max - n_j))` synthetic samples, where `n_max` is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.

Details

Year: 2008.

Adaptive Synthetic Sampling (ADASYN) is an extension of the Synthetic Minority Over-sampling Technique (SMOTE) algorithm, which is used to generate synthetic examples for the minority class (He et al., 2008). In contrast to SMOTE, ADASYN adaptively generates synthetic examples by focusing on the minority class examples that are harder to learn, meaning those examples that are closer to the decision boundary.

Can work with classes more than 2.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic data.
C	Number of synthetic samples for each positive class samples.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008, June). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence) (pp. 1322-1328). IEEE.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)

# resampling
m <- ADASYN(x = x, y = y, k = 3)

plot(m$x_new, col = m$y_new)
```

---

banana *Banana data*

Description

A two-dimensional, two-class imbalanced data set with an imbalance ratio of 1:9. The classes cannot be separated linearly but can be separated non-linearly, and both are multimodal. The data set contains class overlap and noise as well, so it shows many of the difficulties of imbalanced classification in a plot. It is used in the examples and the vignette of the package.

Usage

```
banana
```

Format

A data frame with 2640 rows and 3 variables:

At1 first feature.

At2 second feature.

Class factor with levels `"negative"` (2376 samples) and `"positive"` (264 samples). The imbalance ratio is 9.

Source

The banana data set of the KEEL data set repository, https://sci2s.ugr.es/keel/datasets.php.

References

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., & Herrera, F. (2011). KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing, 17(2-3), 255-287.

Examples

```
table(banana$Class)
plot(banana$At1, banana$At2, col = ifelse(banana$Class == "positive", "red", "grey60"),
     pch = 16, xlab = "At1", ylab = "At2")
```

BiSMOTE *Bi-directional SMOTE (BiSMOTE, 2027)*

Description

Resampling with bi-directional SMOTE.

Usage

```
BiSMOTE(x, y, k = 5, ovRate = 1, ...)
```

Arguments

x	feature matrix or dataframe.
y	a factor class variable. Can have more than two classes.
k	number of nearest neighbors within the class that define the interpolation directions. Default is 5. Reduced automatically for classes with fewer than k + 1 samples.
ovRate	Oversampling rate. Class j receives round(ovRate * (n_max - n_j)) synthetic samples, where n_max is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.
...	not used.

Details

Year: 2027.

BiSMOTE (Saglam & Yamak, 2027) is a geometrically constrained oversampling method. For each synthetic sample, a center sample and one of its k nearest neighbors from the same class are selected. Unlike SMOTE, which places the new sample between the two, BiSMOTE interpolates *bi-directionally* along the line through them. The interpolation coefficient is drawn uniformly so that the synthetic sample stays within the distance from the center sample to its nearest sample of another class. In this way, generation is restricted to a locally bounded region and does not cross towards other classes, which reduces class overlap and synthetic noise.

Can work with classes more than 2.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic feature data.
y_syn	Generated synthetic label data.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Saglam, F., & Yamak, U. (2027). Bi-SMOTE: bi-directional oversampling for reducing class overlap and synthetic noise. Expert Systems with Applications, 334, 134648.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)

# resampling
m <- BiSMOTE(x = x, y = y, k = 5)

plot(m$x_new, col = m$y_new)
```

BLSMOTE *Borderline-SMOTE (BLSMOTE, 2005)*

Description

BLSMOTE() applies BLSMOTE (Borderline-SMOTE) which is a variation of the SMOTE algorithm that generates synthetic samples only in the vicinity of the borderline instances in imbalanced datasets.

Usage

```
BLSMOTE(x, y, k1 = 5, k2 = 5, type = c("type1", "type2"), ovRate = 1)
```

Arguments

x	feature matrix or data.frame.
y	a factor class variable. Can have more than two classes.
k1	number of neighbors to link. Default is 5.
k2	number of neighbors to determine safe levels. Default is 5.
type	"type1" or "type2". Default is "type1".
ovRate	Oversampling rate. Class j receives `round(ovRate * (n_max - n_j))` synthetic samples, where `n_max` is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.

Details

Year: 2005.

BLSMOTE works by focusing on the instances that are near the decision boundary between the minority and majority classes, known as borderline instances. These instances are more informative and potentially more challenging for classification, and thus generating synthetic samples in their vicinity can be more effective than generating them randomly.

Can work with classes more than 2.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic data.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Han, H., Wang, W. Y., & Mao, B. H. (2005). Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In Advances in Intelligent Computing: International Conference on Intelligent Computing, ICIC 2005, Hefei, China, August 23-26, 2005, Proceedings, Part I 1 (pp. 878-887). Springer Berlin Heidelberg.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)
```

```
# resampling
m <- BLSMOTE(x = x, y = y, k1 = 5, k2 = 5)

plot(m$x_new, col = m$y_new)
```

`complexity` *Calculate Several Complexity Measures*

Description

Calculates the complexity measures of `complexity_measures` in one call.

Usage

```
complexity(
  x,
  y,
  measures = NULL,
  k_R = 7,
  theta = 3,
  k_type = 5,
  scale = "minmax",
  seed = NULL
)
```

Arguments

`x`	feature matrix or data.frame (numeric).
`y`	a factor class variable.
`measures`	character vector with the names of the measures, for example `c("IR", "R_aug", "N3")`. The default `NULL` calculates all of them (`"IR"`, `"C1"`, `"C2"`, `"ID"`, `"F1"`, `"F1v"`, `"F2"`, `"F3"`, `"F4"`, `"N1"`, `"N2"`, `"N3"`, `"N3_bal"`, `"N4"`, `"LSC"`, `"R"`, `"R_aug"`, `"SAFE"`, `"BORDERLINE"`, `"RARE"`, `"OUTLIER"`).
`k_R`	number of neighbors of `R` and `R_aug`. Default is 7.
`theta`	threshold of `R` and `R_aug`. Default is 3.
`k_type`	number of neighbors of the types of minority examples. Default is 5.
`scale, seed`	see `complexity_measures`.

Details

The distance matrix is calculated once and shared by the measures that need it, so asking for several measures at once is faster than calling them one by one. The measures that use all pairwise distances (`N1`, `N2`, `LSC`) need memory of the order of n^2.

Value

a named numeric vector.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

see complexity_measures.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(400, 3, 1), ncol = 2), matrix(rnorm(40, 4, 1), ncol = 2))
y <- factor(rep(c("negative", "positive"), c(200, 20)))
complexity(x, y)
complexity(x, y, measures = c("IR", "R", "R_aug", "N4"), seed = 1)
```

---

complexity_measures *Complexity Measures for Imbalanced Classification Data*

Description

Measures of the class imbalance, of the class overlap and of the noise of a classification data set, taken from the literature. Higher values mean a more imbalanced or a more complex (harder) problem. Each measure is a function of its own; complexity calculates all of them or the selected ones.

