Package ‘smotefamily’

October 14, 2022

Title A Collection of Oversampling Techniques for Class Imbalance Problem Based on SMOTE

Version 1.3.1

Date 2019-05-30

Maintainer Wacharasak Siriseriwan <wacharasak.s@gmail.com>

Description A collection of various oversampling techniques developed from SMOTE is provided. SMOTE is an oversampling technique which synthesizes a new minority instance between a pair of one minority instance and one of its K nearest neighbors. (see <https://www.jair.org/media/953/live-953-2037-jair.pdf> for more information) Other techniques adopt this concept with other criteria in order to generate balanced dataset for class imbalance problem.

License GPL (>= 3)

Depends R(>= 3.0.0)

Imports FNN, dbscan, igraph,

NeedsCompilation no

Author Wacharasak Siriseriwan [aut, cre]

Repository CRAN

Date/Publication 2019-05-30 07:30:02 UTC

R topics documented:

smotefamily-package ........................................ 2
ADASYN .................................................. 3
ANS ....................................................... 4
Borderline-SMOTE ........................................ 5
DBSMOTE ................................................ 6
gap .......................................................... 8
kncount ................................................... 8
knearest .................................................. 9
n_dup_max ............................................... 10
RSL ....................................................... 11
sample_generator ....................................... 12
Description

A more detailed description of what the package does. A length of about one to five lines is recommended.

Details

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Your Name, email optional.

Maintainer: Your Name <your@email.com>

References

This optional section can contain literature or other references for background information.

See Also

Optional links to other man pages

Examples

## Not run:
## Optional simple examples of the most important functions
## These can be in \dontrun{} and \donttest{} blocks.

## End(Not run)
ADASYN

Adaptive Synthetic Sampling Approach for Imbalanced Learning

Description

Generate synthetic positive instances using ADASYN algorithm. The number of majority neighbors of each minority instance determines the number of synthetic instances generated from the minority instance.

Usage

ADAS(X, target, \( K = 5 \))

Arguments

- **X**: A data frame or matrix of numeric-attributed dataset
- **target**: A vector of a target class attribute corresponding to a dataset X.
- **K**: The number of nearest neighbors during sampling process

Value

- **data**: A resulting dataset consists of original minority instances, synthetic minority instances and original majority instances with a vector of their respective target class appended at the last column
- **syn_data**: A set of synthetic minority instances with a vector of minority target class appended at the last column
- **orig_N**: A set of original instances whose class is not oversampled with a vector of their target class appended at the last column
- **orig_P**: A set of original instances whose class is oversampled with a vector of their target class appended at the last column
- **K**: The value of parameter K for nearest neighbor process used for generating data
- **K_all**: Unavailable for this method
- **dup_size**: A vector of times of synthetic minority instances over original majority instances in the oversampling in each instances
- **outcast**: Unavailable for this method
- **eps**: Unavailable for this method
- **method**: The name of oversampling method used for this generated dataset (ADASYN)

Author(s)

Wacharasak Siriseriwan <wacharasak.s@gmail.com>
References


Examples

data_example = sample_generator(10000, ratio = 0.80)
genData = ADAS(data_example[, -3], data_example[, 3])
genData_2 = ADAS(data_example[, -3], data_example[, 3], K=7)

ANS

Adaptive Neighbor Synthetic Majority Oversampling Technique

Description

Generate a oversampling dataset from imbalanced dataset using Adaptive Neighbor SMOTE which provides the parameter K to each minority instance automatically

Usage

ANS(X, target, dupSize = 0)

Arguments

X A data frame or matrix of numeric-attributed dataset
target A vector of a target class attribute corresponding to a dataset X.
dupSize A number of vector representing the desired times of synthetic minority instances over the original number of majority instances, 0 for balanced dataset.

