DTO-SMOTE: Delaunay Tessellation Oversampling for Imbalanced Data Sets

Alexandre M. de Carvalho † and Ronaldo C. Prati *,†
Center of Mathematics, Computing and Cognition (CMCC), Federal University of ABC (UFABC), Santo André, São Paulo (09210-580), Brazil; carvalho.alexandre@ufabc.edu.br
* Correspondence: ronaldo.prati@ufabc.edu.br; Tel.: +55-11-4996-8309
† These authors contributed equally to this work.
Received: 5 November 2020; Accepted: 25 November 2020; Published: 28 November 2020

Abstract: One of the significant challenges in machine learning is the classification of imbalanced data. In many situations, standard classifiers cannot learn how to distinguish minority class examples from the others. Since many real problems are unbalanced, this problem has become very relevant and deeply studied today. This paper presents a new preprocessing method based on Delaunay tessellation and the preprocessing algorithm SMOTE (Synthetic Minority Over-sampling Technique), which we call DTO-SMOTE (Delaunay Tessellation Oversampling SMOTE). DTO-SMOTE constructs a mesh of simplices (in this paper, we use tetrahedrons) for creating synthetic examples. We compare results with five preprocessing algorithms (GEOMETRIC-SMOTE, SVM-SMOTE, SMOTE-BORDERLINE-1, SMOTE-BORDERLINE-2, and SMOTE), eight classification algorithms, and 61 binary-class data sets. For some classifiers, DTO-SMOTE has higher performance than others in terms of Area Under the ROC curve (AUC), Geometric Mean (GEO), and Generalized Index of Balanced Accuracy (IBA).

Keywords: machine learning; SMOTE; oversampling; DTO-SMOTE; imbalanced data

1. Introduction

Imbalanced data sets [1] are among the pattern classification problems. To illustrate some applications in recent years, methods for dealing with class imbalance were applied in different areas, such as: Medicine [2], Agriculture [3], Computer Networks Protection [4], analysis of social media [5], and Financial Risks [6,7]. Although researchers and professionals have intensely studied this problem, today, this challenge is still open-ended. Most of the real-world data sets are naturally imbalanced [8], and classifiers have difficulties learning from them. Different techniques to solve or mitigate this problem emerge all the time [9–12]. Nevertheless, this problem is far from being addressed entirely.

Synthetic Minority Oversampling Technique (SMOTE) [13] is one of the most used preprocessing methods. It works by synthetically generating instances in the line segment joining two examples. SMOTE has several variations in the literature. However, they also follow the same procedure for synthetic instance generation. This procedure leads to a bias of SMOTE (and its variants) to generate a chain trail pattern. This preference is discussed further in Section 3.2. As far as we know, this bias was unnoticed in the literature. We argue that this undesirable tendency may limit the effective use of SMOTE in some domains.

To overcome this problem, we propose a new preprocessing method that relies on constructing a proximity graph based on Delaunay Tessellation. The Delaunay Tessellation (also known as triangulation in the plane) is a fundamental computational geometry structure. The Voronoi diagram’s dual and provides a connection graph for closer points in the space, forming a tessellation composed
of a simplex set that completely covers the space. The Delaunay graph is interesting in data analysis, as it can represent the geometry of a point set or approximate its density [14]. We can use the fact that any point inside the simplex can be described as a linear combination of its vertices to generate new synthetic samples.

We name this new method Delaunay Tessellation Oversampling SMOTE (DTO-SMOTE). Like SMOTE [13], the main idea of Delaunay-Oversampling is to synthetically generate new instances, aiming to balance the training data distribution before applying the learning algorithms. The main difference is that, instead of creating synthetic examples in the line segment joining two instances, we generate synthetic examples inside a simplex selected from the Delaunay tessellation. The vertices of the simplex are instances of the data set. The creation of synthetic instances follows a Dirichlet distribution, and we can use the distribution parameters to manipulate where instances can be generated according to the classes of the vertices. This approach makes it possible to avoid SMOTE’s chain trail pattern, allowing artificial instances to cover the input space better.