Usage

```
complexity_IR(x = NULL, y)

complexity_C1(x = NULL, y)

complexity_C2(x = NULL, y)

complexity_ID(x = NULL, y)

complexity_F1(x, y)

complexity_F1v(x, y)

complexity_F2(x, y)

complexity_F3(x, y)
```

```
complexity_F4(x, y)

complexity_N1(x, y, scale = "minmax")

complexity_N2(x, y, scale = "minmax")

complexity_N3(x, y, scale = "minmax")

complexity_N3_bal(x, y, scale = "minmax")

complexity_N4(x, y, scale = "minmax", seed = NULL)

complexity_LSC(x, y, scale = "minmax")

complexity_R(x, y, k = 7, theta = 3, scale = "minmax")

complexity_R_aug(x, y, k = 7, theta = 3, scale = "minmax")

complexity_SAFE(x, y, k = 5, scale = "minmax")

complexity_BORDERLINE(x, y, k = 5, scale = "minmax")

complexity_RARE(x, y, k = 5, scale = "minmax")

complexity_OUTLIER(x, y, k = 5, scale = "minmax")
```

Arguments

x	feature matrix or data.frame (numeric).
y	a factor class variable. Can have more than two classes (see Details).
scale	scaling of the features before distances are calculated: "minmax" (default, to [0, 1]), "zscore" or "none". Feature based measures (F1, F1v, F2, F3, F4) use the original features.
seed	seed of the random interpolation of N4. Default is NULL (the random number stream is not touched).
k	number of nearest neighbors. Default is 7 for R and R_aug (Oh, 2011) and 5 for the types of minority examples (Napierala & Stefanowski, 2016).
theta	threshold of the R-value: a sample is in the overlapping region if more than theta of its k nearest neighbors are from other classes. Default is 3.

Details

Class imbalance (they use y only; x may be NULL):

- IR: imbalance ratio, size of the largest class divided by the size of the smallest class.
- C1: $1 + \sum_k p_k \log p_k / \log K$, one minus the normalized entropy of the class proportions p_k (Lorena et al., 2019). 0 means balanced, 1 means one class has all the samples.

- C2: $1 - 1/IR_K$ with $IR_K = \frac{K-1}{K} \sum_k n_k/(n - n_k)$ (Lorena et al., 2019).
- ID: imbalance degree (Ortigosa-Hernandez et al., 2017), with the Euclidean distance. 0 means balanced; the integer part is the number of minority classes minus one.

Overlap, feature based (Ho & Basu, 2002; Lorena et al., 2019). They are defined for two classes; with more classes the mean over all pairs of classes (one-versus-one) is returned. They are transformed so that higher means more overlap:

- F1: $1/(1 + r)$, with r the maximum Fisher's discriminant ratio over the features. F1v: the same with the directional (vector) Fisher's ratio (Orriols-Puig et al., 2010).
- F2: volume of the overlapping region (product over the features of the relative overlap).
- F3: maximum individual feature efficiency, inverted: the smallest proportion of samples in the overlapping interval of a single feature.
- F4: collective feature efficiency: proportion of samples that remain after the features have been used one after another to separate the classes.

Overlap, neighborhood based (they work with any number of classes; Ho & Basu, 2002; Leyva et al., 2015):

- N1: proportion of samples that are connected to a sample of another class in the minimum spanning tree.
- N2: ratio of the intra-class to the inter-class nearest neighbor distances, $r/(1 + r)$.
- N3: error rate of the leave-one-out 1-nearest neighbor classifier. N3_bal: the mean of its class-wise error rates (a class-balanced variant, not in Ho & Basu).
- N4: error rate of the 1-nearest neighbor classifier on randomly interpolated samples (of the same class) (needs a random number stream, see seed).
- LSC: local set cardinality, $1 - \sum |LS|/n^2$.

Overlap and imbalance together:

- R: R-value (Oh, 2011), the proportion of samples in the overlapping region ($> theta$ of the k nearest neighbors from another class). For two classes it is $(IR \cdot R_{maj} + R_{min})/(IR + 1)$, so it is dominated by the majority class.
- R_aug: augmented R-value (Borsos et al., 2018), $(R_{maj} + IR \cdot R_{min})/(IR + 1)$: the weights are swapped so that the minority class, which is more affected by overlap, is dominant. For more than two classes the class weights are n_{max}/n_k (normalized); this generalization is ours.

Noise, types of minority examples (Napierala & Stefanowski, 2016). The minority samples (all classes except the largest one) are labelled by the proportion s of their k nearest neighbors from their own class: safe ($s \geq 0.8$), borderline ($0.4 \leq s < 0.8$), rare ($0 < s < 0.4$) and outlier ($s = 0$). With k = 5 these are 4-5, 2-3, 1 and 0 neighbors. The measures SAFE, BORDERLINE, RARE and OUTLIER are the proportions of minority samples of each type. Larger RARE and OUTLIER mean a noisier minority class.

Value

a number.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Borsos, Z., Lemnaru, C., & Potolea, R. (2018). Dealing with overlap and imbalance: a new metric and approach. Pattern Analysis and Applications, 21(2), 381-395.

Ho, T. K., & Basu, M. (2002). Complexity measures of supervised classification problems. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(3), 289-300.

Leyva, E., Gonzalez, A., & Perez, R. (2015). A set of complexity measures designed for applying meta-learning to instance selection. IEEE Transactions on Knowledge and Data Engineering, 27(2), 354-367.

Lorena, A. C., Garcia, L. P., Lehmann, J., Souto, M. C., & Ho, T. K. (2019). How complex is your classification problem? A survey on measuring classification complexity. ACM Computing Surveys, 52(5), 1-34.

Napierala, K., & Stefanowski, J. (2016). Types of minority class examples and their influence on learning classifiers in imbalanced domains. Journal of Intelligent Information Systems, 46(3), 563-597.

Oh, S. (2011). A new dataset evaluation method based on category overlap. Computers in Biology and Medicine, 41(2), 115-122.

Orriols-Puig, A., Macia, N., & Ho, T. K. (2010). Documentation for the data complexity library in C++. Technical report, La Salle, Universitat Ramon Llull.

Ortigosa-Hernandez, J., Inza, I., & Lozano, J. A. (2017). Measuring the class-imbalance extent of multi-class problems. Pattern Recognition Letters, 98, 32-38.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(400, 3, 1), ncol = 2), matrix(rnorm(40, 4, 1), ncol = 2))
y <- factor(rep(c("negative", "positive"), c(200, 20)))

complexity_IR(y = y)
complexity_R_aug(x, y)
complexity_N3(x, y)
```

`DatRel` *Data Relocation (DatRel, 2025)*

Description

DatRel (Saglam, 2025) is a post-processing method for synthetic samples. Synthetic samples that fall outside the region covered by their class are moved back into it. The synthetic samples can come from any oversampling method; `DatRel` only needs the original data and the synthetic data.

Usage

```
DatRel(x, y, x_syn, y_syn, proportion = 1, p_of = 0)
```

Arguments

x	feature matrix of the original data.
y	a factor class variable of the original data. Can have more than two classes.
x_syn	feature matrix of the synthetic samples.
y_syn	factor (or vector) with the class of each synthetic sample. Every class is relocated against the union of the other classes (one-versus-rest), so any number of classes is allowed.
proportion	proportion of the samples of a class that has to be covered by the dominating balls. Default is 1 (all samples).
p_of	overflow proportion: the radii of the balls are multiplied by 1 + p_of, so the synthetic samples may stay slightly outside of the class region. Default is 0.

Details

Year: 2025. For every class a ball is placed around each sample whose radius is the distance to the nearest sample of the other classes (times 1 + p_of), so that the ball contains no sample of another class. A small set of dominating samples whose balls cover the class (pure and proper class cover catch digraph, PCCCD) is selected greedily; the cover algorithm is the C++ implementation of the **rcccd** package, which is included here (the **imbalanceDatRel** package is not needed any more). Each synthetic sample is assigned to the dominating ball of its class with the smallest ratio of distance to radius. If it is outside that ball, it is moved onto the border of the ball along the line to the dominating sample.

Value

a list with

x_new	Original data followed by the relocated synthetic samples.
y_new	Class variable of x_new.
x_syn	Relocated synthetic samples, in the order of the input.
y_syn	Classes of the synthetic samples.
i_dominant, x_dominant, radii_dominant, cover_proportion, n_relocated	Lists with one element per class that has synthetic samples: indices and coordinates of the dominating samples, radii of their balls, proportion of the class covered, and the number of relocated samples.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Saglam, F. (2025). DatRel: a noise-tolerant data relocation approach for effective synthetic data generation in imbalanced classifiers. Machine Learning. doi:10.1007/s10994-025-06755-8

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(400, 3, 1), ncol = 2), matrix(rnorm(40, 5, 1), ncol = 2))
y <- factor(rep(c("negative", "positive"), c(200, 20)))

m0 <- SMOTE(x, y)                       # any method, or your own samples
m <- DatRel(x, y, m0$x_syn, m0$y_syn)
m$n_relocated
```

EBSMOTE *Evaluation-Based SMOTE (EBSMOTE, 2019)*

Description

Resampling with EBSMOTE.

