Geometric SMOTE
A geometrically enhanced drop-in replacement for SMOTE

Georgios Douzas[1], Fernando Bacao[1]*

[1]NOVA Information Management School, Universidade Nova de Lisboa

*Corresponding Author

Postal Address: NOVA Information Management School, Campus de Campolide, 1070-312 Lisboa, Portugal

Telephone: +351 21 382 8610

Classification of imbalanced datasets is a challenging task for standard algorithms. Although many methods exist to address this problem in different ways, generating artificial data for the minority class is a more general approach compared to algorithmic modifications. SMOTE algorithm, as well as any other oversampling method based on the SMOTE mechanism, generates synthetic samples along line segments that join minority class instances. In this paper we propose Geometric SMOTE (G-SMOTE) as a enhancement of the SMOTE data generation mechanism. G-SMOTE generates synthetic samples in a geometric region of the input space, around each selected minority instance. While in the basic configuration this region is a hyper-sphere, G-SMOTE allows its deformation to a hyper-spheroid. The performance of G-SMOTE is compared against SMOTE as well as baseline methods. We present empirical results that show a significant improvement in the quality of the generated data when G-SMOTE is used as an oversampling algorithm. An implementation of G-SMOTE is made available in the Python programming language.

1 Introduction

Learning from imbalanced data is a non trivial and important problem for the research community and the industry practitioners [Chawla et al., 2003]. An imbalanced learning problem is defined as a classification task for binary or multi-class datasets where a significant asymmetry exists between the number of instances for the various classes. The dominant class is called the majority class while the rest of the classes are called the minority classes [Chawla et al., 2003]. The Imbalance Ratio (IR), defined as the ratio between the majority class and each of the minority classes, depends on the type of application and for binary problems values between 100 and 100.000 have been observed [Chawla et al., 2002], [Barua et al., 2014].

The imbalance learning problem can be found in numerous practical domains, such as chemical and biochemical engineering, financial management, information technology, security, business, agriculture or emergency management, for a more in depth review the reader is referred to [Haixiang et al., 2017]. Standard learning methods induce a bias in favor of the majority class during training. This happens because the minority classes contribute less to the minimization of the objective function, defined often as the classification accuracy. Additionally, the distinction between noisy and minority class instances is frequently difficult. As a result the performance of the classifiers, evaluated on metrics suitable for imbalanced data, is low. It is also important to consider that the costs of misclassifying the minority class are frequently much higher than the costs of misclassification of the majority class [Domingos, 1999], [Ting, 2002]. Diseases screening tests are a typical situation in in which false negatives involve a much higher cost than the false positives. Therefore, fundamentally the class imbalance challenge is to propose smart and simple ways is to improve the accuracy of classifiers for the minority class.

We can classify the approaches to deal with class imbalance into three main groups [Fernández et al., 2013]. The first consists in the modification or creation of algorithms that reinforce the learning towards the minority class. The second is the application of cost-sensitive methods to minimize higher cost errors. The last and more general approach involves the modification at the data level by re-balancing the class distribution. This is usually done through the use of undersampling, oversampling or hybrid methods.

Our focus in this paper is oversampling techniques, which result in the generation of artificial data for the minority class. Synthetic Minority Oversampling Technique (SMOTE) [Chawla et al., 2002] is the most popular algorithm in this category. SMOTE can be decomposed into two parts: a set of selection rules for the minority class instances and a data generation mechanism once these samples are selected. Specifically, the selection phase constitutes of a process that repeatedly identifies one minority class sample and a random minority class k-nearest neighbor of it while the data generation mechanism creates synthetic examples along the line segment that joins them. Most existing variations of the SMOTE algorithm modify the selection phase by imposing a set of heuristic rules. Contrary to this, the method proposed in this paper, G-SMOTE, substitutes the data generation mechanism by defining a flexible geometric region around each minority class instance. Then synthetic instances are generated inside the boundaries of the region which are controlled by an appropriate parametrization of the algorithm.

For the evaluation of G-SMOTE as an oversampling method an experimental analysis is performed. The selected imbalanced datasets are publicly available from the UCI [Lichman, 2013] and KEEL [Alcalá-Fdez et al., 2011] repositories. In order to test the performance of the algorithm on more extreme cases of imbalance, undersampled modifications of the aforementioned datasets as well as simulated binary class imbalanced data are provided in a total of 69 datasets. Since this work aims to show that G-SMOTE is an enhanced generalization of SMOTE, the experimental procedure includes a comparison between the two algorithms using 4 classifiers and 3 appropriate evaluation metrics. Additionally, applying no oversampling and Random Oversampling are included as baseline methods.

The sections in the paper are organized as follows. In section 2, an overview of related previous works and existing sampling methods is given. In Section 3, the motivation for G-SMOTE is presented, while section 4 describes the proposed method in detail. The experimental results as well as conclusions from their analysis are presented in section 5.

2 Related work

In this section we provide a brief review of the most popular oversampling methods. The reader interested in undersampling and hybrid methods is referred to [Galar et al., 2012], [Chawla, 2005] and [Fernandez et al., 2018]. The fundamental idea of oversampling methods consists in the generation of synthetic examples for the minority class, which should then be added to the training set. The simplest approach, Random Oversampling, duplicates randomly selected minority class instances. The disadvantage of this approach is that the exact replication of training examples can increase the risk of over-fitting since no new information is created and the classifier will use the same information. An alternative approach that aims to reduce this problem and generate new data is SMOTE, which allows for the generation of synthetic instances along a line segment that joins original minority class instances. SMOTE has been the first and most popular oversampling algorithm. Although SMOTE has been shown to be an effective and simple option for oversampling it also has some weaknesses, such as the fact that the separation between majority and minority class clusters is not often clear and the generation of noisy instances [He and Garcia, 2009]. In order to mitigate these problems many variations to SMOTE have been proposed.

2.1 Modifications of the selection phase

Imbalance learning can be divided into two different problems: between-class imbalance and within-class imbalance [Jo and Japkowicz, 2004]. The between-class imbalance refers to the classical problem of the skewness in the distribution between majority and minority classes. The within-class imbalance is a subtler, but equally relevant, problem and refers to the possible existence of dense or sparse sub-clusters of minority or majority instances. Both of these problems are relevant in imbalance learning.

SMOTE + Edited Nearest Neighbor [Batista et al., 2004] is and example of a between-class imbalance algorithm, which modifies the selection phase of the SMOTE algorithm. SMOTE + Edited Nearest Neighbor combination starts by generating artificial instances using SMOTE and then applies the edited nearest neighbor rule [Wilson, 1972] to remove misclassified instances, based on the classification by its three nearest neighbors. Borderline-SMOTE [Han et al., 2005], MWMOTE (Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning) [Barua et al., 2014], ADASYN and its variation KernelADASYN [Tang and He, 2015] share the same overall objective, which is to prevent the generation of noisy instances through the identification of the borderline instances for both, majority and minority classes, which, in turn, are used to identify the informative minority class instances.

The typical approach to deal with the within-class imbalance problem is to apply some type of clustering procedure, for the identification of eventual minority or majority sub-clusters, followed by the application of sampling methods, such as SMOTE, to correct the size of different clusters. Cluster-SMOTE [Cieslak et al., 2006] applies the k-means algorithm and then generates artificial data using SMOTE. Similarly DBSMOTE [Bunkhumpornpat et al., 2012] uses DB-SCAN, density-based, algorithm to identify arbitrarily shaped clusters and generates synthetic instances along a shortest path from each minority class instance to a pseudo-centroid of the cluster. A-SUWO [Nekooeimehr and Lai-Yuen, 2016] creates clusters of the minority class instances with a size, which is determined using cross validation and generates synthetic instances based on a proposed weighting system. SOMO [Douzas and Bacao, 2017b] creates a two dimensional representation of the input space (U-matrix) and based on it, applies the SMOTE procedure to generate intra-cluster and inter-cluster synthetic instances that preserve the underlying manifold structure. Similarly to SOMO, a combination of k-means and SMOTE [Douzas et al., 2018] can be applied to re-balance the class distribution based on the density of the identified clusters. Finally, other types of oversampling approaches are based on ensemble methods [Wang et al., 2015], [Sun et al., 2015] such as SMOTEBoost [Chawla et al., 2003] and DataBoost-IM [Guo and Viktor, 2004].

2.2 Modifications of the data generation mechanism

The modification or substitution of the SMOTE data generation mechanism is a less common approach. Safe-Level SMOTE [Bunkhumpornpat et al., 2009] applies a weight degree, the safe level, in the data generation process. Based on the safe level, another quantity called the safe level ratio is calculated, that effectively truncates a part of the line segment joining minority class samples. Therefore the Safe-Level SMOTE modification acts as a restriction of the SMOTE data-generation mechanism. A different type of oversampler, that completely substitutes the SMOTE data generation mechanism, applies the Conditional Generative Adversarial Networks (CGAN) [Douzas and Bacao, 2017a] to generate data for the minority class. Contrary to the other methods, CGAN oversampler does not rely on local information of the input space but aims to approximate directly the true data distribution.

3 Motivation

In the previous section various informative oversampling methods were presented as an effective way to re-balance the data distribution. However, there are scenarios where the SMOTE data generation mechanism may encounter a variety of problems. This section describe some of these cases providing suitable examples and motivates the proposed G-SMOTE algorithm. In what follows x is the initially selected minority class instance of the SMOTE mechanism. Some of the SMOTE algorithm inefficiencies are the following:

1. *Generation of noisy instances due to the selection of k-nearest neighbors.*

In SMOTE, the value of k is determined in advance and there are cases where the results of oversampling are sensitive to it as it is shown in the following example. The decision boundary of Fig. 1 identifies areas of the input space where instances from either the positive or negative class dominate. A few of the minority class instances, called noisy observations, that are located near the decision boundary, penetrate into the majority class area. A large k value can result in the generation of additional noisy examples since x', the selected k-nearest neighbor of x, might be one of the aforementioned noisy observations.