This paper substantially extends our previous research [15] by introducing a simplex selection step based on quality measures and data generation and widely extends the empirical evaluation, including other learning algorithms, baselines, and data sets. The main contributions of this paper are:

- We point out (for the first time) the chain trail pattern formed by the artificial instances that SMOTE generates.
- We propose a new preprocessing technique, named Delaunay Tessellation Oversampling SMOTE (DTO-SMOTE), which uses Delaunay Tessellation to build a simplex mesh. Then, we use simplex quality measures to select candidates (our previous study draws a simplex at random) for instance generation and use a Dirichlet distribution to control where synthetic instances creation inside a simplex (our former study uses the barycenter of the simplex).
- We conduct an extensive experimental evaluation with 61 bi-class data sets (our previous study only considers 15 binary data sets). This empirical comparison includes five preprocessing methods and ten learning algorithms (our former analysis only compares with SMOTE and uses kNN as the learning algorithm). It shows our approach’s appropriateness in many situations, with better average performance on binary data sets.

The remaining of this paper is organized as follows: Section 2 presents issues related to class imbalance and some methods for dealing with them and performance measures. Section 3 discusses the SMOTE family of preprocessing algorithms and their bias in generating synthetic examples in a chain trail pattern. Section 4 presents our new method, DTO-SMOTE, in detail. Section 5 presents the experiments and results. Section 6 discusses the achieved results and describes insights, comments, and future work directions.

2. The Imbalanced Data Set Classification Problem

Classification problems in which one of the classes have more instances than the others are called Imbalanced Classification problems. In practice, although a strict imbalance ratio threshold does not exist, when there is a significant difference in the number of cases in each class, it is considered a matter of a Class Imbalanced Problem. Several authors in the literature acknowledge that this imbalance may degrade classification performance [16]. Learning from imbalanced data is an important issue as these data sets frequently occur in real-life problems, where practical problems are rarely balanced. This problem occurs in areas like Engineering, Information Technology, Bioinformatics, Medicine, Security, Business Management, among others [16].

Facing these problems, researchers have proposed some techniques to deal with imbalanced data set classification problems. There are three primary directives in this area: Preprocessing methods, Cost-Sensitive Learning, and Ensembles [16]. Section 2.2 briefly describes these three approaches, and Section 2.1 discusses the evaluation of classification problems. Based on these measures, we can assess if specific methods are working or not to solve imbalanced problems.
2.1. Performance Evaluation

Choosing adequate performance metrics to evaluate imbalanced data set classification problems is crucial to select the right preprocessing method and classifier’s algorithm [17]. In sequence, we review the most used measures for imbalanced classification problems [16,18]. Without loss of generality, here, we consider two-class classification problems to present these measures.

Figure 1 shows a Confusion Matrix in which the two classes are generally named Class 0 (Negative class or Majority class) and Class 1 (Positive class or Minority class). In this confusion matrix, \( C_{R0} \) corresponds to the actual number of majority class instances, and \( C_{R1} \) corresponds to the actual number of minority class instances. Furthermore, \( C_{P0} \) corresponds to the number of instances predicted as the majority class, and \( C_{P1} \) corresponds to the number of instances predicted as the minority class. True Positive values, or \( TP \), means that a classifier correctly classified majority class samples. False Positive, or \( FP \), means that examples of which correct label is the minority class misclassified as majority class. False Negative, or \( FN \), means that a real majority class instance is misclassified as belonging to the minority class. Finally, True Negative, or \( TN \), means that an actual minority class example is correctly classified as belonging to the minority class.

Based on this table, accuracy is defined as a rate between the total samples classified correctly by the total test samples submitted to the classifier. Equation (1) presents \( A_{cc} \) or Accuracy.

\[
A_{cc} = \frac{TP + TN}{TP + FN + TN + FP}.
\]

Figure 1. Confusion Matrix.

Considering that, in real-life applications, the positive class is the most important, \( A_{cc} \) is generally inadequate for imbalanced classifications problems. For instance, a trivial classifier could achieve 95% accuracy in a classification problem where the majority (negative) class instances account for 95%. Nevertheless, it could mispredict all positive class instances. To compromise with these issues, Sensitivity or Recall, (\( REC \)), and Specificity, (\( SPE \)), Equations (2) and (3), respectively, consider the correct classification in each class separately. Here, True Positive Rate, or \( TPrate \), is equal to Recall, or \( REC \), and True Negative rate, or \( TNrate \), is equal to \( SPE \), like in Equations (2) and (3), respectively.