Usage

```
EBSMOTE(
  x,
  y,
  m_alpha = 5,
  m_beta = 5,
  lambda = 3,
  v_alpha = 5,
  w_alpha = 5,
  expand_alpha = 0.3,
  ovRate = 1,
  ...
)
```

Arguments

x	feature matrix or dataframe.
y	a factor class variable. Can have more than two classes.
m_alpha, m_beta	number of nearest neighbours that decide whether a sample of the class (m_alpha) or of the other classes (m_beta) is on the border. Default is 5.
lambda	a sample is on the border if at least m / lambda but not all of its m nearest neighbours are from the other side. Default is 3.
v_alpha	number of the nearest borderline samples of the other side from which one is selected. Default is 5.
w_alpha	number of the nearest borderline samples of the class, around the selected other side sample, from which one is selected. Default is 5.
expand_alpha	weight of the other side sample in the generated sample, in $[0, 1)$. Default is 0.3.

ovRate	Oversampling rate. Class j receives round(ovRate * (n_max - n_j)) synthetic samples, where n_max is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.
...	not used.

Details

Year: 2019.

A borderline sample of the class, a nearby borderline sample of the other classes and a borderline sample of the class near that one are combined: the generated sample is $(1-\alpha)(\phi p_i + (1-\phi)p_k) + \alpha t_j$ with a uniform ϕ. If no borderline samples exist, random oversampling is used with a warning.

The method is defined for two classes. With more than two classes every class is oversampled against the union of the other classes (one-versus-rest).

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic feature data.
y_syn	Generated synthetic label data.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Hussein, A. S., Diallo, B., & Liu, J. (2019). EBSMOTE: Evaluation-based synthetic minority oversampling technique for imbalanced dataset learning. In 2019 IEEE 14th International Conference on Intelligent Systems and Knowledge Engineering (ISKE) (pp. 1226-1233). IEEE.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(600, 3, 1), ncol = 2),
           matrix(rnorm(60, 5, 1), ncol = 2))
y <- as.factor(c(rep("negative", 300), rep("positive", 30)))

m <- EBSMOTE(x = x, y = y)
table(m$y_new)
```

`evaluate` *Evaluate a Classifier with Several Metrics*

Description

Calculates the performance metrics of `metrics` in one call.

Usage

```
evaluate(
  y_true,
  y_pred = NULL,
  y_prob = NULL,
  metrics = NULL,
  average = "auto",
  positive = NULL,
  n_bins = 10
)
```

Arguments

`y_true`	factor of the true classes.
`y_pred`	predicted classes. If `NULL`, the class with the highest probability in `y_prob` is used.
`y_prob`	predicted class probabilities (matrix with one column per class, or for two classes a vector with the probability of `positive`). Needed for the probability based metrics.
`metrics`	character vector with the names of the metrics. The default `NULL` calculates all metrics that the inputs allow: without `y_prob` only the metrics based on predicted classes.
`average`, `positive`, `n_bins`	see `metrics`.

Details

The available metrics are `ACC`, `BACC`, `F1`, `GMEAN`, `MCC`, `ROCAUC`, `PRAUC`, `Brier`, `PBrier`, `BBrier`, `ECE`, `BECE`, `P4`, `IAM`, `PRECISION`, `RECALL`, `SPECIFICITY`, `KAPPA` and `LOGLOSS`. `average` is used by `F1`, `P4`, `PRECISION`, `RECALL`, `SPECIFICITY`, `ROCAUC` and `PRAUC`; the other metrics are defined for any number of classes.

Value

a named numeric vector.

Examples

```
set.seed(1)
y_true <- factor(sample(c("neg", "pos"), 200, replace = TRUE, prob = c(0.9, 0.1)))
p_pos <- ifelse(y_true == "pos", rbeta(200, 3, 2), rbeta(200, 2, 4))
evaluate(y_true, y_prob = p_pos, positive = "pos")
evaluate(y_true, y_prob = p_pos, positive = "pos", metrics = c("MCC", "IAM", "BECE"))
```

GQEO *Quadrilateral Element Oversampling (GQEO, 2025)*

Description

Resampling with GQEO.

Usage

```
GQEO(x, y, k = 5, check_planarity = TRUE, ovRate = 1, ...)
```

Arguments

x	feature matrix or dataframe.
y	a factor class variable. Can have more than two classes.
k	number of nearest neighbours from which the corners of the quadrilaterals are chosen (at least 3). Default is 5.
check_planarity	if TRUE, quadrilaterals whose edges violate the polygon inequality are removed. Default is TRUE.
ovRate	Oversampling rate. Class j receives round(ovRate * (n_max - n_j)) synthetic samples, where n_max is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.
...	not used.

Details

Year: 2025.

Synthetic samples are generated inside quadrilaterals made of a sample and three of its k nearest neighbours of the same class, by bilinear interpolation of the four corners. Classes that are too small to form a quadrilateral are oversampled randomly with a warning.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic feature data.
y_syn	Generated synthetic label data.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Dai, Q., Wang, L., Zhang, J., Ding, W., & Chen, L. (2025). GQEO: Nearest neighbor graph-based generalized quadrilateral element oversampling for class-imbalance problem. Neural Networks, 184, 107107.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(600, 3, 1), ncol = 2),
           matrix(rnorm(60, 5, 1), ncol = 2))
y <- as.factor(c(rep("negative", 300), rep("positive", 30)))

m <- GQEO(x = x, y = y)
table(m$y_new)
```

GSMOTE *Geometric SMOTE (GSMOTE, 2019)*

Description

Resampling with GSMOTE.

Usage

```
GSMOTE(
  x,
  y,
  k = 5,
  alpha_sel = c("combined", "minority", "majority"),
  alpha_trunc = 0.5,
  alpha_def = 0.5,
  ovRate = 1
)
```

Arguments

x	feature matrix.
y	a factor class variable. Can have more than two classes.
k	number of neighbors. Default is 5.
alpha_sel	selection method. Can be "minority", "majority" or "combined". Default is "combined".
alpha_trunc	truncation factor. A numeric value in $[-1, 1]$. Default is 0.5.
alpha_def	deformation factor. A numeric value in $[0, 1]$. Default is 0.5
ovRate	Oversampling rate. Class j receives `round(ovRate * (n_max - n_j))` synthetic samples, where `n_max` is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.

Details

Year: 2019.

GSMOTE (Douzas & Bacao, 2019) is an oversampling method which creates synthetic samples geometrically around selected minority samples. Details are in the paper (Douzas & Bacao, 2019).