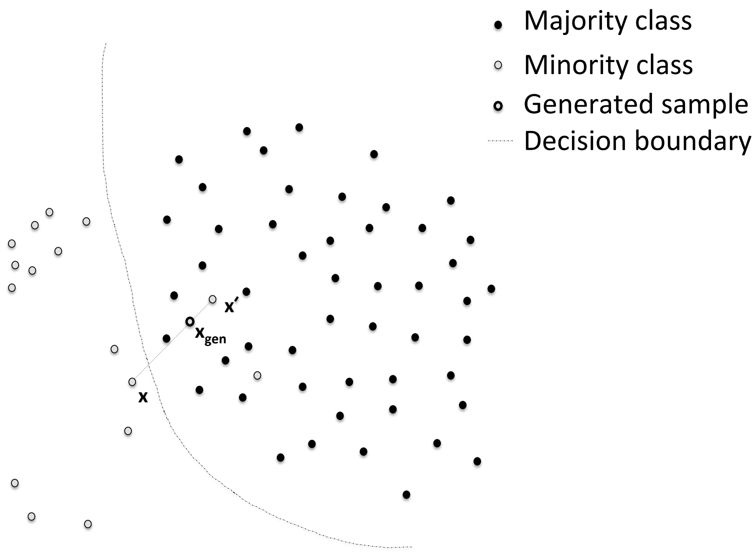

Figure 1: An instance near the decision boundary and one of its 4-nearest neighbors are selected randomly. A noisy observation is generated.

2. *Generation of noisy examples due to the selection of an initial observation.*

In order to avoid the previous scenario k is set to a small value. This choice does not eliminate the generation of noisy samples when x itself is a noisy instance as can be seen in Fig. 2.

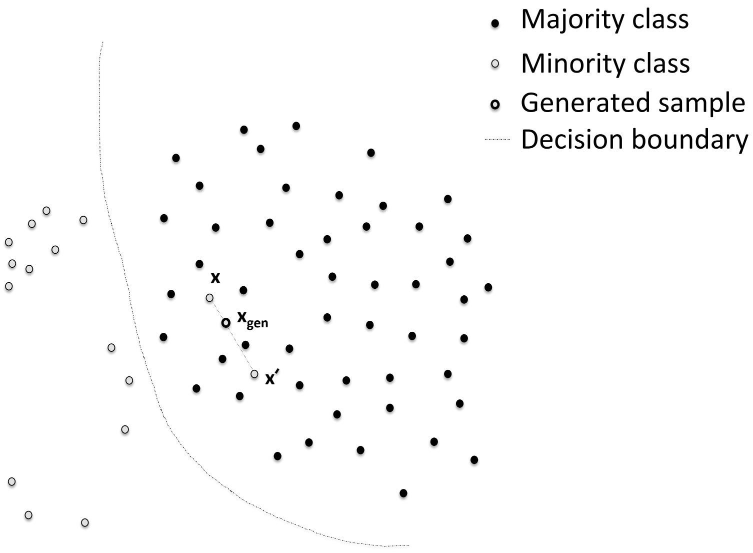

Figure 2: A 2-nearest neighbor does not avoid the generation of noise when noisy samples are initially selected.

3. *Generation of nearly duplicated instances.*

Minority and majority regions can be organized in different clusters, having complex decision boundaries. In this case, being conservative in the selection of k may be a valid strategy since a small value of it can minimize the probability of generating noisy instances. On the other hand, it may increase the probability of generating synthetic instances in dense minority class areas as is shown in Fig 3, where x and x' belong in to the same cluster. These artificial samples are less useful because they do not add new information to the data set and are conducive to overfitting. Consequently, it is desirable to expand the data generation process in areas where minority examples are absent.

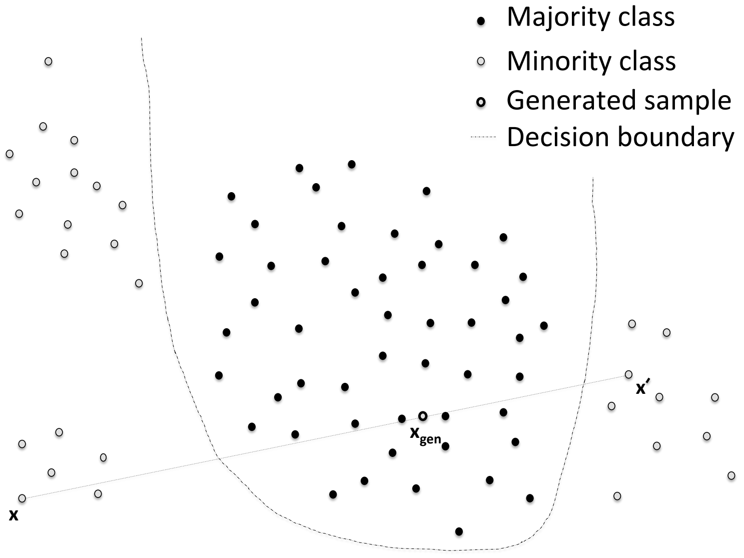

Figure 3: An instance belonging to a minority class cluster and one of its 5-nearest neighbors are selected. An observation belonging to the same cluster is generated.

4. *Generation of noisy instances due to the use of observations from two different minority class clusters.*

Increasing k in order to avoid the previous scenario, may result to a selected x' such that x and x' belong to different clusters. This in turn may lead to the generation of a new instance within the majority region as it is exemplified in Fig. 4.

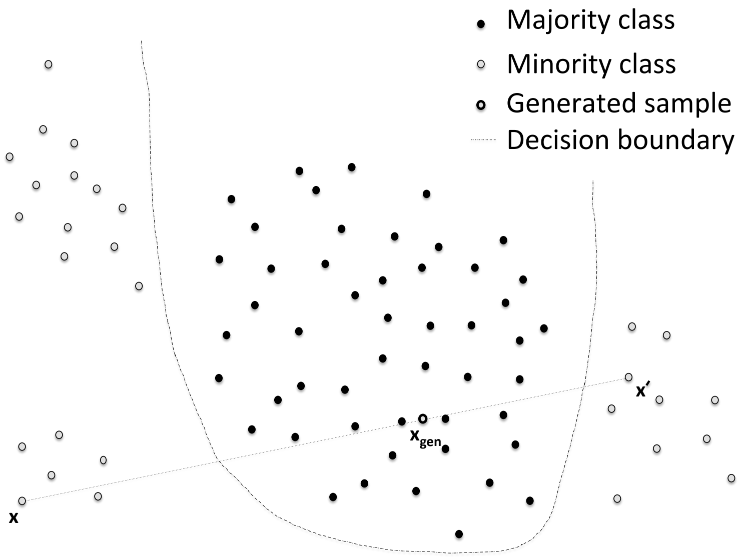

Figure 4: An attempt to generate inter-cluster instances by increasing the number of k-nearest neighbors of the selected instances. One of the generated instances penetrates in the majority class area.

4 The proposed method

In the previous section some insufficiencies of the SMOTE data generation mechanism in various scenarios were described. Some of these insufficiencies apply also to other SMOTE-based oversamplers. We propose a novel data generation procedure, G-SMOTE, which is an extension of the SMOTE algorithm and has three main objectives:

1. *To define a safe area around each selected minority class instance such that the generated artificial minority instances inside this area are not noisy.*

2. *To increase the variety of generated samples by expanding the minority class area.*

3. *To parametrize the above characteristics based on a small number of transformations with a geometrical interpretation.*

G-SMOTE can be considered as a drop-in replacement for SMOTE in the sense that any method relying on the SMOTE data generation mechanism can replace it with the one proposed by G-SMOTE without any further modifications. As it was mentioned on the previous sections, in this paper we aim to a direct comparison of the two data generation mechanisms, with the effect of the above replacement being an investigation of future work.