Recall or True Positive Rate (\( TPrate \)—Equation (2)) is the percentage of positive instances correctly classified, whereas Specificity or True Negative Rate (\( TNrate \)—Equation (3)) is the percentage of negative cases correctly classified.

\[
TPrate = REC = \frac{TP}{TP + FN},
\]

\[
TNrate = SPE = \frac{TN}{TN + FP}.
\]
False Positive Rate ($FPrate$—Equation (4)) is the percentage of misclassified negative instances, and False Negative Rate ($FNrate$—Equation (5)) is the percentage of misclassified positive instances.

\[
FPrate = \frac{FP}{TN + FP}, \\
FNrate = \frac{FN}{TP + FN}.
\]

There are situations where the focal point is to achieve high assertiveness for positive class. In this case, precision ($PRE$—Equation (6)) is more suitable. Precision is the percentage of instances predicted as positive and correctly classified as positive.

\[
Precision = PRE = \frac{TP}{TP + FP}
\]

Precision and Recall are two conflicting factors, where improving one may imply in degrading the other. One measure that relates to Precision and Recall is called $F1$. Equation (7) shows the $F1$ score, which is the weighted harmonic mean between Precision and Recall.

\[
F1 = \frac{2 \cdot PRE \cdot REC}{PRE + REC}
\]

$GEO$ or Geometric Mean (Equation 8) is associated with the ROC curve [19]. Note that, when $REC$ and $SPE$ have high values, $GEO$ also has a high value. On the other hand, when one of them has a low value, $GEO$ diminishes.

\[
GEO = \sqrt{REC + SPE}
\]

AUC or Area Under Curve is related to the ROC (Receiver Operating Characteristic) curve [19]. This curve plots $TPR$ and $FPR$ when for various thresholds of the positive class likelihood. For a single threshold, $AUC$ could be calculated by Equation (9). For multiple thresholds, the trapezoidal rule [19].

\[
AUC = \frac{REC + SPE}{2}
\]

Generalized Index of Balanced Accuracy ($IBA$—Equation (10)), described in Reference [18], quantifies an arrangement between $Gmean$ and a ratio of how balanced the two-class precision in each class is in the data set [18]. The parameter $\alpha$ trades-off the influence dominance of $(TPrate - TNrate)$ and the $GMean$. The authors recommend that $\alpha \leq 0.5$, and, when $\alpha$ is set to 0, $IBA = Gmean^2$.

\[
IBA_\alpha = (1 + (\alpha \cdot (TPrate - TNrate))) \cdot Gmean^2
\]

In this present work, we adopted $\alpha = 0$, as recommended by the authors who proposed the metric [18].

### 2.2. Methods for Dealing with Class Imbalance

Dealing with imbalanced data is a challenging task. Real-world problems are imbalanced, and, to overcome this problem, several techniques have been proposed in the literature. In this section, we review some of them.

#### 2.2.1. Preprocessing

Preprocessing occurs before the data set is submitted to the classifier training process [16,20]. The main advantage of these methods is that they are algorithm agnostic, and it does not necessarily alter the original classifier algorithm. Therefore, they can be used together with any classifier. On the
other hand, preprocessing could be cost-intensive in terms of processing time. Main techniques for preprocessing imbalanced data sets are resampling algorithms [16]. Resampling works to achieve a balanced data set as a result. There are three resampling categories: Oversampling, Undersampling, and Hybrid methods.

**Oversampling** Oversampling creates synthetic or duplicate minority class samples [16,20] to match the same number of samples from the majority class. As a result, the training data set becomes balanced before the training phase. The Synthetic Minority Oversampling Technique [13,21], discussed in Section 3, is the primary method in this category.

**Undersampling** Undersampling discards some majority class instances to matches the number of samples from the minority class. The primary method in this category is Random Under-Sampling (RUS) [16,20,22]. Nevertheless, there are some issues regarding these techniques. When some data suffer deletion, important information could be discarded, leading to a weak classifier’s training due to the lack of relevant information. To deal with the lost of information, some strategies were proposed in the literature. In Evolutionary Undersampling [23], the undersampling is framed as a search problem for prototypes. This process reduces the number of instances from a data set, aiming not to lose a significant accuracy classification rate, using an evolutionary algorithm. Another interesting method is ACOSampling [24]. This method is based on ant colony optimization [25] in the search phase. It adopts this strategy to determine the best subset of majority class instances to keep in the training set.