Can work with classes more than 2.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic feature data.
y_syn	Generated synthetic label data.
C	List with one element per class: the number of synthetic samples generated from each sample of that class.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Douzas, G., & Bacao, F. (2019). Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE. Information sciences, 501, 118-135.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)

# resampling
m <- GSMOTE(x = x, y = y, k = 7)

plot(m$x_new, col = m$y_new)
```

LEAO *Local Entropy-Adversarial Oversampling (LEAO, 2025)*

Description

Resampling with LEAO.

Usage

```
LEAO(
  x,
  y,
  iters = 50,
  alpha = 1,
  prob_type = c("bagging", "logistic", "knn", "nb"),
  k = 3,
  stop_prob = 0.3,
  step_max = 0.1,
  step_min = NULL,
  eps = 1e-06,
  cov_ridge = 1e-08,
  ovRate = 1,
  ...
)
```

Arguments

x	feature matrix or dataframe.
y	a factor class variable. Can have more than two classes.
iters	maximum number of gradient steps per synthetic sample. Default is 50.
alpha	decay of the potential. Default is 1.
prob_type	how the class probabilities, which give the entropies, are estimated: "bagging" (classification trees, default), "logistic", "knn" or "nb" (Gaussian naive Bayes).

k	number of nearest samples of the other classes (Mahalanobis distance) from which one is chosen. Default is 3.
stop_prob	probability to stop at each step. Default is 0.3.
step_max, step_min	range of the random step size. step_min is step_max / 10 if NULL.
eps	numerical constant of the entropy. Default is 1e-6.
cov_ridge	value added to the diagonal of the covariance matrix. Default is 1e-8.
ovRate	Oversampling rate. Class j receives round(ovRate * (n_max - n_j)) synthetic samples, where n_max is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.
...	not used.

Details

Year: 2025.

Seeds are drawn with probabilities proportional to the entropy of their class probability. Each synthetic sample starts at its seed and moves by gradient descent on a potential that is high near the seed and low near a nearby sample of the other classes, until it is not positive, the maximum number of steps is reached or the random stop happens.

The method is defined for two classes. With more than two classes every class is oversampled against the union of the other classes (one-versus-rest).

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic feature data.
y_syn	Generated synthetic label data.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Tao, L. (2025). Local entropy-adversarial oversampling for imbalanced datasets. Neurocomputing, 131959.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(600, 3, 1), ncol = 2),
           matrix(rnorm(60, 5, 1), ncol = 2))
y <- as.factor(c(rep("negative", 300), rep("positive", 30)))

m <- LEAO(x = x, y = y)
table(m$y_new)
```

`MCSMOTE` *Markov Chain SMOTE (MCSMOTE, 2025)*

Description

Transition matrix driven oversampling with MCSMOTE.

Usage

```
MCSMOTE(x, y, thresh_qntl = 0.1, ovRate = 1, tol = 1e-04, max_iter = 1000, ...)
```

Arguments

`x`	feature matrix or dataframe.
`y`	a factor class variable. Can have more than two classes.
`thresh_qntl`	quantile, in $(0, 1]$, of the distances from a sample to the other samples of its class that determines the bandwidth of the Gaussian kernel. Smaller values favor transitions to the nearest samples (local structure), larger values give more diverse transitions. Default is 0.1.
`ovRate`	Oversampling rate. Class j receives `round(ovRate * (n_max - n_j))` synthetic samples, where `n_max` is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.
`tol`	convergence threshold of the Sinkhorn normalization. Default is 1e-4.
`max_iter`	maximum number of Sinkhorn iterations. Default is 1000.
`...`	not used.

Details

Year: 2025.

MCSMOTE (Saglam & Cengiz, 2025) models oversampling as a Markov chain over the samples of the class to be oversampled. A Gaussian kernel similarity matrix with sample specific bandwidths is calculated from the pairwise distances, where the bandwidth of sample i is the `thresh_qntl` quantile of its distances. The matrix is turned into a doubly stochastic transition matrix by Sinkhorn-Knopp normalization. A synthetic sample is generated by randomly selecting an initial sample,

drawing a target sample with the transition probabilities of the initial sample, and interpolating between the two with a uniform random coefficient.

Can work with classes more than 2. Each class smaller than the largest one is oversampled using only its own samples.

Value

a list with resampled dataset.

`x_new`	Resampled feature matrix.
`y_new`	Resampled target variable.
`x_syn`	Generated synthetic feature data.
`y_syn`	Generated synthetic label data.
`converged`	Logical vector (one element per class) telling whether the Sinkhorn normalization converged within `max_iter`.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Saglam, F., & Cengiz, M. A. (2025). MCSMOTE: A transition matrix-driven oversampling technique for class imbalance. Applied Soft Computing, 185, 113906.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)

# resampling
m <- MCSMOTE(x = x, y = y, thresh_qntl = 0.05)

plot(m$x_new, col = m$y_new)
```

---

metrics *Performance Metrics for Imbalanced Classification*

Description

Metrics that are suitable for imbalanced and multi-class classification. All of them take the true classes `y_true` (a factor) and either the predicted classes `y_pred` or the predicted class probabilities `y_prob`. Use `evaluate` to calculate several metrics at once.

Usage

```
ACC(y_true, y_pred)

BACC(y_true, y_pred)

GMEAN(y_true, y_pred)

MCC(y_true, y_pred)

KAPPA(y_true, y_pred)

IAM(y_true, y_pred)

F1(y_true, y_pred, average = "auto", positive = NULL)

PRECISION(y_true, y_pred, average = "auto", positive = NULL)

RECALL(y_true, y_pred, average = "auto", positive = NULL)

SPECIFICITY(y_true, y_pred, average = "auto", positive = NULL)

P4(y_true, y_pred, average = "auto", positive = NULL)

ROCAUC(y_true, y_prob, average = "auto", positive = NULL)

PRAUC(y_true, y_prob, average = "auto", positive = NULL)

Brier(y_true, y_prob, positive = NULL)

PBrier(y_true, y_prob, positive = NULL)

BBrier(y_true, y_prob, positive = NULL)

ECE(y_true, y_prob, n_bins = 10, positive = NULL)

BECE(y_true, y_prob, n_bins = 10, positive = NULL)