The complete algorithm in pseudo-code is presented in the following figure:

```
Algorithm 1: G-SMOTE

Input: $S_{maj}, S_{min}, N, k, \alpha_{sel}, \alpha_{trunc}, \alpha_{def}$
Output: $S_{gen}$

Function Surface($\alpha_{sel}$, $\boldsymbol{x}_{center}$, $S_{maj}$, $S_{min}$):
    if $\alpha_{sel} = minority$ then
        $\mathbf{x}_{surface} \in S_{min,k}$
    else if $\alpha_{sel} = majority$ then
        $\mathbf{x}_{surface} \in S_{maj,1}$
    else if $\alpha_{sel} = combined$ then
        $\mathbf{x}_{min} \in S_{min,k}$
        $\mathbf{x}_{maj} \in S_{maj,1}$
        $\mathbf{x}_{surface} \leftarrow argmin_{\mathbf{x}_{min},\mathbf{x}_{maj}} \Big( \|\mathbf{x}_{center} - \mathbf{x}_{min}\|, \|\mathbf{x}_{center} - \mathbf{x}_{maj}\| \Big)$
    return $\mathbf{x}_{surface}$

Function Vectors($\boldsymbol{x}_{center}$, $\boldsymbol{x}_{surface}$):
    $\mathbf{e}_{//} \leftarrow \frac{\mathbf{x}_{surface} - \mathbf{x}_{center}}{\|\mathbf{x}_{surface} - \mathbf{x}_{center}\|}$
    $\mathbf{x}_{//} \leftarrow (\mathbf{x}_{gen} \cdot \mathbf{e}_{//})\ \mathbf{e}_{//}$
    $\mathbf{x}_{\perp} \leftarrow \mathbf{x}_{gen} - \mathbf{x}_{//}$
    return $(\mathbf{x}_{//}, \mathbf{x}_{\perp})$

Function Hyperball():
    $v_i \sim N(0,1)$
    $r \sim U(0,1)$
    $\mathbf{x}_{gen} \leftarrow r^{1/p} \frac{(v_1, \cdots, v_p)}{\|(v_1, \cdots, v_p)\|}$
    return $\mathbf{x}_{gen}$

Function Truncate($\alpha_{trunc}, \boldsymbol{x}_{gen}, \boldsymbol{x}_{center}, \boldsymbol{x}_{surface}$):
    if $|\alpha_{trunc} - x_{//}| > 1$ then
        $\mathbf{x}_{gen} \leftarrow \mathbf{x}_{gen} - 2\mathbf{x}_{//}$
    return $\mathbf{x}_{gen}$

Function Deform($\alpha_{dist}, \boldsymbol{x}_{gen}, \boldsymbol{x}_{center}, \boldsymbol{x}_{surface}$):
    return $\mathbf{x}_{gen} - \alpha_{def}\ \mathbf{x}_{\perp}$

Function Translate($\boldsymbol{x}_{gen}, \boldsymbol{x}_{center}, R$):
    return $\mathbf{x}_{center} + R\ \mathbf{x}_{gen}$

begin
    $S_{gen} = \emptyset$
    while $|S_{gen}| < N$ do
        $\mathbf{x}_{center} \in\ S_{min}$
        $\mathbf{x}_{surface} \leftarrow$ Surface($\alpha_{sel}$, $\boldsymbol{x}_{center}$, $S_{maj}$, $S_{min}$)
        $(\mathbf{x}_{//}, \mathbf{x}_{\perp}) \leftarrow$ Vectors( $\boldsymbol{x}_{center}$, $\boldsymbol{x}_{surface}$)
        $\mathbf{x}_{gen} \leftarrow$ Hyperball()
        $\mathbf{x}_{gen} \leftarrow$ Truncate($\alpha_{trunc}, \boldsymbol{x}_{gen}, \boldsymbol{x}_{center}, \boldsymbol{x}_{surface}$)
        $\mathbf{x}_{gen} \leftarrow$ Deform( $\alpha_{dist}, \boldsymbol{x}_{gen}, \boldsymbol{x}_{center}, \boldsymbol{x}_{surface}$)
        $\mathbf{x}_{gen} \leftarrow$ Translate( $\boldsymbol{x}_{gen}, \boldsymbol{x}_{center}, \|\boldsymbol{x}_{center} - \boldsymbol{x}_{surface}\|$)
        $S_{gen} \leftarrow S_{gen} \cup \{\mathbf{x}_{gen}\}$
```

4.1 G-SMOTE algorithm

The inputs of the G-SMOTE algorithm are the following:

- The sets S_{maj}, S_{min} of majority and minority class samples, respectively.
- The total number N of synthetic samples to be generated.
- The number k of nearest neighbors.
- The neighbor selection strategy α_{sel} with $\alpha_{sel} \in \Big\{minority, majority, combined\Big\}$.
- The truncation factor α_{trunc} with $-1 \leq \alpha_{trunc} \leq 1$.
- The deformation factor α_{def} with $0 \leq \alpha_{def} \leq 1$.

The output of G-SMOTE is the set S_{gen} of generated synthetic examples.

The algorithmic procedure is the following:

- The S_{min} elements are shuffled and an empty set S_{gen} is initialized.
- The following loop is repeated until N minority instances are selected, each multiple times if necessary, in the order that appear in S_{min}:
    - Let $\mathbf{x}_{center} \in S_{min}$ the selected minority class instance of p components.
    - A surface vector is defined from the relation $\mathbf{x}_{surface} \leftarrow$ `Surface(`α_{sel}, $\boldsymbol{x}_{center}$, S_{maj}, S_{min}`)` .
    - A set of direction vectors $(\mathbf{x}_{//}, \mathbf{x}_{\perp}) \leftarrow$ `Vectors(` $\boldsymbol{x}_{center}$, $\boldsymbol{x}_{surface}$`)` is extracted.
    - A synthetic sample $\mathbf{x}_{gen} \leftarrow$ `Hyperball()` is generated.
    - The transformation functions `Truncate`, `Deform` and `Translate` are applied to $\mathbf{x}_{gen}$.
    - $\mathbf{x}_{gen}$ is added to S_{gen}.

4.2 Functions

The above algorithmic procedure relies on the following definitions of functions:

- Function `Surface`:

  The sets $S_{min,k}$ and $S_{maj,1}$ are the sets of k and $k = 1$ nearest neighbors of $\mathbf{x}_{center}$ from S_{min} and S_{maj}, respectively.

  If $\alpha_{sel} = minority$ then an element $\mathbf{x}_{surface} \in S_{min,k}$ is randomly selected.

  If $\alpha_{sel} = majority$ then an element $\mathbf{x}_{surface} \in S_{maj,1}$ is randomly selected.

If $\alpha_{sel} = combined$ then $S_{min,k}$ and $S_{maj,1}$ are defined as above and the elements $\mathbf{x}_{min} \in S_{min,k}$ and $\mathbf{x}_{maj} \in S_{maj,1}$ are randomly selected. Finally the element $\mathbf{x}_{surface}$ is defined as either $\mathbf{x}_{min}$ or $\mathbf{x}_{maj}$ by selecting the one with smallest distance from $\mathbf{x}_{center}$.

– Function `Hyperball`:

A vector $\mathbf{v}_{normal} \leftarrow (v_1, \cdots, v_p)$ is generated of p random numbers from the normal distribution $N(0,1)$. The unit vector $\mathbf{e}_{sphere} \leftarrow \frac{\mathbf{v}_{normal}}{\|\mathbf{v}_{normal}\|}$ (1) and the vector $\mathbf{x}_{gen} \leftarrow r^{1/p}\mathbf{e}_{sphere}$ (2) are calculated where r is a random number from the uniform distribution $U(0,1)$.

– Function `Vectors`:

The unit vector $\mathbf{e}_{//} \leftarrow \frac{\mathbf{x}_{surface} - \mathbf{x}_{center}}{\|\mathbf{x}_{surface} - \mathbf{x}_{center}\|}$ (3) and the projection $x_{//} = \mathbf{x}_{gen} \cdot \mathbf{e}_{//}$ are defined. Using this projection, the vectors $\mathbf{x}_{//} \leftarrow x_{//}\mathbf{e}_{//}$ (4) and $\mathbf{x}_{\perp} \leftarrow \mathbf{x}_{gen} - \mathbf{x}_{//}$ are also defined.

– Function `Truncate`:

If the relation $|\alpha_{trunc} - x_{//}| > 1$ (5) holds then the transformation $\mathbf{x}_{gen} \leftarrow \mathbf{x}_{gen} - 2\mathbf{x}_{//}$ (6) is applied.

– Function `Deform`:

The transformation $\mathbf{x}_{gen} \leftarrow \mathbf{x}_{gen} - \alpha_{def}\ \mathbf{x}_{\perp}$ (7) is applied.

– Function `Translate`:

The transformation $\mathbf{x}_{center} + R\ \mathbf{x}_{gen}$ (8) is applied.

4.3 Justification of the G-SMOTE algorithm

As explained above, SMOTE compared to Random Oversampling, improves the diversity of generated samples by linearly interpolating generated samples between two minority class instances. However on high-dimensional data SMOTE does not change the class-specific mean values while it decreases the data variability and it introduces correlation between samples [Blagus and Lusa, 2013]. Contrary to this, G-SMOTE extends the linear interpolation mechanism by introducing a geometric region where the data generation process occurs. At the most general choice of hyper-parameters, this geometric region of the input space is a truncated hyper-spheroid. The various steps of the G-SMOTE algorithm can be described in detail as follows:

1. An empty set S_{gen} is initialized.

2. The S_{min} elements are shuffled and the process described below is repeated N times until N artificial points have been generated.

3. A minority class instance $\mathbf{x}_{center}$ is selected as the center of a geometric region. The order of selection follows the order of dataset points after shuffling on the previous step. Therefore if N is greater than S_{min}, then some of the minority class samples will be selected more than once.

4. At first glance, this step generalizes the selection phase of the SMOTE algorithm. More specifically, it results to a randomly selected sample called $\mathbf{x}_{surface}$ which might belong to either the minority or majority class, depending on the values of α_{sel}, k and $\mathbf{x}_{center}$. However, we consider it as a part

of the G-SMOTE data generation mechanism since it does not just filters the selected minority class instances based on heuristic rules similarly to SMOTE variations. Based on the neighbor selection strategy α_{sel}, we distinguish 3 different cases:

a) Case $\alpha_{sel} = minority$:

In this case the neighbor selection strategy is based only on the minority class and it is identical to the selection strategy of SMOTE. Initially the k nearest neighbors of $\mathbf{x}_{center}$ from the set S_{min} are identified and one of them, $\mathbf{x}_{surface}$, is randomly selected. Fig. 5 presents an example of a minority class instance selection among the $k = 4$ nearest neighbors of $\mathbf{x}_{center}$. The time complexity of this selection strategy depends on the choice of the algorithm and increases with the dimensionality of the input space as well as the value of the k parameter [Vaidya, 1989]. Therefore for a wide set of realistic cases restricting the search of nearest neighbors to the minority class has a lower computational cost than including the majority class instances in the search space.