**Hybrid** The hybrid method’s main idea is to minimize drawbacks from undersampling and oversampling, while taking into account their benefits, to achieve a balanced data set. To illustrate this combination, we can cite some methods, like SMOTE + Tomek Link [26], SMOTE + ENN [27], SMOTE-RSB [28], and SMOTE-IPF [29], which combine the SMOTE oversampling technique with different data cleaning, to remove some spurious artificial instances introduced in the oversampling phase, and data clustering followed by oversampling [30].

### 2.2.2. Cost-Sensitive Learning

The main idea about this strategy is that misclassification costs are uneven for the majority and minority classes. It is necessary to alter the original classifier’s algorithm to implement this strategy so that the algorithm considers the different misclassification costs. [31] present a recent review of cost-sensitive approaches.

### 2.2.3. Ensemble Learning

Ensemble methods combine two or more classifiers resulting in one classifier with (potentially) more performance than if they were used separately [32]. However, only using standard ensembles cannot solve imbalanced problems [16,33]. The reason is that ensembles aim mainly to improve accuracy, and, for imbalanced data classification, accuracy is not adequate due to the prevalence of the majority class. So, the most common technique to apply ensemble in imbalanced classification problems are variations of Bagging [34] and Boosting [33] especially developed for imbalanced data:

**Bagging**

Bagging [33] consists of training several classifiers with different sampled data sets, randomly drawn from the original one with replacement. The final classification uses a majority or weighted vote from the pool of classifiers [16,33]. One way to extend Bagging to imbalanced data is to train different classifiers with bootstrapped replicas of the original data set. To ensure that minority class instances are present in each sampled data set, stratified sampling is performed. Within each data set, oversampling methods are used to balance the classes [33].
Boosting

Boosting [33] differs from Bagging as Boosting weights samples to measure classification difficulty in the learning phase. At the beginning of the process, all instances have the same weight, and classifiers are trained iteratively, changing their weights. Difficult examples receive higher weights than simpler ones. To extend boosting to imbalanced scenarios, oversampling methods are introduced in the process to balance the original data set in each iteration [33].

3. The SMOTE Family of Oversampling

As our method was developed to relieve bias of the SMOTE oversampling method (and many variations that uses the procedure of SMOTE for generating a synthetic instance based on the interpolation within the line segment of two instances in the data set) suffers; in this section, we review the SMOTE family of oversampling methods.

The preprocessing technique SMOTE [13,21] is an oversampling technique in which the main goal is to artificially generate new instances of the minority class interpolating pairs of neighbor instances. The target is to achieve a balanced distribution among all the classes in the data set. To do that, SMOTE selects one minority class instance \( i \) and calculates its \( k \) minority class nearest-neighbors. The generation of a new synthetic instance consists of interpolating in a line segment formed by selected sample \( i \) and a random instance \( j \), which lies in the \( k \) minority nearest neighborhood of \( i \). In the original paper [13], the parameter \( k \) is set to 5. The generation of a new instance follows Equation (11).

\[
x_{\text{new}} = x_i + r(x_j - x_i), \quad 0 \leq r \leq 1, \quad (11)
\]

where:

- \( x_{\text{new}} \) is the new instance vector;
- \( x_i \) is the feature vector of instance \( i \);
- \( x_j \) is the feature vector of instance \( j \);
- \( r \) is a random number between 0 and 1.

In 2018, SMOTE was 15 years old [21], and an extensive overview of its applicability and variants based on it are surveyed. We briefly describe some of these methods next.

3.1. SMOTE Variations

3.1.1. Borderline-SMOTE

Borderline-SMOTE [35] is a variation that considers samples of the minority class that are far from the majority class boundary may contribute less than samples on the border to build the classifier. Thus, the Borderline-SMOTE preferentially generates synthetic instances that lie near to the decision border. Han describes in his paper [35] that data set examples on the borderline and nearby are more misclassified than cases in other regions. Here, borderline means a region where minority class examples are close to the majority ones.