LOGLOSS(y_true, y_prob, positive = NULL)
```

Arguments

y_true	factor of the true classes.
y_pred	predicted classes. A factor or a character vector with labels from the levels of y_true.
average	how class-wise values are combined for metrics that are defined for one class: "binary" uses only the positive class, "macro" is the mean over classes, "micro" pools the counts (or scores) of all classes, "weighted" is the mean

	weighted by class sizes. The default `"auto"` is `"binary"` for two classes and `"macro"` for more.
`positive`	the positive class of the binary versions. Default is the smallest class of `y_true`.
`y_prob`	predicted class probabilities: a matrix with one column per class (named after the levels of `y_true`, or in level order), or, for two classes, a vector with the probability of `positive`.
`n_bins`	number of equal width confidence bins of `ECE` and `BECE`. Default is 10.

Details

Metrics based on predicted classes:

- `ACC`: accuracy.
- `BACC`: balanced accuracy, the mean recall over the classes.
- `GMEAN`: geometric mean of the class recalls (for two classes $\sqrt{sensitivity \cdot specificity}$).
- `MCC`: Matthews correlation coefficient, in the multi-class form of Gorodkin (2004).
- `KAPPA`: Cohen's kappa.
- `IAM`: imbalanced accuracy measure (Mortaz, 2020), in $[-1, 1]$.
- `F1`, `PRECISION`, `RECALL`, `SPECIFICITY`, `P4`: class-wise metrics combined with `average`. `P4` is the harmonic mean of precision, recall, specificity and negative predictive value (Sitarz, 2022). A class-wise value with an empty denominator is 0; classes that do not occur in `y_true` are ignored by `"macro"` and `"weighted"`. Micro averaged `F1`, `PRECISION` and `RECALL` equal `ACC`.

Metrics based on probabilities:

- `ROCAUC`: area under the ROC curve. Multi-class values are one-versus-rest and combined with `average`.
- `PRAUC`: area under the precision-recall curve, calculated as average precision. Multi-class values are one-versus-rest and combined with `average`.
- `Brier`: the Brier score, the mean over observations of the sum over classes of the squared differences between the probabilities and the one-hot true class. Lies in $[0, 2]$; for two classes it is twice the usual $(p - y)^2$ form.
- `PBrier`: penalized Brier score (Ahmadian et al., 2024): the Brier score plus $(K - 1)/K$ for every misclassified observation, which keeps a wrongly classified observation from scoring better than a correctly classified one.
- `BBrier`: balanced Brier score, the mean of the class-wise Brier scores, so that every class counts equally.
- `ECE`: expected calibration error of the predicted class (confidence), $\sum_b (n_b/n)|acc_b - conf_b|$ over equal width confidence bins.
- `BECE`: balanced expected calibration error, `ECE` where every observation is weighted by the inverse size of its true class.
- `LOGLOSS`: logarithmic loss.

The probabilities of `ECE`, `BECE`, `PBrier` use the class with the highest probability as the predicted class.

Value

a number.

References

Gorodkin, J. (2004). Comparing two K-category assignments by a K-category correlation coefficient. Computational Biology and Chemistry, 28(5-6), 367-374.

Mortaz, E. (2020). Imbalance accuracy metric for model selection in multi-class imbalance classification problems. Knowledge-Based Systems, 210, 106490.

Sitarz, M. (2022). Extending F1 metric, probabilistic approach. arXiv:2210.11997.

Ahmadian, S. et al. (2024). Superior scoring rules for probabilistic evaluation of single-label multiclass classification tasks. arXiv:2407.17697.

Examples

```
set.seed(1)
y_true <- factor(sample(c("a", "b", "c"), 100, replace = TRUE, prob = c(0.7, 0.2, 0.1)))
prob <- matrix(runif(300), 100, 3, dimnames = list(NULL, levels(y_true)))
prob <- prob / rowSums(prob)
y_pred <- factor(levels(y_true)[max.col(prob)], levels = levels(y_true))
BACC(y_true, y_pred)
F1(y_true, y_pred, average = "micro")
MCC(y_true, y_pred)
ROCAUC(y_true, prob, average = "macro")
Brier(y_true, prob)
```

---

oversampleDatRel *Oversampling followed by DatRel (2025)*

Description

Generates synthetic samples with a base method and post-processes them with DatRel. DatRel (Saglam, 2025) makes the synthetic samples of any oversampling method safer: synthetic samples that fall outside the regions that the class covers are moved back into them. DatRel is not an oversampling method itself; it relocates the samples generated by a base method.

Usage

```
oversampleDatRel(
  x,
  y,
  method = "SMOTE",
  ovRate = 1,
  proportion = 1,
  p_of = 0,
  ...
)
```

Arguments

x	feature matrix.
y	a factor class variable. Can have more than two classes.
method	base oversampling method that generates the synthetic samples. One of "ROS", "RWO", "ROSE", "SMOTE", "ADASYN", "BLSMOTE", "SLSMOTE", "RSLSMOTE", "GSMOTE", "SMOTEWB", "BiSMOTE", "MCSMOTE", "EBSMOTE", "GQEO", "LEAO", "SyMProD", "RadiusSMOTE" or "VSSMOTE". Default is "SMOTE".
ovRate	Oversampling rate, as in the base method: class j receives round(ovRate * (n_max - n_j)) synthetic samples. Default is 1.
proportion	proportion of the samples of a class that has to be covered by the dominating balls. Default is 1 (all samples).
p_of	overflow proportion: the radii of the balls are multiplied by 1 + p_of, so the synthetic samples are allowed to stay slightly outside of the class region. Default is 0.
...	arguments of the base method, for example k = 7 for method = "SMOTE".

Details

Year: 2025. For every class (one-vs-rest for more than two classes) a ball is placed around each sample whose radius is the distance to the nearest sample of the other classes (times 1 + p_of), so that the ball contains no sample of another class. A small set of dominating samples whose balls cover the class (pure and proper class cover catch digraph, PCCCD) is selected greedily. Each synthetic sample is assigned to the dominating ball with the smallest ratio of distance to radius. If the sample is outside of that ball, it is moved onto the border of the ball along the line to the dominating sample. The cover algorithm is the C++ implementation of the **rcccd** package, which is included here.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated (relocated) synthetic feature data.
y_syn	Generated synthetic label data.
i_dominant	List (one element per class) of the indices of the dominating samples within the class.
x_dominant	List of the dominating samples.
radii_dominant	List of the radii of the dominating balls.
cover_proportion	List of the proportions of the class covered.
n_relocated	List of the numbers of relocated synthetic samples.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Saglam, F. (2025). DatRel: a noise-tolerant data relocation approach for effective synthetic data generation in imbalanced classifiers. Machine Learning. doi:10.1007/s10994-025-06755-8

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

m <- oversampleDatRel(x = x, y = y, method = "SMOTE", k = 7)
m$n_relocated
plot(m$x_new, col = m$y_new)
```

RadiusSMOTE *Radius-SMOTE (RadiusSMOTE, 2021)*

Description

Resampling with RadiusSMOTE.

Usage

```
RadiusSMOTE(x, y, k = 5, ovRate = 1, ...)
```

Arguments

x	feature matrix or data.frame.
y	a factor class variable. Can have more than two classes.
k	number of neighbors used to decide whether a sample is safe (a sample is safe when more than k / 2 of its k nearest neighbors in the whole data are from its own class). Default is 5.
ovRate	Oversampling rate. Class j receives round(ovRate * (n_max - n_j)) synthetic samples, where n_max is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.
...	not used.

Details

Year: 2021. Samples of the class that are safe are used as centers. Synthetic samples are generated on the line between a center and its nearest sample of the other classes, either towards it or in the opposite direction, at a random distance of at most the distance to that sample (the safe radius), so they do not reach the other classes. If no sample is safe, all samples are used with a warning.

Can work with classes more than 2: every class is oversampled against the union of the other classes (one-versus-rest).

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic feature data.
y_syn	Generated synthetic label data.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Pradipta, G. A., Wardoyo, R., Musdholifah, A., & Sanjaya, I. N. H. (2021). Radius-SMOTE: a new oversampling technique of minority samples based on radius distance for learning from imbalanced data. IEEE Access, 9, 74763-74777.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

m <- RadiusSMOTE(x = x, y = y, k = 5)
table(m$y_new)
plot(m$x_new, col = m$y_new)
```

ROS *Random Oversampling (ROS)*

Description

Resampling with ROS.

Usage

```
ROS(x, y, ovRate = 1, ...)
```

Arguments

x	feature matrix.
y	a factor class variable. Can have more than two classes.
ovRate	Oversampling rate. Class j receives `round(ovRate * (n_max - n_j))` synthetic samples, where `n_max` is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.
...	not used.

Details

Random Oversampling (ROS) is a method of copying and pasting of positive samples until balance is achieved.

Can work with classes more than 2.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.

Author(s)

Fatih Saglam, saglamf89@gmail.com

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)

# resampling
m <- ROS(x = x, y = y)

plot(m$x_new, col = m$y_new)
```

ROSE *Random Over-Sampling Examples (ROSE, 2014)*

Description

Generates synthetic data for each class to balance imbalanced datasets using kernel density estimations. Can be used for multiclass datasets.