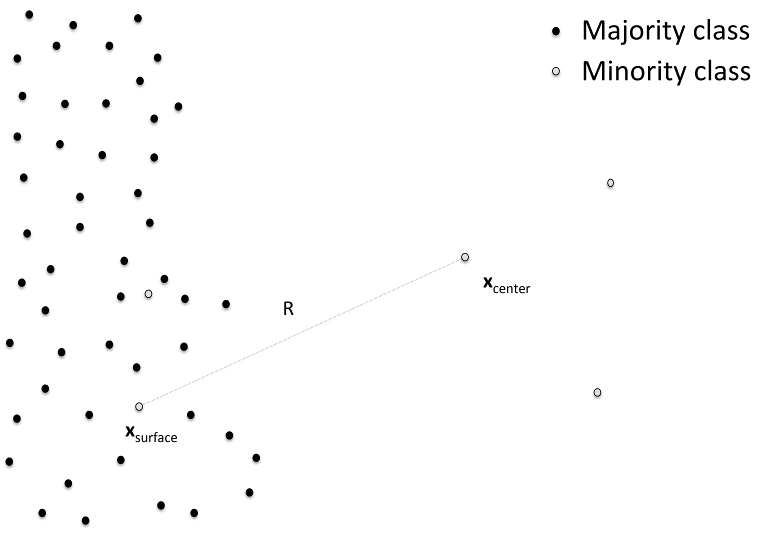

Figure 5: An example of the minority selection strategy. A minority class instance is defined as the center of the hyper-spheroid and one of its $k = 4$ minority class nearest neighbors is selected as the surface point. The radius R of the hyper-spheroid is defined to be equal to the distance of these minority instances.

b) Case $\alpha_{sel} = majority$:

As explained in section 3, one of the drawbacks of the minority selection strategy is that it may lead to the generation of data penetrating deeply in the majority class area. The majority selection strategy eliminates this scenario. More specifically, the nearest neighbor of $\mathbf{x}_{center}$ from the set S_{maj} is identified as $\mathbf{x}_{surface}$. The consequence of this selection is that when a random minority class point is generated inside a hypersphere of center $\mathbf{x}_{center}$ and radius $R = \|\mathbf{x}_{center} - \mathbf{x}_{surface}\|$, it is ensured that its distance from $\mathbf{x}_{center}$ is not higher than the distance between $\mathbf{x}_{center}$ and any majority class instance. On the other hand, since any information about the minority class is discarded, this strategy might aggressively expand the minority class area, resulting effectively to noise generation. Fig. 6 presents an example of the nearest majority class instance selection among the majority class neighbors of $\mathbf{x}_{center}$. A disadvantage of the majority selection strategy is that the computational cost compared to the minority selection strategy may be higher, especially for datasets with high IR values.

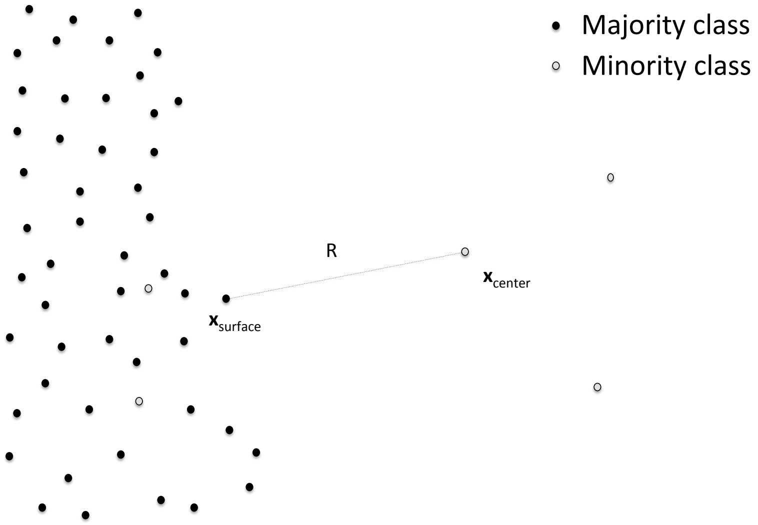

Figure 6: An example of the majority selection strategy. A minority class instance is defined as the center of the hyper-spheroid and its closest majority class neighbor is selected as the surface point. The radius R of the hyper-spheroid is defined to be equal to the distance of these instances.

c) Case $\alpha_{sel} = combined$:

The combined selection strategy initially applies the minority and majority selection strategies, identifying $\mathbf{x}_{min}$ and $\mathbf{x}_{maj}$ as the selected minority and majority class instances, respectively. The surface point $\mathbf{x}_{surface}$ is defined to be either $\mathbf{x}_{min}$ or $\mathbf{x}_{maj}$, so that its distance from the center $\mathbf{x}_{center}$ is minimized. Fig. 7 and Fig. 8 present both of these scenarios i.e. when $\mathbf{x}_{surface}$ is identified either as a minority or majority class instance. Following the combined selection strategy, the expansion of the minority class area relative to the selected as a center minority class sample is restricted by the nearest majority class neighbor of the center, ensuring that the generation of noisy samples is avoided. Contrary to pure majority selection strategy, the expansion is not only safe but it is further restricted by the presence of minority class instances. The drawback of the combined, similarly to the majority selection strategy, is that it has a higher computational cost compared to the SMOTE/minority selection strategy.

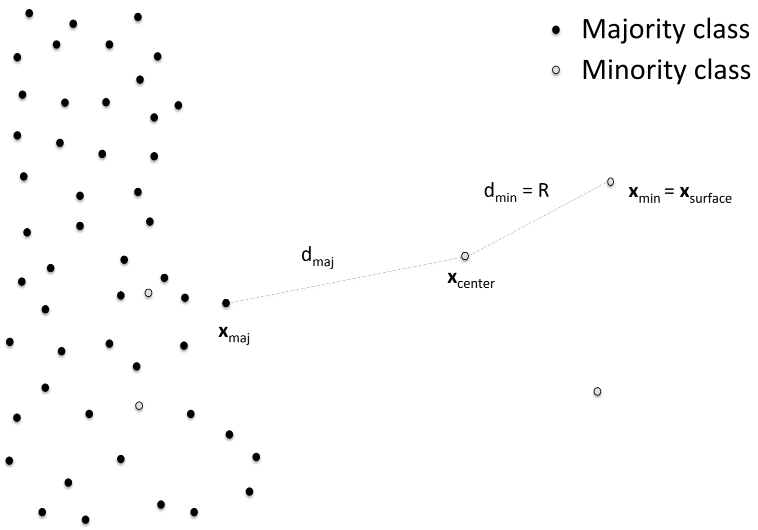

Figure 7: A minority class sample is identified as the surface point since it is closer to the center than the nearest majority class instance.

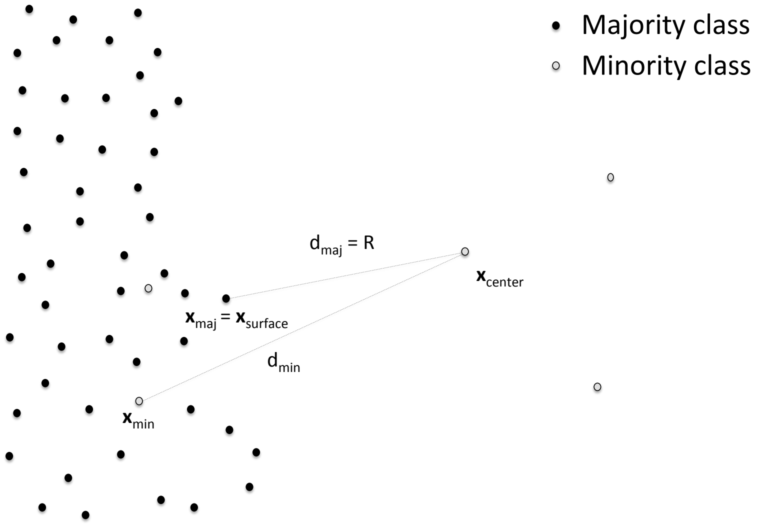

Figure 8: The closest to the center majority class sample is identified as the surface point since it is closer to the center than the selected instance from the k nearest minority class neighbors of the center.

5. Two special directions in the input space are generated: $\mathbf{x}_{//}$ and $\mathbf{x}_{\perp}$. The first one represents the projection of $\mathbf{x}_{gen}$ to the unit vector $\mathbf{e}_{//}$ of equation (3) that connects $\mathbf{x}_{center}$ to $\mathbf{x}_{surface}$, while the second is perpendicular to the same vector belonging also to the hyperplane defined by $\mathbf{x}_{gen}$ and $\mathbf{e}_{//}$.

6. This step starts the data generation process. A random point $\mathbf{e}_{sphere}$ is generated on the surface of a unit hyper-sphere centered at the origin of the input space, using equation (1). Applying equation (2), the point $\mathbf{e}_{sphere}$ is transformed to a random generated point $\mathbf{x}_{gen}$ inside the unit

hyper-sphere. The final result of this process is a random generated point, uniformly distributed, within the unit hyper-sphere [DasGupta, 2011]. Fig. 9 shows an example in two dimensions.

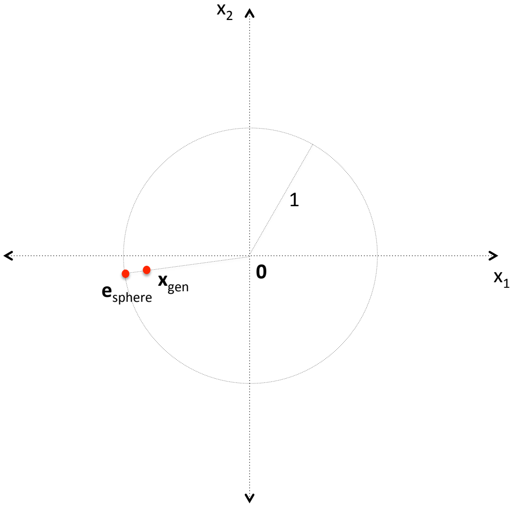

Figure 9: A unit hyper-sphere centered at the origin of the input space. A point is randomly generated on the surface and moved to the interior of the unit hyper-sphere.