3.1.2. SMOTE-SVM

SMOTE-SVM [36] generates artificial samples along the decision boundary. Like Borderline-Smote, SMOTE-SVM assumes that the decision boundary is the best place to create new synthetic samples, as this region is the most critical one for the training process. However, SMOTE-SVM uses support vectors to determine the decision boundaries in SMOTE-SVM.
3.1.3. ADASYN

The basic idea of ADASYN [37] is to use a density distribution of each class’s instances as a criterion to automatically choose the number of synthetic minority samples that need to be generated. Considering a binary class data set, and let $m_s$ and $m_l$ be the number of the minority class and the majority class samples. The degree of class imbalance $d$ is calculated by Equation (12).

$$d = \frac{m_s}{m_l}. \quad (12)$$

The number of minority class samples to be generated is calculated using Equation (13), $G$ is the number of artificial minority class instances to be created, and $\beta$ is a desired balanced coefficient parameter. When $\beta = 1$, the target is a fully balanced data set.

$$G = (m_l - m_s) \cdot \beta. \quad (13)$$

The ratio $r_i$ around each minority class instance in Equation (14) is defined by the distance of the $K$ nearest neighbor calculated in Euclidean distance, and $\Delta_i$ is the number of examples in $K$ nearest neighbors of $x_i$ which belongs to the majority class.

$$r_i = \frac{\Delta_i}{K}, \quad i = 1, ..., m_s. \quad (14)$$

Normalizing $r_i$ result in $\hat{r}_i$ calculated by Equation (15).

$$\hat{r}_i = \frac{r_i}{\sum_{i=1}^{m_s} r_i}. \quad (15)$$

After that, for each minority class instance $x_i$, the number of synthetic data samples that needs to be generated $g_i$ is calculated by Equation (16).

$$g_i = \hat{r}_i \times G. \quad (16)$$

Finally, for each $g_i$, generate new synthetic samples by Equation (17), where $s_i$ is a new synthetic sample, $x_{zi}$ is a randomly chosen minority class sample, and $\lambda$ is a random number between 0 and 1.

$$s_i = x_i + (x_{zi} - x_i) \times \lambda. \quad (17)$$

After all these steps, the output is a balanced data set.

3.1.4. Geometric SMOTE

Geometric SMOTE [38] generates synthetic samples in a geometric region called hyperspheroid. There are three main steps in this method. First, it creates new synthetic examples in the surface of a hypersphere centered at an original minority class example covering only minority class instances, ensuring that new examples are not noise. Second, it increases the variety of generated examples by expanding the respective hyperspheroid around the minority class instances applying hyper-sphere expansion factors. Finally, Geometric SMOTE has the options to use geometric transformations, such as translation, deformations, and truncation of the hyperspheroids, to cover other areas of the space.

3.1.5. Manifold-Based Synthetic Oversampling

Manifold-based synthetic oversampling [39] transforms the original data set into a manifold structure and generates instances in the manifold space. Finally, the algorithm maps the synthetic artificial samples to feature space. Aiming to improve SMOTE, the manifold oversampling method requires that the data set conform to the manifold property [39]. If this requirement is fulfilled, a manifold embedding method can be used to induce a manifold representation of the minority class.
Then, oversampling is applied in this embedded space. The most common manifold methods are Principal Component Analysis (PCA) and Denoising Autoencoders (DAE), according to Bellinger [39].

### 3.2. The SMOTE Chain Trail Pattern Bias

In general, SMOTE variants differ where the synthetic instances are generated. However, they are similar to how these synthetic instances are created. In this section, we argue that SMOTE and other methods based on its artificial instance generation process have an undesirable bias. The synthetic instances follow a chain trail pattern. The reason is related to the pairwise interpolation process, in which artificial examples are generated in the line segment joining two examples. For highly imbalanced data sets, the interpolation process has a high likelihood to select the same pair, as the selection procedure uses the neighborhood of the instances, and an instance can be selected as a seed several times. Therefore, the generation process creates synthetic examples within the same line segment, and a chain trail pattern emerges. Recently, Reference [40] noticed that the SMOTE has a bias where instances are inward-placed diverging from the original class distribution. Still, they did not correlate this with the synthetic data generation process. As far as we knew, this chain trail pattern is unnoticed in the literature.