Value

data A resulting dataset consists of original minority instances, synthetic minority instances and original majority instances with a vector of their respective target class appended at the last column
syn_data A set of synthetic minority instances with a vector of minority target class appended at the last column
orig_N A set of original instances whose class is not oversampled with a vector of their target class appended at the last column
orig_P A set of original instances whose class is oversampled with a vector of their target class appended at the last column
K A vector of parameter K for each minority instance
K_all The value of parameter C for nearest neighbor process used for identifying outcasts
Borderline-SMOTE

dup_size The maximum times of synthetic minority instances over original majority instances in the oversampling
outcast A set of original minority instances which is defined as minority outcast
eps The value of eps which determines automatic K
method The name of oversampling method used for this generated dataset (ANS)

Author(s)
Wacharasak Siriseriwan <wacharasak.s@gmail.com>

References

Examples
data_example = sample_generator(5000, ratio = 0.80)
genData = ANS(data_example[,-3],data_example[,3])

---

Description
Generate synthetic positive instances using Borderline-SMOTE algorithm. The number of majority neighbor of each minority instance is used to divide minority instances into 3 groups; SAFE/DANGER/NOISE, only the DANGER are used to generate synthetic instances.

Usage
BLSMOTE(X,target,K=5,C=5,dupSize=0,method =c("type1","type2"))

Arguments
X A data frame or matrix of numeric-attributed dataset
target A vector of a target class attribute corresponding to a dataset X.
K The number of nearest neighbors during sampling process
C The number of nearest neighbors during calculating safe-level process
dupSize The number or vector representing the desired times of synthetic minority instances over the original number of majority instances, 0 for duplicating until balanced
method A parameter to indicate which type of Borderline-SMOTE presented in the paper is used
Value

- **data**: A resulting dataset consists of original minority instances, synthetic minority instances and original majority instances with a vector of their respective target class appended at the last column.
- **syn_data**: A set of synthetic minority instances with a vector of minority target class appended at the last column.
- **orig_N**: A set of original instances whose class is not oversampled with a vector of their target class appended at the last column.
- **orig_P**: A set of original instances whose class is oversampled with a vector of their target class appended at the last column.
- **K**: The value of parameter K for nearest neighbor process used for generating data.
- **K_all**: The value of parameter C for nearest neighbor process used for determining SAFE/DANGER/NOISE.
- **dup_size**: The maximum times of synthetic minority instances over original majority instances in the oversampling.
- **outcast**: Unavailable for this method.
- **eps**: Unavailable for this method.
- **method**: The name of oversampling method and type used for this generated dataset (BLSMOTE type 1/2).

Author(s)

Wacharasak Siriseriwan <wacharasak.s@gmail.com>

References


Examples

```r
data_example = sample_generator(5000, ratio = 0.80)
genData = BLSMOTE(data_example[,-3],data_example[,3])
genData_2 = BLSMOTE(data_example[,-3],data_example[,3],K=7, C=5, method = "type2")
```

Description

Generate a oversampling dataset from imbalance dataset using Density-based SMOTE. Using density reachability concept to cluster minority instances and generate synthetic instances.
Usage

DBSMOTE(X, target, dupSize = 0, MinPts = NULL, eps = NULL)

Arguments

X A data frame or matrix of numeric-attributed dataset
target A vector of a target class attribute
dupSize A number of vector representing the desired times of synthetic minority instances over the original number of majority instances
MinPts The minimum instance parameter to decide whether each instance inside eps is reachable, the automatic algorithm is used to find the value instead if there is no positive integer value given for it.
eps The radius to consider neighbor.

Value

data A resulting dataset consists of original minority instances, synthetic minority instances and original majority instances with a vector of their respective target class appended at the last column
syn_data A set of synthetic minority instances with a vector of minority target class appended at the last column
orig_N A set of original instances whose class is not oversampled with a vector of their target class appended at the last column
orig_P A set of original instances whose class is oversampled with a vector of their target class appended at the last column
K Unavailable for this method
K_all Unavailable for this method
dup_size The maximum times of synthetic minority instances over original majority instances in the oversampling
outcast A set of original minority instances which is defined as NOISE/minority outcast
eps The value of parameter eps
method The name of oversampling method used for this generated dataset

Author(s)

Wacharasak Siriseriwan <wacharasak.s@gmail.com>

References


Examples

data_example = sample_generator(5000,ratio = 0.90)
genData = DBSMOTE(data_example[,,-3],data_example[,3])
**gaps**  
*The function to provide a random number which is used as a location of synthetic instance*

**Description**  
The function to provide a random number which uses to identify the location of each synthetic instance. The interval of possible values depends from safe-level values of instances in a pair.