Usage

```
ROSE(x, y, h = 1, ovRate = 1)
```

Arguments

x	feature matrix or data.frame.
y	a factor class variable. Can be multiclass.
h	A numeric vector of length one or number of classes in y. If one is given, all classes will have same shrink factor. If a value is given for each classes, it will match respectively to `levels(y)`. Default is 1.
ovRate	Oversampling rate. Class j receives `round(ovRate * (n_max - n_j))` synthetic samples, where `n_max` is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.

Details

Year: 2014.

Randomly Over Sampling Examples (ROSE) (Menardi and Torelli, 2014) is an oversampling method which uses conditional kernel densities to balance dataset. There is already an R package called 'ROSE' (Lunardon et al., 2014). Difference is that this one is much faster and can be applied for more than two classes.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.

Author(s)

Fatih Saglam, saglamf89@gmail.com

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

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)

# resampling
m <- ROSE(x = x, y = y, h = c(0.12, 1))

plot(m$x_new, col = m$y_new)
```

RSLSMOTE *Relocating Safe-Level SMOTE (RSLSMOTE, 2016)*

Description

The Relocating Safe-Level SMOTE (RSLS) algorithm improves the quality of synthetic samples generated by Safe-Level SMOTE (SLS) by relocating specific synthetic data points that are too close to the majority class distribution towards the original minority class distribution in the feature space.

Usage

```
RSLSMOTE(x, y, k1 = 5, k2 = 5, ovRate = 1)
```

Arguments

x	feature matrix or data.frame.
y	a factor class variable. Can have more than two classes.
k1	number of neighbors to link. Default is 5.
k2	number of neighbors to determine safe levels. Default is 5.
ovRate	Oversampling rate. Class *j* receives `round(ovRate * (n_max - n_j))` synthetic samples, where `n_max` is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.

Details

Year: 2016.

In Safe-level SMOTE (SLS), a safe-level threshold is used to control the number of synthetic samples generated from each minority instance. This threshold is calculated based on the number of minority and majority instances in the local neighborhood of each minority instance. SLS generates synthetic samples that are located closer to the original minority class distribution in the feature space.

In Relocating safe-level SMOTE (RSLS), after generating synthetic samples using the SLS algorithm, the algorithm relocates specific synthetic data points that are deemed to be too close to the majority class distribution in the feature space. The relocation process moves these synthetic data points towards the original minority class distribution in the feature space.

This relocation process is performed by first identifying the synthetic data points that are too close to the majority class distribution. Then, for each identified synthetic data point, the algorithm calculates a relocation vector based on the distance between the synthetic data point and its k nearest minority class instances. This relocation vector is used to move the synthetic data point towards the minority class distribution in the feature space.

Can work with classes more than 2.

Value

a list with resampled dataset.

`x_new`	Resampled feature matrix.
`y_new`	Resampled target variable.
`x_syn`	Generated synthetic data.
`C`	Number of synthetic samples for each positive class samples.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Siriseriwan, W., & Sinapiromsaran, K. (2016). The effective redistribution for imbalance dataset: Relocating safe-level SMOTE with minority outcast handling. Chiang Mai J. Sci, 43(1), 234-246.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)

# resampling
m <- RSLSMOTE(x = x, y = y, k1 = 5, k2 = 5)
```

```
plot(m$x_new, col = m$y_new)
```

RUS *Random Undersampling (RUS)*

Description

Resampling with RUS.

Usage

```
RUS(x, y, n_neededToRemove = NULL, ...)
```

Arguments

x	feature matrix.
y	a factor class variable. Can have more than two classes.
n_neededToRemove	vector of desired number removal for each class. A vector of integers for each class. Default is NULL meaning full balance. Must be equal or lower than the number of samples in each class.
...	not used.

Details

Random Undersampling (RUS) is a method of removing negative samples until balance is achieved.

Can work with classes more than 2.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.

Author(s)

Fatih Saglam, saglamf89@gmail.com

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)

# resampling
m <- RUS(x = x, y = y)

plot(m$x_new, col = m$y_new)
```

RWO *Random Walk Oversampling (RWO, 2014)*

Description

Resampling with RWO

Usage

```
RWO(x, y, ovRate = 1)
```

Arguments

x	feature matrix.
y	a factor class variable. Can have more than two classes.
ovRate	Oversampling rate. Class j receives `round(ovRate * (n_max - n_j))` synthetic samples, where `n_max` is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.

Details

Year: 2014.

RWO (Zhang and Li, 2014) is an oversampling method which generates data using variable standard error in a way that it preserves the variances of all variables.

Can work with classes more than 2.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic feature data.
y_syn	Generated synthetic label data.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Zhang, H., & Li, M. (2014). RWO-Sampling: A random walk over-sampling approach to imbalanced data classification. Information Fusion, 20, 99-116.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)

# resampling
m <- RWO(x = x, y = y)

plot(m$x_new, col = m$y_new)
```

SLSMOTE *Safe-Level SMOTE (SLSMOTE, 2009)*

Description

`SLSMOTE()` generates synthetic samples by considering a safe level of the nearest minority class examples.

Usage

```
SLSMOTE(x, y, k1 = 5, k2 = 5, ovRate = 1)
```

Arguments

`x`	feature matrix or data.frame.
`y`	a factor class variable. Can have more than two classes.
`k1`	number of neighbors to link. Default is 5.
`k2`	number of neighbors to determine safe levels. Default is 5.
`ovRate`	Oversampling rate. Class *j* receives `round(ovRate * (n_max - n_j))` synthetic samples, where `n_max` is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.

Details

Year: 2009.

SLSMOTE uses the safe-level distance metric to identify the minority class samples that are safe to oversample. Safe-level distance measures the distance between a minority class sample and its k-nearest minority class neighbors. A sample is considered safe to oversample if its safe-level is greater than a threshold. The safe-level of a sample is the ratio of minority class samples among its k-nearest neighbors.

In SLSMOTE, the oversampling process only applies to the safe minority class samples, which avoids the generation of noisy samples that can lead to overfitting. To generate synthetic samples, SLSMOTE randomly selects a minority class sample and finds its k-nearest minority class neighbors. Then, a random minority class neighbor is selected, and a synthetic sample is generated by adding a random proportion of the difference between the selected sample and its neighbor to the selected sample.

Can work with classes more than 2.

Value

a list with resampled dataset.