Figure 2 illustrates this bias. This figure is a two-dimensional projection for a binary class (thyroid) data set(a), alongside their over-sampled versions using SMOTE (b), Borderline SMOTE with its two versions (c) and (d), SMOTE-SVM (e), and Geometric SMOTE (f), as well as our proposed DTO-SMOTE algorithm (g). Observe that the synthetic instances generated by SMOTE, Borderline SMOTE, and SMOTE-SVM follow a chain trail. For Borderline and SMOTE-SVM, these trails concentrate near class boundaries.

![Figure 2. SMOTE trail path patterns, thyroid-hypothyroid data set.](image)

On the other hand, although the chain trail pattern is not present in Geometric SMOTE, its synthetic data generation also leads to an undesirable bias. The artificial data generation process considers
hyper-spheres centered in some instances, where the radii may grow. Furthermore, the seed candidates are instances far from the decision boundary, and the bias pattern follows these hyper-spheres. Therefore, Geometric SMOTE also has a bias related to the synthetic data generation procedure.

As we will discuss in the next section, our approach can interpolate over a simplex from a simplex mesh, and thus is not restricted to a line segment, avoiding the chain trail pattern (as can be seen in the Figure 2g. We also use simplex mesh quality measures to spread the synthetic generated data over the instance space.

4. Delaunay Tessellation Oversampling—DTO-SMOTE

4.1. Simplex Geometry

In computation geometry, triangulation is one of the main primitives in two-dimensional space. It is often used to determine the vicinity of a point by forming triangles with other nearby points. For higher-dimensional spaces, the analogous to triangulation is a tessellation, and to a triangle is a simplex, respectively.

The Delaunay triangulation of a set of points \( P \) in the two-dimensional Euclidean space is defined as follows: the Delaunay triangulation \( DT(P) \) is a triangulation such that no point in \( P \) is inside the circumscribed spheres of other triangles in \( DT(P) \). The definition extends to higher-dimensional spaces, where no point in \( P \) is inside the circumscribed hyper-spheres of other simplices in \( DT(P) \).

The Delaunay tessellation has some interesting properties. In the plane, the triangulation is a maximum planar graph and completely divides the space by triangles (if no four points are co-linear). The center of the circumsphere of a Delaunay polyhedron is a vertex of the Voronoi cells. Figure 3 shows an example of a Voronoi diagram (in grey) and a Delaunay tessellation (in orange) for a two-dimensional space. The closest neighbor \( P_c \) to any point \( P_x \) lies on edge \( P_cP_x \) in the Delaunay triangulation, as the nearest neighbor graph is a subgraph of the Delaunay triangulation. This unique set of neighboring points defines the neighborhood of the point and represents a parameter-free definition of a point’s surroundings. The triangulation maximizes the minimum angle of the triangles, avoiding the occurrence of silver triangles. A silver triangle is a triangle with one or two extremely acute angles, hence with a long/thin shape. Intuitively, the Delaunay tessellation is a group of simplices that are most regularized in shape, compared to any other triangulation type. Delaunay tessellation may be used to approximate a manifold by a simple mesh [41,42], in computational geometry [43,44], and planning in automated driving [45], among others.

![Figure 3. Delaunay triangulation (orange) and Voronoi diagram.](image-url)
For a D-dimensional space, a simplex has $D + 1$ points. For instance, in the plane, a simplex has three points (a triangle), while, in a three-dimensional space, a simplex has four points (a tetrahedron).

Each point from a simplex is called a simplex’s vertex. Another simplex description is a convex hull of its vertices, or $S$ is the set of all points $p \in \mathbb{R}^D$ that can be expressed as a convex combination of its vertices $s_i$. $S$ boundaries consist of faces, which are simplices of lower dimensions composed by a subset of the simplex vertices. Furthermore, any point inside a simplex can be expressed as a combination of its vertices as $p = \sum \left| s_i \right|^2$. The vector $x$ is called the barycentric coordinates of the point $p$ concerning the simplex $S$ [46].