**Usage**  
gap(sl_p = 1, sl_n = 1)

**Arguments**  
- **sl_p**  
The safe-level value of the first instance  
- **sl_n**  
The safe-level value of the second instance

**Value**  
A value between 0 to 1 which is used to identify the location of synthetic instance. If sl_p >= sl_n, it gives the random number between 0 to sl_n/sl_p. If sl_p < sl_n, it gives the random number between 1-sl_p/sl_n to 1.

**Author(s)**  
Wacharasak Siriseriwan <wacharasak.s@gmail.com>

**Examples**  

```r  
r_num = gap()  
r_num_2 = gap(sl_p = 4, sl_n = 2)  
```

**knecount**  
*Counting the number of each class in K nearest neighbor*

**Description**  
The function to count how many neighbor of each instance belong to each class.

**Usage**  
knecount(knindex, classArray)
**knearest**

**Arguments**

- **knidex**
  - The matrix of K nearest neighbor of dataset

- **classArray**
  - The index of last instance of the first class in the dataset or the vector containing indices of last instances of each class.

**Details**

The dataset is expected to be sorted as all m1 instances in the first class are in the first m1 instances of the dataset following with all m2 instances in the next m2 instances etc. before performing k-nearest neighbor with the knearest function.

**Value**

The matrix with the number of columns equal to the number of classes. Each a[i][j] represents the number of K-nearest neighbors of i th instance belonging to the class j th

**Author(s)**

Wacharasak Siriseriwan <wacharasak.s@gmail.com>

**Examples**

```r
D = sample_generator(1000,ratio = 0.8)
P = D[D[,3]=="p",]
N = D[D[,3]=="n",]
D_arr=rbind(P,N)
knear=knearest(D_arr[,-3],P[,-3],5)
kncount_result = kncount(knear,nrow(P))
```

**knearest**

The function to find n_clust nearest neighbors of each instance, always removing the index of that instance if it is reported.

**Description**

The function will find n_clust nearest neighbors of each instance using Fast nearest neighbors (through KD-tree method) but will correct the result if it reports the index of that instance as its neighbors.

**Usage**

```r
knearest(D, P, n_clust)
```

**Arguments**

- **D**
  - a query data matrix.

- **P**
  - an input data matrix

- **n_clust**
  - the maximum number of nearest neighbors to search
Details

This function will perform K-nearest neighbor of instances in P on instances in P based on FNN. Then, it will verify if one of neighbors of each instance is itself then removes if it is.

Value

The index matrix of K nearest neighbour of each instance

Author(s)

Wacharasak Siriseriwan <wacharasak.s@gmail.com>

Examples

data_example = sample_generator(10000, ratio = 0.80)
P = data_example[data_example[,3]=="p",-3]
N = data_example[data_example[,3]=="n",-3]
D = rbind(P,N)
knear = knearest(D,P,n_clust = 5)

n_dup_max

The function to calculate the maximum round each sampling is repeated

Description

The function to calculate the maximum round each sampling is repeated, if dup_size is given as 0 then, it calculates the maximum round the number of positive instances to be duplicated to nearly match the number of negative instances

Usage

n_dup_max(size_input, size_P, size_N, dup_size = 0)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>size_input</td>
<td>The size of overall dataset</td>
</tr>
<tr>
<td>size_P</td>
<td>The number of positive instances</td>
</tr>
<tr>
<td>size_N</td>
<td>The number of negative instances</td>
</tr>
<tr>
<td>dup_size</td>
<td>A number or vector of the number of times to be duplicated.</td>
</tr>
<tr>
<td></td>
<td>The default is zero which means duplicating until nearly</td>
</tr>
<tr>
<td></td>
<td>balanced.</td>
</tr>
</tbody>
</table>