`x_new`	Resampled feature matrix.
`y_new`	Resampled target variable.
`x_syn`	Generated synthetic data.
`C`	Number of synthetic samples for each positive class samples.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2009). Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13 (pp. 475-482). Springer Berlin Heidelberg.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)

# resampling
m <- SLSMOTE(x = x, y = y, k1 = 5, k2 = 5)

plot(m$x_new, col = m$y_new)
```

SMOTE *Synthetic Minority Oversampling Technique (SMOTE, 2002)*

Description

Resampling with SMOTE.

Usage

```
SMOTE(x, y, k = 5, ovRate = 1, ...)
```

Arguments

x	feature matrix.
y	a factor class variable. Can have more than two classes.
k	number of neighbors. Default is 5.
ovRate	Oversampling rate. Class j receives `round(ovRate * (n_max - n_j))` synthetic samples, where `n_max` is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.
...	not used.

Details

Year: 2002.

SMOTE (Chawla et al., 2002) is an oversampling method which creates links between positive samples and nearest neighbors and generates synthetic samples along that link.

It is well known that SMOTE is sensitive to noisy data. It may create more noise.

Can work with classes more than 2.

Note: Much faster than `smotefamily::SMOTE()`.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic feature data.
y_syn	Generated synthetic label data.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)

# resampling
m <- SMOTE(x = x, y = y, k = 7)

plot(m$x_new, col = m$y_new)
```

SMOTEWB *SMOTE with Boosting (SMOTEWB, 2022)*

Description

Resampling with SMOTE with boosting.

Usage

```
SMOTEWB(
  x,
  y,
  n_weak_classifier = 100,
  class_weights = NULL,
  k_max = NULL,
  ovRate = 1,
  ...
)
```

Arguments

x	feature matrix.
y	a factor class variable. Can have more than two classes.
n_weak_classifier	number of weak classifiers for boosting.
class_weights	numeric vector of length two. First number is for positive class, and second is for negative. Higher the relative weight, lesser noises for that class. By default, $2 \times n_{neg}/n$ for positive and $2 \times n_{pos}/n$ for negative class.

k_max	to increase maximum number of neighbors. Default is ceiling(n_neg/n_pos).
ovRate	Oversampling rate. Class j receives round(ovRate * (n_max - n_j)) synthetic samples, where n_max is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.
...	not used.

Details

Year: 2022.

SMOTEWB (Saglam & Cengiz, 2022) is a SMOTE-based oversampling method which can handle noisy data and adaptively decides the appropriate number of neighbors to link during resampling with SMOTE.

Trained model based on this method gives significantly better Matthew Correlation Coefficient scores compared to others.

Can work with classes more than 2.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic data.
y_syn	Generated synthetic data labels.
w	Boosting weights for original dataset.
k	Number of nearest neighbors for positive class samples.
C	Number of synthetic samples for each positive class samples.
fl	"good", "bad" and "lonely" sample labels

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Sağlam, F., & Cengiz, M. A. (2022). A novel SMOTE-based resampling technique trough noise detection and the boosting procedure. Expert Systems with Applications, 200, 117023.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

plot(x, col = y)
```

```
# resampling
m <- SMOTEWB(x = x, y = y, n_weak_classifier = 150)

plot(m$x_new, col = m$y_new)
```

SyMProD *Synthetic Minority Based on Probabilistic Distribution (SyMProD, 2020)*

Description

Resampling with SyMProD.

Usage

```
SyMProD(x, y, NT = 3, CT = 0, K = 5, M = 5, ovRate = 1, ...)
```

Arguments

x	feature matrix or dataframe.
y	a factor class variable. Can have more than two classes.
NT	z-score threshold of the noise removal. Samples with an absolute z-score above NT in any feature are removed from the calculations. Default is 3.
CT	threshold of the selection: a sample is used if its density in the class is larger than CT times its density of the other classes. Default is 0.
K	number of neighbours of the density calculation. Default is 5.
M	number of neighbours that are combined with the selected base sample. Default is 5.
ovRate	Oversampling rate. Class j receives round(ovRate * (n_max - n_j)) synthetic samples, where n_max is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.
...	not used.

Details

Year: 2020.

After noise removal, each sample of the class gets a closeness factor and a density from its neighbours in the class and in the other classes. Base samples are selected with probabilities proportional to their density ratio and replaced by a random convex combination of the base sample and its M nearest neighbours. If too few samples remain, random oversampling is used with a warning.

The method is defined for two classes. With more than two classes every class is oversampled against the union of the other classes (one-versus-rest).

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic feature data.
y_syn	Generated synthetic label data.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Kunakorntum, I., Hinthong, W., & Phunchongharn, P. (2020). A synthetic minority based on probabilistic distribution (SyMProD) oversampling for imbalanced datasets. IEEE Access, 8, 114692-114704.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(600, 3, 1), ncol = 2),
           matrix(rnorm(60, 5, 1), ncol = 2))
y <- as.factor(c(rep("negative", 300), rep("positive", 30)))

m <- SyMProD(x = x, y = y)
table(m$y_new)
```

UGO *Uncertainty Guided Oversampling (UGO, 2026)*

Description

UGO (Saglam & Cengiz, 2026) is not a resampling method on its own. It is a classifier-guided procedure that *uses* a base oversampling method (type) and a classifier: the classifier decides which samples are informative, which class needs more samples and when to stop.

Usage

```
UGO(
  x,
  y,
  type = c("RWO", "SMOTE", "ROS", "MCSMOTE"),
  classifier = "RF",
  comparison = c("anova", "kruskal"),
  prop_generate = 0.1,
```

```
  prop_sampling = 0.5,
  k_max = 5,
  k_noise = 5,
  pen_noise = 1,
  k_after = 1,
  lambda_max = 2.5,
  alpha = 0.05,
  classifier_args = list(),
  generator_args = list(),
  verbose = FALSE,
  ...
)
```

Arguments

x	feature matrix or dataframe.
y	a factor class variable. Can have more than two classes.
type	base oversampling method used to generate synthetic samples around the selected informative samples. One of "RWO", "SMOTE", "ROS" or "MCSMOTE". Default is "RWO".
classifier	classifier used to calculate the uncertainty. One of the built-in classifiers "CART" (**rpart**), "RF" (**randomForest**), "LogReg" (logistic regression, one-vs-rest for more than two classes), "KNN" (**FNN**), "NaiveBayes" (Gaussian), "LDA" or "NNET" (**nnet**), or a function function(x_train, y_train, x_test, ...) that returns a matrix of class probabilities for the rows of x_test, with one column per level of y_train. The old names "rpart" and "randomForest" are still accepted. UGO is classifier specific, so the classifier that will be used for the final model should be used here. Default is "RF".
comparison	test used to compare the average uncertainty of the classes: "anova" (Welch's one-way test) or "kruskal" (Kruskal-Wallis). Generation stops when classes are not significantly different.
prop_generate	proportion of the sample size generated at each iteration (λ_{step} in the paper). Default is 0.10.
prop_sampling	proportion of the samples of the selected class that are chosen as the most uncertain samples to generate data from (λ_{select} in the paper). Default is 0.5.
k_max	number of nearest neighbors for type = "SMOTE". Default is 5. If NULL, ceiling(n / number of selected samples).
k_noise	number of neighbors for the initial noise detection (k_{g-Pre} in the paper). Default is 5. If 0, no sample is treated as noise.
pen_noise	multiplier of the threshold of the noise detection. A sample is noise when fewer than k_noise / K * pen_noise of its neighbors share its class, where K is the number of classes. Default is 1.
k_after	number of neighbors for the detection of noise among the generated samples ($k_{g-After}$ in the paper). Default is 1.
lambda_max	the algorithm stops when the data size exceeds lambda_max * K * n (ϕ in the paper). Default is 2.5.

alpha	significance level of the comparison test used as the stopping criterion. Default is 0.05.
classifier_args	list of additional arguments for the classifier (e.g. list(k = 7) for "KNN").
generator_args	list of additional arguments for the base method type. Only thresh_qntl, tol and max_iter of MCSMOTE are used.
verbose	print the progress. Default is FALSE.
...	not used.