7. In this step a transformation is applied to the generated point $\mathbf{x}_{gen}$. As it was explained above, the center $\mathbf{x}_{center}$ and the selected surface point $\mathbf{x}_{surface}$ define a special direction in the input space which is represented by the unit vector $\mathbf{e}_{//}$ of equation (3). SMOTE mechanism, that always selects a minority class instance as a surface point, exploits this direction by generating synthetic samples at the line segment between $\mathbf{x}_{center}$ and $\mathbf{x}_{surface}$. G-SMOTE algorithm parametrizes a generalized version of the SMOTE mechanism. More specifically, the unit vector $\mathbf{e}_{//}$ defines a family of parallel hyper-planes which are perpendicular to it. We define a linear mapping between α_{trunc} and the point determined by the intersection of each hyper-plane and the parallel to $\mathbf{e}_{//}$ diameter. Therefore each one of these hyper-planes corresponds to a particular value of α_{trunc} and partitions the hyper-sphere interior in to two areas. Let P the hyper-plane that passes through the origin and P' the hyper-plane for a specific non-zero value of α_{trunc}. When $a_{trunc} > 0$, the area that does not include the $\mathbf{e}_{//}$ point is truncated from the interior of the hyper-sphere, in the sense that if the $\mathbf{x}_{gen}$ point belongs to it then it is mapped with respect to P to the symmetric point $\mathbf{x}_{gen} - 2\mathbf{x}_{//}$ of equation (6), where $\mathbf{x}_{//}$ is defined in equation (4). Condition (5) checks if the $\mathbf{x}_{gen}$ is in the truncated area. Fig. 10 shows an example of the above transformation. When $\alpha_{trunc} < 0$, the transformation is similarly defined but in this case the truncation occurs in the area that includes the $\mathbf{e}_{//}$ point. In both cases, the absolute value of the hyper-parameter α_{trunc} controls the extent of the truncation. Fig. 11 presents the truncated hyper-sphere areas for various positive and negative values of α_{trunc}. A final observation is that the above transformation effectively corresponds to a modification of the initial uniform probability distribution in the hyper-sphere. The truncated area acquires a zero value for the p.d.f., while its P-symmetric mapped area doubles its initial p.d.f. value.

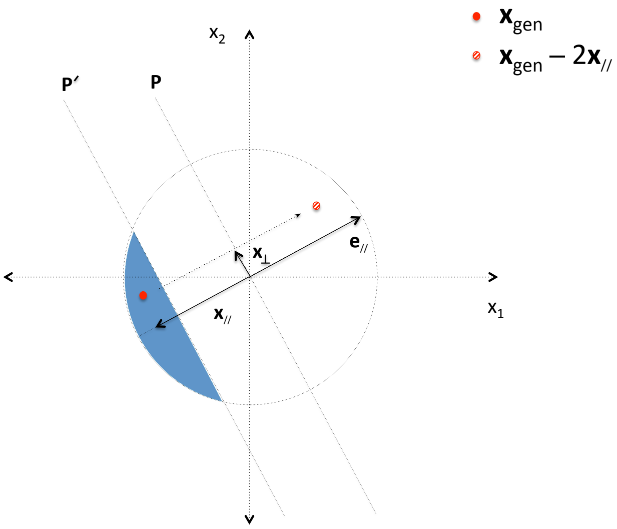

Figure 10: An example of applying the `Truncate` transformation. The shaded area corresponds to the resulting truncated area.

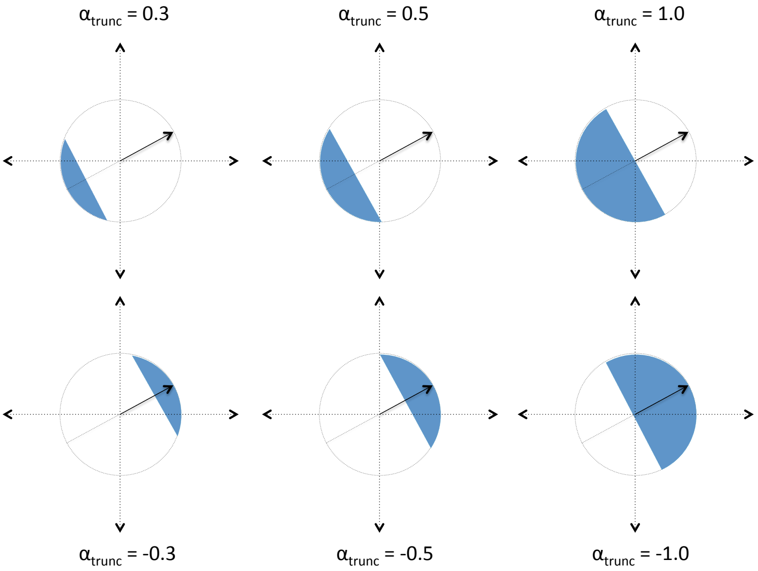

Figure 11: Truncated areas for various values of α_{trunc}.

8. This step describes a transformation that corresponds to the deformation of the hyper-sphere in to a hyper-spheroid. More concretely, the point $\mathbf{x}_{gen}$ is moved to a perpendicular direction to the unit vector $\mathbf{e}_{//}$, towards the parallel to $\mathbf{e}_{//}$ diameter. This mapping is controlled by the α_{def} hyper-parameter and from equation (7) changes linearly with it. Therefore any point located at the surface of the hyper-sphere will remain to the surface of the new boundary while all the axes, except the one defined by the $\mathbf{e}_{//}$ unit vector, rescale by the factor α_{def}. This effectively corresponds to the formation of a hyper-spheroid boundary with symmetry axis at the $\mathbf{e}_{//}$ direction. Similarly to the truncation, the deformation transformation further modifies the initially uniform probability distribution. Fig. 12 presents a deformation of the unit hyper-sphere and the resulting mapping of the $\mathbf{x}_{gen}$ point. Fig. 13 shows the effect of increasing the α_{def} values on the hyper-sphere

deformation.

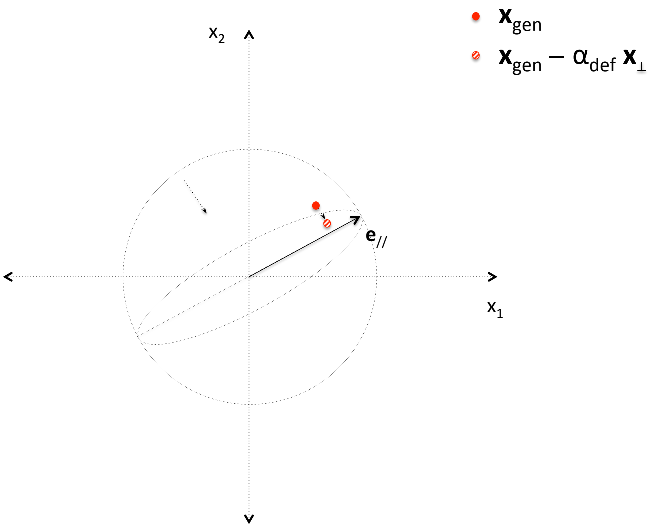

Figure 12: The transformation `Deform` is applied and the generated point is mapped to a new point towards the diameter of the hyper-sphere.

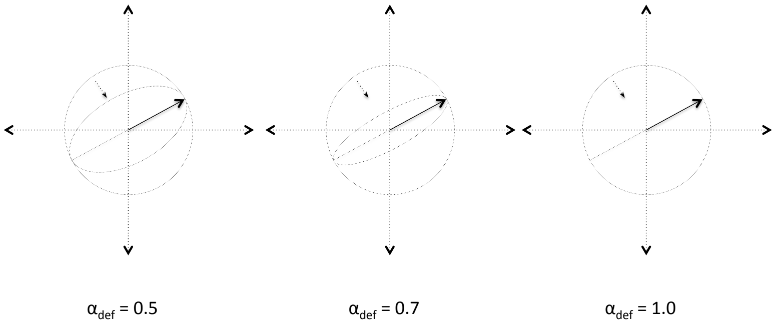

Figure 13: The effect of increasing α_{def} on the hypersphere deformation. The last case corresponds to a line segment.

9. The final step of the algorithm is the translation of the generated point by the $\mathbf{x}_{center}$ vector and the rescaling by the value of the radius R. The combined result of this two transformations is described in equation (8). Fig. 14 and fig. 15 show the resulting boundaries of the permissible data generation area as well as a random generated point $\mathbf{x}_{center}$, of the two different scenarios presented in fig. 7 and fig. 8, after the application of truncation, deformation and translation.

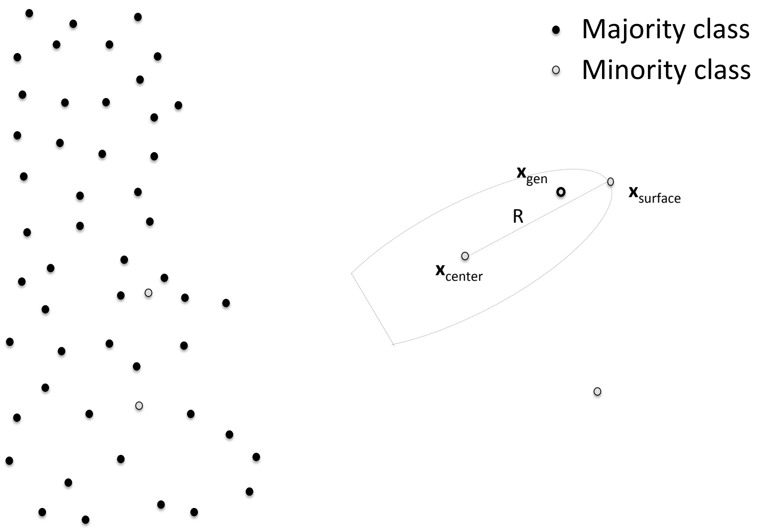

Figure 14: Boundaries of permissible data generation area for the scenario of Fig 7.

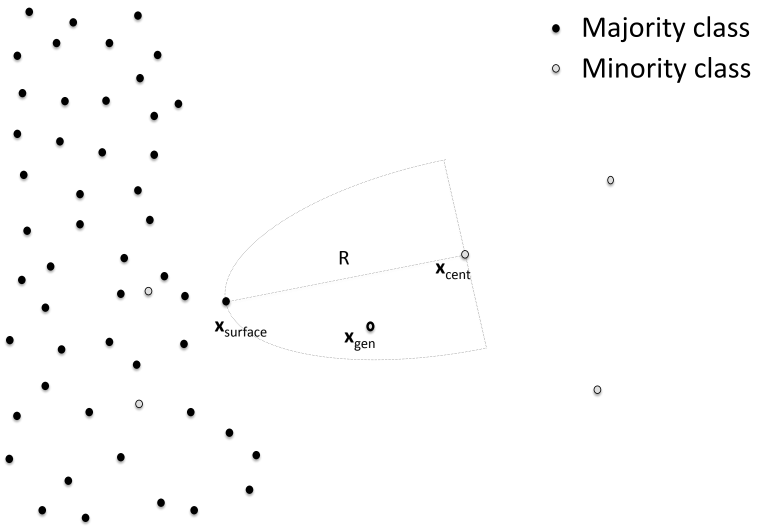

Figure 15: Boundaries of permissible data generation area for the scenario of Fig 8.