Value

If dup_size is zero or contains zero, the number of rounds to duplicate positive to nearly equal to the number of negative instances If dup_size is not zero or contains no zero, the maximum value in dup_size
Author(s)
Wacharasak Siriseriwan <wacharasak.s@gmail.com>

Examples

data_example = sample_generator(10000,ratio = 0.80)
P = data_example[data_example[,3]=="p",-3]
N = data_example[data_example[,3]=="n",-3]
D = rbind(P,N)
max_round =n_dup_max(nrow(D),nrow(P),nrow(N),dup_size= 0)

RSLS

Relocating Safe-level SMOTE

Description
Generate synthetic positive instances using Relocating Safe-level SMOTE algorithm. Using the parameter “Safe-Level” to determine the possible location and relocating synthetic instances if there is too close to majority instances.

Usage
RSLS(X, target, K = 5, C = 5, dupSize = 0)

Arguments
X A data frame or matrix of numeric-attributed dataset
target A vector of a target class attribute corresponding to a dataset X.
K The number of nearest neighbors during sampling process
C The number of nearest neighbors during calculating safe-level process
dupSize The number or vector representing the desired times of synthetic minority instances over the original number of majority instances

Value
data A resulting dataset consists of original minority instances, synthetic minority instances and original majority instances with a vector of their respective target class appended at the last column
syn_data A set of synthetic minority instances with a vector of minority target class appended at the last column
orig_N A set of original instances whose class is not oversampled with a vector of their target class appended at the last column
orig_P A set of original instances whose class is oversampled with a vector of their target class appended at the last column
12

sample_generator

K
The value of parameter K for nearest neighbor process used for generating data
K_all
The value of parameter C for nearest neighbor process used for calculating safe-level
dup_size
The maximum times of synthetic minority instances over original majority instances in the oversampling
outcast
A set of original minority instances which has safe-level equal to zero and is defined as the minority outcast
eps
Unavailable for this method
method
The name of oversampling method used for this generated dataset (RSLS)

Author(s)
Wacharasak Siriseriwan <wacharasak.s@gmail.com>

References

Examples

library(smotefamily)
data_example = sample_generator(5000, ratio = 0.80)
genData = RLS(data_example[, -3], data_example[, 3])
genData_2 = RLS(data_example[, -3], data_example[, 3], K = 7, C = 5)

Description
The function to generate 2-dimensional dataset given the number of instances and the ratio between the number of negative instances to total instances. The positive instances will be distributed uniformly as the circle in the center while negative instances are around over the domain. The random positive outcasts are also generated. The dataset is used to show the difference between datasets generated by each sampling technique.

Usage

sample_generator(n, ratio = 0.8, xlim = c(0, 1), ylim = c(0, 1),
radius = 0.25, overlap = -0.05, outcast_ratio = 0.01)
**Arguments**

- **n**: The number of instances in the dataset
- **ratio**: The ratio of negative instances to the total number of instances
- **xlim**: The range of values in the first dimension
- **ylim**: The range of values in the second dimension
- **radius**: The radius of the circle of positive instances
- **overlap**: The gap between the set of positive and negative instances
- **outcast_ratio**: The ratio of outcast to be generate in this dataset.

**Value**

A 2-dimensional dataset with the 3rd column as its target class vector.

**Author(s)**

Wacharasak Siriseriwan <wacharasak.s@gmail.com>

**Examples**

```r
data_example = sample_generator(5000, ratio = 0.80)
plot(data_example[data_example[,3]=="n",1],
data_example[data_example[,3]=="n",2],col="yellow")
points(data_example[data_example[,3]=="p",1],
data_example[data_example[,3]=="p",2],col="red",pch=14)
```

---

**SLS**

*Safe-level SMOTE*

**Description**

Generate synthetic positive instances using Safe-level SMOTE algorithm. Using the parameter "Safe-level" to determine the possible location of synthetic instances.