Details

Year: 2026. UGO (Saglam & Cengiz, 2026) is an iterative oversampling method which uses the uncertainty of a classifier as in active learning. At each iteration, a classifier is trained on the current data and the modified margin uncertainty (the highest probability of the other classes minus the probability of the true class) is calculated for the original samples that are not noise. The class with the highest average uncertainty is selected and prop_generate * n synthetic samples are generated around its prop_sampling most uncertain samples with the base method type. Generated samples that fall in the region of another class are relocated to the nearest selected sample. The algorithm stops when the average uncertainties of the classes are not significantly different, so the number of synthetic samples does not need to be specified. Samples in class overlap regions (noise) are excluded from generation.

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic feature data.
y_syn	Generated synthetic label data.
i_noise	Logical vector of samples detected as noise.
stop_reason	Why the algorithm stopped: "no_difference" (the uncertainties of the classes are not significantly different), "max_size", "too_much_noise" or "no_uncertainty".
n_iter	Number of completed generation iterations.
unc_mean_initial, unc_mean_final	Average uncertainty of each class (noise excluded) before the first and at the last iteration.
unc_rank_mean_initial, unc_rank_mean_final	Average rank of the uncertainty of each class before the first and at the last iteration.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Saglam, F., & Cengiz, M. A. (2026). Uncertainty guided oversampling: A classifier-specific approach for effective handling of class imbalance in machine learning. Knowledge-Based Systems, 342, 115886.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(600, 3, 1), ncol = 2, nrow = 300),
           matrix(rnorm(60, 5, 1), ncol = 2, nrow = 30))
y <- as.factor(c(rep("negative", 300), rep("positive", 30)))

plot(x, col = y)

# resampling
if (requireNamespace("randomForest", quietly = TRUE)) {
  m <- UGO(x = x, y = y, type = "SMOTE", classifier = "RF")
  m$stop_reason

  plot(m$x_new, col = m$y_new)
}

# built-in classifiers that need no extra package
m <- UGO(x = x, y = y, type = "SMOTE", classifier = "LDA")
m$stop_reason

# any classifier that returns class probabilities can be used
f_logistic <- function(x_train, y_train, x_test, ...) {
  dat <- data.frame(x_train, y = y_train)
  model <- glm(y ~ ., data = dat, family = binomial())
  p1 <- predict(model, newdata = data.frame(x_test), type = "response")
  cbind(1 - p1, p1)
}
m <- UGO(x = x, y = y, type = "ROS", classifier = f_logistic)
table(m$y_new)
```

VSSMOTE *VS-SMOTE (VSSMOTE, 2025)*

Description

Resampling with VSSMOTE.

Usage

```
VSSMOTE(x, y, k = 5, k1 = 0, k2 = k, ovRate = 1, ...)
```

Arguments

x	feature matrix or data.frame.
y	a factor class variable. Can have more than two classes.
k	number of nearest neighbors of the class that each sample is linked to (connecting lines) and size of the neighborhood of the middle points. Default is 5.
k1, k2	a connecting line is used if the number of samples of the class among the k nearest neighbors of its middle point is between k1 and k2 (high-value space). k1 = 0 and k2 = k (default) give the ordinary SMOTE. Raising k1 removes lines in majority dominated (noisy) spaces, lowering k2 removes lines in the safest spaces. If NULL, k2 is k.
ovRate	Oversampling rate. Class j receives round(ovRate * (n_max - n_j)) synthetic samples, where n_max is the size of the largest class (halves are rounded up). Default is 1 meaning full balance. A vector with one value per class is also accepted.
...	not used.

Details

Year: 2025. Each sample is linked to its k nearest neighbors of the class. The middle point of every connecting line is used to judge the space of the line, and synthetic samples are generated uniformly along randomly chosen lines of the high-value spaces only. If no line qualifies, all lines are used with a warning.

Can work with classes more than 2: every class is oversampled against the union of the other classes (one-versus-rest).

Value

a list with resampled dataset.

x_new	Resampled feature matrix.
y_new	Resampled target variable.
x_syn	Generated synthetic feature data.
y_syn	Generated synthetic label data.

Author(s)

Fatih Saglam, saglamf89@gmail.com

References

Qiu, F., Yang, F., Xiong, Y., Chen, T., Xiao, P., Zheng, W., ... & Zhou, K. (2025). VS-SMOTE: Leveraging high-value spaces to balance noise control and data diversity for class imbalanced learning. Neurocomputing, 131321.

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(2000, 3, 1), ncol = 2, nrow = 1000),
           matrix(rnorm(100, 5, 1), ncol = 2, nrow = 50))
y <- as.factor(c(rep("negative", 1000), rep("positive", 50)))

m <- VSSMOTE(x = x, y = y, k = 5, k1 = 2, k2 = 5)
table(m$y_new)
plot(m$x_new, col = m$y_new)
```

Index

∗ **datasets**
banana, 4

ACC *(*metrics*)*, 23
ADASYN, 3

BACC *(*metrics*)*, 23
banana, 4
BBrier *(*metrics*)*, 23
BECE *(*metrics*)*, 23
BiSMOTE, 5
BLSMOTE, 6
Brier *(*metrics*)*, 23

complexity, 8, *9*
complexity_BORDERLINE *(*complexity_measures*)*, 9
complexity_C1 *(*complexity_measures*)*, 9
complexity_C2 *(*complexity_measures*)*, 9
complexity_F1 *(*complexity_measures*)*, 9
complexity_F1v *(*complexity_measures*)*, 9
complexity_F2 *(*complexity_measures*)*, 9
complexity_F3 *(*complexity_measures*)*, 9
complexity_F4 *(*complexity_measures*)*, 9
complexity_ID *(*complexity_measures*)*, 9
complexity_IR *(*complexity_measures*)*, 9
complexity_LSC *(*complexity_measures*)*, 9
complexity_measures, *8*, 9, *9*
complexity_N1 *(*complexity_measures*)*, 9
complexity_N2 *(*complexity_measures*)*, 9
complexity_N3 *(*complexity_measures*)*, 9
complexity_N3_bal *(*complexity_measures*)*, 9
complexity_N4 *(*complexity_measures*)*, 9
complexity_OUTLIER *(*complexity_measures*)*, 9
complexity_R *(*complexity_measures*)*, 9
complexity_R_aug *(*complexity_measures*)*, 9
complexity_RARE *(*complexity_measures*)*, 9
complexity_SAFE *(*complexity_measures*)*, 9

DatRel, 12, *26*

EBSMOTE, 14
ECE *(*metrics*)*, 23
evaluate, 16, *23*

F1 *(*metrics*)*, 23

GMEAN *(*metrics*)*, 23
GQEO, 17
GSMOTE, 18

IAM *(*metrics*)*, 23

KAPPA *(*metrics*)*, 23

LEAO, 20
LOGLOSS *(*metrics*)*, 23

MCC *(*metrics*)*, 23
MCSMOTE, 22, *44*
metrics, *16*, 23

oversampleDatRel, 26

P4 *(*metrics*)*, 23
PBrier *(*metrics*)*, 23
PRAUC *(*metrics*)*, 23
PRECISION *(*metrics*)*, 23

RadiusSMOTE, 28
RECALL *(*metrics*)*, 23
ROCAUC *(*metrics*)*, 23
ROS, 29
ROSE, 31
RSLSMOTE, 32
RUS, 34
RWO, 35

SLSMOTE, 36

SMOTE, 38
SMOTEWB, 39
SPECIFICITY (metrics), 23
SyMProD, 41

UGO, 42

VSSMOTE, 45