5 Research methodology

This section describes the evaluation process of G-SMOTE using a variety of classifiers, datasets and metrics. A description of the experimental data, the evaluation measures, the classification algorithms, the experimental procedure as well as the software implementation is provided.

5.1 Experimental data

In order to test the performance of G-SMOTE we used the following imbalanced datasets:

- UCI Machine Learning Repository, 13 datasets.
- KEEL repository, 13 datasets.
- Simulated data based on variations of the ”MANDELON” dataset [Guyon, 2003], 2 datasets.

Furthermore, additional datasets with even higher imbalance ratios were generated, by randomly undersampling the minority classes of the aforementioned datasets. For each one of them, its initial IR was multiplied by a factor of 2 and 3 but the resulting dataset was used for the experiments only if the total number of minority samples was not less than 15. Table 1 shows a summary of the final 69 data sets that can be download from `https://github.com/IMS-ML-Lab/publications/raw/master/assets/data/imbalanced_data.db`. The multiplication factor is appended in parentheses to the dataset's names while the table is sorted by ascending IR:

Dataset name	Features	Instances	Minority instances	Majority instances	Imbalance Ratio
HEART	13	270	120	150	1.25
LIVER	6	345	145	200	1.38
WINE	13	178	71	107	1.51
PIMA	8	769	268	501	1.87
BREAST TISSUE	9	106	36	70	1.94
IRIS	4	150	50	100	2.00
GLASS	9	214	70	144	2.06
YEAST 1	8	1,484	429	1,055	2.46
HEART (2)	13	210	60	150	2.50
LIVER (2)	6	272	72	200	2.78
HABERMAN	3	306	81	225	2.78
WINE (2)	13	142	35	107	3.06
VEHICLE	18	846	199	647	3.25
PIMA (2)	8	635	134	501	3.74
HEART (3)	13	190	40	150	3.75
BREAST TISSUE (2)	9	88	18	70	3.89
IRIS (2)	4	125	25	100	4.00
GLASS (2)	9	179	35	144	4.11
LIVER (3)	6	248	48	200	4.17
WINE (3)	13	130	23	107	4.65
YEAST 1 (2)	8	1,269	214	1,055	4.93
NEW THYROID 2	5	215	35	180	5.14
NEW THYROID 1	5	215	35	180	5.14
ECOLI	7	336	52	284	5.46
EUCALYPTUS	8	642	98	544	5.55
HABERMAN (2)	3	265	40	225	5.62
PIMA (3)	8	590	89	501	5.63
SEGMENTATION	16	2,310	330	1,980	6.00
IRIS (3)	4	116	16	100	6.25
GLASS (3)	9	167	23	144	6.26
VEHICLE (2)	18	746	99	647	6.54
YEAST 1 (3)	8	1,198	143	1,055	7.38
YEAST 3	8	1,484	163	1,321	8.10

Dataset name	Features	Instances	Minority instances	Majority instances	Imbalance Ratio
HABERMAN (3)	3	252	27	225	8.33
PAGE BLOCKS 0	10	5,472	559	4,913	8.79
VEHICLE (3)	18	713	66	647	9.80
VOWEL	13	988	90	898	9.98
NEW THYROID 1 (2)	5	197	17	180	10.59
NEW THYROID 2 (2)	5	197	17	180	10.59
ECOLI (2)	7	310	26	284	10.92
LED	7	443	37	406	10.97
EUCALYPTUS (2)	8	593	49	544	11.10
SEGMENTATION (2)	16	2,145	165	1,980	12.00
LIBRAS	90	360	24	336	14.00
PAGE BLOCKS 1 3	10	472	28	444	15.86
YEAST 3 (2)	8	1,402	81	1,321	16.31
ECOLI (3)	7	301	17	284	16.71
DERMATOLOGY	34	358	20	338	16.90
EUCALYPTUS (3)	8	576	32	544	17.00
PAGE BLOCKS 0 (2)	10	5,192	279	4,913	17.61
SEGMENTATION (3)	16	2,090	110	1,980	18.00
VOWEL (2)	13	943	45	898	19.96
LED (2)	7	424	18	406	22.56
YEAST 3 (3)	8	1,375	54	1,321	24.46
PAGE BLOCKS 0 (3)	10	5,099	186	4,913	26.41
MANDELON 1	20	4,000	142	3,858	27.17
MANDELON 2	200	3,000	105	2,895	27.57
YEAST 4	8	1,484	51	1,433	28.10
VOWEL (3)	13	928	30	898	29.93
YEAST 5	8	1,484	44	1,440	32.73
YEAST 6	8	1,484	35	1,449	41.40
MANDELON 1 (2)	20	3,929	71	3,858	54.34
MANDELON 2 (2)	200	2,947	52	2,895	55.67
YEAST 4 (2)	8	1,458	25	1,433	57.32
YEAST 5 (2)	8	1,462	22	1,440	65.45
MANDELON 1 (3)	20	3,905	47	3,858	82.09
MANDELON 2 (3)	200	2,930	35	2,895	82.71
YEAST 4 (3)	8	1,450	17	1,433	84.29
YEAST 6 (2)	8	1,466	17	1,449	85.24

Table 1: Description of the datasets.

5.2 Evaluation measures

Various assessment metrics can be used to evaluate a classifier's performance. However, not all of them are suitable for imbalanced datasets [He and Garcia, 2009]. The most common metric for binary classification problems is accuracy defined as the ratio of correct predictions for both of the classes over the total number of predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP/TN and FP/FN are the true positives/negatives and false positives/negatives, respectively. When the class distribution is imbalanced, assuming that the minority class is identified as the P label, the contribution of TP and FP to the above formula might be negligible, i.e. accuracy is mainly determined by the majority class contribution. Based on this, the selected evaluation metrics of the experimental procedure are following:

– Area Under the ROC Curve (AUC):

  ROC curve results from varying the decision threshold and plotting the true positive rate against the false positive rate.

– F-score:

  It is defined as the harmonic mean of $Precision$ and $Recall$, where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$.

– G-mean:

  It is defined as the geometric mean of $Sensitivity$ and $Specificity$, where $Sensitivity = \frac{TP}{TP+FN}$ and $Specificity = \frac{TN}{TN+FP}$.

5.3 Machine learning algorithms

The main objective of the paper is to compare G-SMOTE and SMOTE algorithms. Therefore their performance was evaluated on the aforementioned 69 datasets. Additionally, the use of unmodified data (NO OVERSAMPLING) and random oversampling (RANDOM OVERSAMPLING) were included as a baseline methods.

For the evaluation of the oversampling methods the following 4 classifiers were used: Logistic Regression (LR) [McCullagh and Nelder, 1989], K-Nearest Neighbors (KNN) [Cover and Hart, 1967], Decision Tree (DT) [Salzberg, 1994] and Gradient Boosting (GBC) [Friedman, 2001].

The implementation of the classifiers and standard oversampling algorithms was based on the Python libraries Scikit-Learn [Pedregosa et al., 2011] and Imbalanced-Learn [Lemaitre et al., 2016].

5.4 Experimental procedure

In order to evaluate the performance of each combination of oversampler and classifier, n-fold cross validation was applied with $n = 5$. Let D one of the datasets. Before starting the training of the classifier, in each stage $i \in \{1, 2, ..., n\}$ of the n-fold cross validation procedure, synthetic data $T_{g,i}$ were generated from the oversampler based on the training data T_i of the $n - 1$ folds such that the resulting $T_{g,i} \cup T_i$ training set becomes perfectly balanced. This enhanced training set in turn was used to train the classifier. The performance evaluation of the classifiers was done on the validation data V_i of the remaining fold, where $V_i \cup T_i = D$ and $V_i \cap T_i = \emptyset$.

A variety of hyper-parameters were used for the the oversamplers and classifiers. For SMOTE the optimal value of k nearest neighbors was selected as $k \in \{3, 5\}$ while for G-SMOTE a hyper-parameter grid was generated from the Cartesian product of the three different selection strategies, the number of nearest neighbors $k \in \{3, 5\}$, the truncation factor $\alpha_{trunc} \in \{-1.0, -0.5, 0.0, 0.25, 0.5, 0.75, 1.0\}$ and the deformation factor $\alpha_{def} \in \{0.0, 0.2, 0.4, 0.5, 0.6, 0.8, 1.0\}$. As far the classifiers are concerned, the number of nearest neighbors for KNN was selected as $k \in \{3, 5\}$, the max depth for DT was selected as *max depth* $\in \{3, 6\}$ while GBC hyper-parameter grid included the four combinations resulting from the Cartesian product of *max depth* $\in \{3, 6\}$ and *number of estimators* $\in \{50, 100\}$.

Using the above hyper-parameters, the highest cross validation score for each combination of datasets, classifiers, oversamplers and evaluation metrics was reported. The experimental procedure was repeated 3 times and the reported results include the average values between the experiments. A ranking score was assigned to each oversampling method with the best and worst performing methods receiving scores equal to 1 and 4, respectively. We apply the Friedman test to confirm the statistical significance of the ranking results of the experiments [Guyon, 2003]. The Friedman test is used to detect differences for a set of experimental attempts when normality assumption may not hold. In this case the null hypothesis represents the situation in which the classifiers show an identical performance, in terms of their mean ranking, independently of the oversampling method and performance metric used. Additionally, we apply the Holm's test, using G-SMOTE as the control method [Guyon, 2003]. The Holm's test is a non-parametric version of the t-test, where the null hypothesis is whether the proposed G-SMOTE algorithm outperforms the other methods as the control method.