**Usage**

```
SLS(X, target, K = 5, C = 5, dupSize = 0)
```

**Arguments**

- **X**: A data frame or matrix of numeric-attributed dataset
- **target**: A vector of a target class attribute corresponding to a dataset X.
- **K**: The number of nearest neighbors during sampling process
- **C**: The number of nearest neighbors during calculating safe-level process
- **dupSize**: The number or vector representing the desired times of synthetic minority instances over the original number of majority instances
Value

- **data**: A resulting dataset consists of original minority instances, synthetic minority instances and original majority instances with a vector of their respective target class appended at the last column.
- **syn_data**: A set of synthetic minority instances with a vector of minority target class appended at the last column.
- **orig_N**: A set of original instances whose class is not oversampled with a vector of their target class appended at the last column.
- **orig_P**: A set of original instances whose class is oversampled with a vector of their target class appended at the last column.
- **K**: The value of parameter K for nearest neighbor process used for generating data.
- **K_all**: The value of parameter C for nearest neighbor process used for calculating safe-level.
- **dup_size**: The maximum times of synthetic minority instances over original majority instances in the oversampling.
- **outcast**: A set of original minority instances which has safe-level equal to zero and is defined as the minority outcast.
- **eps**: Unavailable for this method.
- **method**: The name of oversampling method used for this generated dataset (SLS).

Author(s)

Wacharasak Siriseriwan <wacharasak.s@gmail.com>

References


Examples

```python
data_example = sample_generator(5000,ratio = 0.80)
genData = SLS(data_example[,-3],data_example[,3])
genData_2 = SLS(data_example[,-3],data_example[,3],K=7, C=5)
```

Description

Generate synthetic positive instances using SMOTE algorithm.
SMOTE

Usage

SMOTE(X, target, K = 5, dup_size = 0)

Arguments

X
A data frame or matrix of numeric-attributed dataset

target
A vector of a target class attribute corresponding to a dataset X.

K
The number of nearest neighbors during sampling process

dup_size
The number or vector representing the desired times of synthetic minority instances over the original number of majority instances

Value

data
A resulting dataset consists of original minority instances, synthetic minority instances and original majority instances with a vector of their respective target class appended at the last column

syn_data
A set of synthetic minority instances with a vector of minority target class appended at the last column

orig_N
A set of original instances whose class is not oversampled with a vector of their target class appended at the last column

orig_P
A set of original instances whose class is oversampled with a vector of their target class appended at the last column

K
The value of parameter K for nearest neighbor process used for generating data

K_all
Unavailable for this method

dup_size
The maximum times of synthetic minority instances over original majority instances in the oversampling

outcast
Unavailable for this method

eps
Unavailable for this method

method
The name of oversampling method used for this generated dataset (SMOTE)

Author(s)

Wacharasak Siriseriwan <wacharasak.s@gmail.com>

References


Examples

data_example = sample_generator(10000,ratio = 0.80)
genData = SMOTE(data_example[, -3], data_example[, 3])
genData_2 = SMOTE(data_example[, -3], data_example[, 3], K = 7)
Index

* manip
  ADASYN, 3
  ANS, 4
  Borderline-SMOTE, 5
  DBSMOTE, 6
  RSL, 11
  SLS, 13
  SMOTE, 14
* methods
  ADASYN, 3
  ANS, 4
  Borderline-SMOTE, 5
  DBSMOTE, 6
  RSL, 11
  SLS, 13
  SMOTE, 14
* package
  smotefamily-package, 2

ADAS (ADASYN), 3
ADASYN, 3
ANS, 4

BLSMOTE (Borderline-SMOTE), 5
Borderline-SMOTE, 5

DBSMOTE, 6
gap, 8
kncount, 8
knearest, 9
n_dup_max, 10

RSL, 11

sample_generator, 12
SLS, 13
SMOTE, 14
smotefamily (smotefamily-package), 2
smotefamily-package, 2