5.5 Software implementation

As it was mentioned above, the implementation of the experimental procedure based on the Python programming language and Scikit-Learn/Imbalanced-Learn libraries. Specifically, the function that prepares and runs any comparative experiment can be found at `https://github.com/georgedouzas/scikit-learn-extensions/blob/master/sklearnext/tools/imbalanced_analysis.py`, while the G-SMOTE implementation is available at `https://github.com/georgedouzas/scikit-learn-extensions/blob/master/sklearnext/over_sampling/geometric_smote.py`. Being fully integrated with the Scikit-Learn ecosystem they can be adjusted and used for a variety of custom comparative studies. The experiments reported in this paper as well as the analysis of their results are reproducible using the scripts provided at `https://github.com/IMS-ML-Lab/publications`.

6 Results and discussion

In this section the performance of the different oversamplers is presented. The results are analyzed, by applying various statistical tests, to establish their significance. Additionally, a taxonomy of G-SMOTE hyper-parameters is presented as well as guidelines for their tuning relative to the characteristics of the imbalanced datasets.

6.1 Comparative presentation

The mean and standard error of cross validation scores across datasets for each combination of classifiers, metrics and oversamplers are presented in Table 2:

Classifier	Metric	NO OVERSAMPLING	RANDOM OVERSAMPLING	SMOTE	G-SMOTE
LR	AUC	0.90 ± 0.01	0.90 ± 0.01	0.90 ± 0.01	**0.91 ± 0.01**
LR	F-SCORE	0.44 ± 0.05	0.59 ± 0.03	0.59 ± 0.03	**0.60 ± 0.03**
LR	G-MEAN	0.50 ± 0.05	0.84 ± 0.02	0.84 ± 0.02	**0.86 ± 0.01**
KNN	AUC	0.82 ± 0.02	0.82 ± 0.02	0.84 ± 0.02	**0.86 ± 0.01**
KNN	F-SCORE	0.55 ± 0.03	0.58 ± 0.03	0.59 ± 0.03	**0.62 ± 0.03**
KNN	G-MEAN	0.63 ± 0.03	0.76 ± 0.03	0.78 ± 0.03	**0.82 ± 0.02**

Classifier	Metric	NO OVERSAMPLING	RANDOM OVERSAMPLING	SMOTE	G-SMOTE
DT	AUC	0.84 ± 0.01	0.85 ± 0.01	0.86 ± 0.01	**0.89 ± 0.01**
DT	F-SCORE	0.62 ± 0.03	0.63 ± 0.03	0.64 ± 0.03	**0.69 ± 0.02**
DT	G-MEAN	0.73 ± 0.02	0.82 ± 0.01	0.83 ± 0.01	**0.87 ± 0.01**
GBC	AUC	0.91 ± 0.01	0.91 ± 0.01	0.91 ± 0.01	**0.93 ± 0.01**
GBC	F-SCORE	0.66 ± 0.03	0.68 ± 0.03	0.70 ± 0.03	**0.74 ± 0.02**
GBC	G-MEAN	0.75 ± 0.02	0.82 ± 0.02	0.83 ± 0.01	**0.87 ± 0.01**

Table 2: Results for mean cross validation scores of oversamplers across the datasets.

Table 2 shows that G-SMOTE systematically performs better on average than the rest of the methods. Particularly, the mean and standard error of percentage difference between G-SMOTE and SMOTE is presented in Table 3:

Classifier	Metric	Difference
LR	AUC	0.41 ± 0.07
LR	F-SCORE	1.80 ± 0.43
LR	G-MEAN	2.15 ± 0.42
KNN	AUC	3.24 ± 0.92
KNN	F-SCORE	13.00 ± 2.92
KNN	G-MEAN	149.69 ± 110.16
DT	AUC	3.51 ± 0.49
DT	F-SCORE	12.60 ± 2.62
DT	G-MEAN	4.25 ± 0.63
GBC	AUC	1.65 ± 0.24
GBC	F-SCORE	7.68 ± 1.35
GBC	G-MEAN	4.44 ± 0.54

Table 3: Results for percentage difference between G-SMOTE and SMOTE across the datasets.

As explained in section 5.4 a ranking score in the range 1 to 4 is assigned to each oversampler. Then the mean and standard error of oversampling methods' ranking across the datasets for each combination of a classifier and evaluation metric is presented in Table 4:

Classifier	Metric	NO OVERSAMPLING	RANDOM OVERSAMPLING	SMOTE	G-SMOTE
LR	AUC	2.91 ± 0.14	2.94 ± 0.10	2.74 ± 0.08	**1.41 ± 0.07**
LR	F-SCORE	3.23 ± 0.14	2.88 ± 0.09	2.42 ± 0.10	**1.47 ± 0.08**
LR	G-MEAN	3.74 ± 0.08	2.47 ± 0.07	2.58 ± 0.08	**1.21 ± 0.06**
KNN	AUC	2.98 ± 0.10	3.39 ± 0.08	2.38 ± 0.09	**1.25 ± 0.06**
KNN	F-SCORE	3.16 ± 0.15	2.73 ± 0.09	2.70 ± 0.10	**1.41 ± 0.07**
KNN	G-MEAN	3.70 ± 0.09	2.84 ± 0.07	2.17 ± 0.08	**1.29 ± 0.06**
DT	AUC	3.24 ± 0.10	3.16 ± 0.08	2.43 ± 0.10	**1.17 ± 0.05**
DT	F-SCORE	2.80 ± 0.14	3.16 ± 0.08	2.66 ± 0.11	**1.38 ± 0.07**
DT	G-MEAN	3.65 ± 0.08	2.81 ± 0.08	2.37 ± 0.08	**1.17 ± 0.05**
GBC	AUC	2.99 ± 0.11	3.23 ± 0.09	2.62 ± 0.09	**1.15 ± 0.05**
GBC	F-SCORE	3.20 ± 0.11	3.08 ± 0.08	2.53 ± 0.10	**1.20 ± 0.05**
GBC	G-MEAN	3.64 ± 0.08	2.92 ± 0.08	2.31 ± 0.06	**1.12 ± 0.05**

Table 4: Results for mean ranking of oversamplers across the datasets.

6.2 Statistical analysis

The results of the application of the Friedman test are shown in Table 5:

Classifier	Metric	p-value	Significance
LR	AUC	$2.3 \cdot 10^{-16}$	True
LR	F-SCORE	$1.4 \cdot 10^{-16}$	True
LR	G-MEAN	$1.7 \cdot 10^{-30}$	True
KNN	AUC	$9.3 \cdot 10^{-25}$	True
KNN	F-SCORE	$4.1 \cdot 10^{-16}$	True
KNN	G-MEAN	$2.1 \cdot 10^{-29}$	True
DT	AUC	$1.2 \cdot 10^{-25}$	True
DT	F-SCORE	$8.3 \cdot 10^{-17}$	True
DT	G-MEAN	$3.8 \cdot 10^{-30}$	True
GBC	AUC	$1.6 \cdot 10^{-24}$	True
GBC	F-SCORE	$5.6 \cdot 10^{-24}$	True
GBC	G-MEAN	$1.7 \cdot 10^{-32}$	True

Table 5: Results for Friedman test.

Therefore at a significance level of a = 0.05 the null hypothesis is rejected, i.e. the classifiers do not perform similarly in the mean rankings across the oversampling methods and evaluation metrics. Following the Friedman test, the Holm's method is applied to adjust the p-values of the paired difference test with G-SMOTE algorithm as the control method. The adjusted p-values are presented in Table 6:

Classifier	Metric	NO OVERSAMPLING	RANDOM OVERSAMPLING	SMOTE
LR	AUC	$9.8 \cdot 10^{-4}$	$2.8 \cdot 10^{-8}$	$1.5 \cdot 10^{-7}$
LR	F-SCORE	$1.2 \cdot 10^{-9}$	$2.8 \cdot 10^{-9}$	$9.2 \cdot 10^{-5}$
LR	G-MEAN	$1.5 \cdot 10^{-12}$	$4.0 \cdot 10^{-8}$	$1.4 \cdot 10^{-8}$
KNN	AUC	$2.5 \cdot 10^{-9}$	$2.1 \cdot 10^{-10}$	$5.3 \cdot 10^{-5}$
KNN	F-SCORE	$1.1 \cdot 10^{-7}$	$1.0 \cdot 10^{-9}$	$2.0 \cdot 10^{-10}$
KNN	G-MEAN	$2.7 \cdot 10^{-11}$	$8.0 \cdot 10^{-5}$	$1.6 \cdot 10^{-2}$
DT	AUC	$1.8 \cdot 10^{-12}$	$1.1 \cdot 10^{-13}$	$1.3 \cdot 10^{-10}$
DT	F-SCORE	$1.6 \cdot 10^{-8}$	$4.2 \cdot 10^{-15}$	$1.8 \cdot 10^{-9}$
DT	G-MEAN	$4.9 \cdot 10^{-12}$	$4.9 \cdot 10^{-12}$	$4.6 \cdot 10^{-10}$
GBC	AUC	$5.7 \cdot 10^{-9}$	$5.7 \cdot 10^{-9}$	$1.6 \cdot 10^{-9}$
GBC	F-SCORE	$3.9 \cdot 10^{-12}$	$4.1 \cdot 10^{-15}$	$1.7 \cdot 10^{-12}$
GBC	G-MEAN	$8.3 \cdot 10^{-12}$	$1.0 \cdot 10^{-12}$	$1.7 \cdot 10^{-12}$

Table 6: Adjusted p-values using the Holm's method.

Therefore, the null hypothesis of the Holm's test is rejected for all oversamplers at a significance level of a = 0.05, indicating that the proposed method outperforms all other methods.

6.3 G-SMOTE taxonomy

The G-SMOTE hyper-parameters α_{trunc}, α_{def} and α_{sel}, called geometric hyper-parameters, allow to create a taxonomy of different hyper-parameters' configurations with a simple geometric interpretation.

The various categories are tested during the experimental procedure and compared to the rest of the oversampling methods. A short description is provided below:

- SMOTE algorithm.

  The choice of hyper-parameters that reproduces the SMOTE algorithm is $\alpha_{def} = 1.0$, $\alpha_{trunc} = 1.0$ and $\alpha_{sel} = minority$. The first of them ensures that the initial hypersphere is deformed in to a line segment, the second truncates the half of the resulting line segment and the third chooses the SMOTE selection strategy.

- Modified SMOTE algorithm.

  The permissible data generation area remains a line segment as the above case. Therefore $\alpha_{def} = 1.0$ but with no further restriction on the α_{trunc} values. Also instead of the *minority* selection strategy, it uses either the *majority* or *combined* one. This case can be considered as a modification of the SMOTE mechanism where the line segment is either truncated, expanded or rotated.

- Pure G-SMOTE algorithm.

  This includes any hyper-parameters' configuration where the permissible data generation area is expanded from a line segment of the above cases into a hyper-spheroid. The choice of hyper-parameters in this case is $\alpha_{def} < 1.0$ with no further restriction on the rest of the geometrical hyper-parameters.

Table 7 shows the frequency of the above 3 cases as an optimal configuration across the total 26391 experiments:

Case	Frequency
Pure G-SMOTE	23,743
Modified SMOTE	2,392
SMOTE	256
Total experiments	26,391

Table 7: Optimal configuration frequency for the three G-SMOTE cases.

Therefore pure G-SMOTE represents the configuration with the highest frequency and ranking among the 3 cases, covering the majority of optimal hyper-parameter configurations. Additionally, SMOTE represents a small fraction of the optimal cases while its modification improves significantly the original data generation mechanism.

6.4 Analysis and tuning of optimal hyper-parameters

As it was explained on the above section, the G-SMOTE geometric hyper-parameters control the characteristics of the data generation process. More specifically, the *majority* selection strategy expands aggressively the minority class area, increasing the variety of generated samples. Lower absolute values of the truncation and deformation factors have a similar effect. On the other hand, the *minority* selection strategy and higher absolute values of the truncation and deformation factors result in a data generation process that resembles the SMOTE mechanism.

An analysis of the G-SMOTE geometric hyper-parameters dependence on Imbalance Ratio (IR) and number of samples over number of features ratio (R) shows the following:

- High IR or low R.

  The *majority* (or *combined*) selection strategy and lower absolute values of the truncation and deformation factors dominate as optimal hyper-parameters.

- Low IR or high R.

  The *minority* selection strategy and higher absolute values of the truncation and deformation factors dominate as optimal hyper-parameters.

Therefore datasets with higher IR or lower R values favor the generation of diverse synthetic samples. The explanation for this finding is that for high IR values the SMOTE data generation process creates either nearly duplicate or noisy examples, while in sparse input spaces, where R is low, the linear interpolation mechanism of SMOTE generates samples on a special direction of the input space that effectively makes them nearly duplicate or noisy. Contrary to the above, for lower IR or higher R values, the input space contains enough information to avoid the above shortcomings of the SMOTE mechanism.

7 Conclusions

In this paper we presented G-SMOTE, a new oversampling algorithm, that extends the SMOTE data generation mechanism. G-SMOTE selects a safe radius around each minority class instance and generates artificial data within a (truncated) hyper-spheroid. G-SMOTE performance was evaluated on 69 datasets with different imbalance ratios and compared to no oversampling, Random Oversampling and SMOTE, using Logistic Regression, K-Nearest Neighbors, Decision Tree and Gradient Boosting Machine as classifiers.

The results show that G-SMOTE performs significantly better compared to the other methods. The explanation for this improvement in performance relates to the ability of G-SMOTE to generate artificial data in safe areas of the input space, while, at the same time, aggressively increasing the diversity of the generated instances. G-SMOTE parametrizes efficiently the data generation process and adapts to the special characteristics of each imbalanced dataset. We make available an implementation of G-SMOTE at `https://github.com/georgedouzas/scikit-learn-extensions/blob/master/sklearnext/over_sampling/geometric_smote.py`.

G-SMOTE can be a useful tool for researchers and practitioners since it results in the generation of high quality artificial data and only requires the tuning of a small number of parameters.

References

[Alcalá-Fdez et al., 2011] Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. (2011). Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17.

[Barua et al., 2014] Barua, S., Islam, M. M., Yao, X., and Murase, K. (2014). MWMOTE - Majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425.

[Batista et al., 2004] Batista, G. E. A. P. A., Prati, R. C., and Monard, M. C. (2004). A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets*, 6(1):20–29.

[Blagus and Lusa, 2013] Blagus, R. and Lusa, L. (2013). SMOTE for high-dimensional class-imbalanced data. *BMC Bioinformatics*, 14(1):106.

[Bunkhumpornpat et al., 2009] Bunkhumpornpat, C., Sinapiromsaran, K., and Lursinsap, C. (2009). Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5476 LNAI, pages 475–482.

[Bunkhumpornpat et al., 2012] Bunkhumpornpat, C., Sinapiromsaran, K., and Lursinsap, C. (2012). DBSMOTE: Density-based synthetic minority over-sampling technique. *Applied Intelligence*, 36(3):664–684.

[Chawla, 2005] Chawla, N. V. (2005). Data Mining for Imbalanced Datasets: An Overview. In *Data Mining and Knowledge Discovery Handbook*, pages 853–867. Springer-Verlag.

[Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

[Chawla et al., 2003] Chawla, N. V., Lazarevic, A., Hall, L., and Boyer, K. (2003). SMOTEBoost: improving prediction of the minority class in boosting. *Principles of Knowledge Discovery in Databases, PKDD-2003*, pages 107–119.

[Cieslak et al., 2006] Cieslak, D. A., Chawla, N. V., and Striegel, A. (2006). Combating imbalance in network intrusion datasets. In *2006 IEEE International Conference on Granular Computing*, pages 732–737.

[Cover and Hart, 1967] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

[DasGupta, 2011] DasGupta, A. (2011). *Probability for Statistics and Machine Learning.* Springer New York.

[Domingos, 1999] Domingos, P. (1999). MetaCost: A General Method for Making Classifiers Cost-Sensitive. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 55:155–164.

[Douzas and Bacao, 2017a] Douzas, G. and Bacao, F. (2017a). Effective data generation for imbalanced learning using Conditional Generative Adversarial Networks. *Expert Systems with Applications.*

[Douzas and Bacao, 2017b] Douzas, G. and Bacao, F. (2017b). Self-organizing map oversampling (somo) for imbalanced data set learning. *Expert Systems with Applications*, 82:40 – 52.

[Douzas et al., 2018] Douzas, G., Bacao, F., and Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465:1 – 20.

[Fernandez et al., 2018] Fernandez, A., Garcia, S., Herrera, F., and V. Chawla, N. (2018). Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. 61:863–905.

[Fernández et al., 2013] Fernández, A., López, V., Galar, M., del Jesus, M. J., and Herrera, F. (2013). Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, 42:97–110.

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

[Galar et al., 2012] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2012). A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches.

[Guo and Viktor, 2004] Guo, H. and Viktor, H. L. (2004). Learning from imbalanced data sets with boosting and data Generation : The DataBoost-IM approach. *ACM SIGKD Explorations Newsletter - Special issue on learning from imbalanced datasets*, 6(1):30–39.

[Guyon, 2003] Guyon, I. (2003). Design of experiments of the NIPS 2003 variable selection benchmark. In *NIPS 2003 workshop on feature extraction*.

[Haixiang et al., 2017] Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., and Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239.

[Han et al., 2005] Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *Lecture Notes in Computer Science*, pages 878–887. Springer Berlin Heidelberg.

[He and Garcia, 2009] He, H. and Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

[Jo and Japkowicz, 2004] Jo, T. and Japkowicz, N. (2004). Class imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsletter*, 6(1):40.

[Lemaitre et al., 2016] Lemaitre, G., Nogueira, F., and Aridas, C. K. (2016). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, 18:1–5.

[Lichman, 2013] Lichman, M. (2013). Uci machine learning repository.

[McCullagh and Nelder, 1989] McCullagh, P. and Nelder, J. (1989). *Generalized Linear Models*.

[Nekooeimehr and Lai-Yuen, 2016] Nekooeimehr, I. and Lai-Yuen, S. K. (2016). Adaptive semi-unsupervised weighted oversampling (A-SUWO) for imbalanced datasets. *Expert Systems with Applications*, 46:405–416.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). *Scikit-learn: Machine learning in Python*, volume 12.

[Salzberg, 1994] Salzberg, S. L. (1994). C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16(3):235–240.

[Sun et al., 2015] Sun, Z., Song, Q., Zhu, X., Sun, H., Xu, B., and Zhou, Y. (2015). A novel ensemble method for classifying imbalanced data. *Pattern Recognition*, 48(5):1623–1637.

[Tang and He, 2015] Tang, B. and He, H. (2015). KernelADASYN: Kernel based adaptive synthetic data generation for imbalanced learning. In *2015 IEEE Congress on Evolutionary Computation*. IEEE.

[Ting, 2002] Ting, K. M. (2002). An Instance-Weighting Method to Induce Cost-Sensitive Trees. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):659–665.

[Vaidya, 1989] Vaidya, P. M. (1989). An O(n logn) algorithm for the all-nearest-neighbors Problem. *Discrete & Computational Geometry*, 4(2):101–115.

[Wang et al., 2015] Wang, S., Minku, L. L., and Yao, X. (2015). Resampling-Based Ensemble Methods for Online Class Imbalance Learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1356–1368.

[Wilson, 1972] Wilson, D. L. (1972). Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man and Cybernetics*, 2(3):408–421.