For a set $P$ of points, different algorithms can be used to compute $DT(P)$. A reasonably general approach for computing the tessellation in a $D$-dimensional space consists of converting the tessellation problem into finding the convex hull of $P$ in $(D + 1)$-dimensional space, by giving each point $p$ a new coordinate equal to $|p|^2$. The computational complexity of this approach is $O(n^{\left\lceil \frac{D}{2} \right\rceil})$. This paper uses the Delaunay tessellation algorithm implemented in the Python SciPy package [47].

4.2. Mesh Generation

Given a data set composed by a set of instances $(X, Y)$, where $X$ is an $n$-dimensional feature matrix, and $Y$ is the vector of class labels. Each $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,n}\}$ represents the feature values, and $y_i \in Y$ corresponds to the class value of instance $i$. The number of instances of the data set is $m$.

The first step to applying Delaunay Tetrahedral Oversampling (DTO-SMOTE) is to perform a dimensionality reduction. This operation reduces the $n$-dimensional features matrix $X$ to a three-dimensional space $X_{3d}^{projected}$. The reasons for using a dimensionality reduction are twofold: first, for $m$ points and $n$ dimensions, a generic mesh contains $O(m^{\lceil n/2 \rceil})$ simplices, and $n + 1$ points compose each simplex. Thus, we will have many simplices to interpolated, and many vertices will compose each simplex. Second, the computational complexity of mesh building will increase with large data dimensions. Hence, projecting the high-dimensional data to a lower dimension space is a way to make the mesh building more manageable. As we project into 3D space, the mesh is formed by tetrahedrons, and the interpolation process is carried out within four instances. However, the process is generic and other dimensions (including not applying a feature reduction). Any dimensionality reduction technique can be used for this reduction. However, it should be kept in mind that this dimensionality reduction is only used for the mesh generation, and the artificial instances are generated into the original space. Therefore, the classifiers’ training is also performed using the original space. A similar procedure was used in Reference [39] to immerse the data into a manifold space with lower dimension.

The next step consists of running the Delaunay tessellation algorithm using $X_{3d}^p$. In other words, we calculate $DT(P)$, where $P = X_{3d}^p$. The result of this process is a set of tetrahedrons, in which the vertices lie in $X_{3d}^p$. The method of synthetically generate a new instance corresponds to an interpolation using the vertices of a tetrahedron. To this end, our algorithm randomly selects a tetrahedron and a point $p$ inside this tetrahedron in barycentric coordinates and uses $p$ for interpolation. As there is a one-to-one mapping between the barycentric coordinates the Euclidean coordinates, the new instance’s generation corresponds to an interpolation of the points of the selected tetrahedron in the original $n$-dimensional $X$-space.

For selecting a tetrahedron for interpolation, our algorithm computes some quality indices for each tetrahedron. Furthermore, to consider the class neighborhood of the instance, we use the ratio of vertices of the tetrahedron associated with a minority class to weight the quality indices and associate them with probabilities. These probabilities are then used to randomly select tetrahedrons with replacement. (It is necessary because some data sets do not produce enough tetrahedrons to generate the adequate numbers of artificial samples.)

The generation of the point $p$ inside the tetrahedrons follows a Dirichlet Distribution. The algorithm adjusts the distribution parameters according to the class associated with a vertex for tetrahedrons with vertices belonging to majority class instances. This step assigns higher probabilities
to vertices and facets related to the minority class, similar to Reference [35,36]. These steps will be explained in detail in the next sections.

4.3. Tetrahedral Quality Evaluation

Given a Delaunay tessellation $DT(P)$, our approach selects a tetrahedron from $DT(P)$ to generate a new instance. In principle, we can randomly choose a tetrahedron of $DT(P)$. However, the tetrahedrons have different shapes. Furthermore, the tetrahedrons have different class labels associated with their vertices. Our method uses the shape and class distribution to guide the selection of the tetrahedrons.

The shapes of the tetrahedrons are generally associated with the quality of the tessellation [48]. Smaller tetrahedrons with a regular shape usually emerge from dense areas of the input space [14]. On the other hand, bigger tetrahedrons or very acute angles appear near boundaries.

Maur [48] proposes different tetrahedral quality measures. The main idea is how far a new tetrahedron is from the equilateral tetrahedron. The answer to this question, the following quality measures are proposed: Relative Volume, Radius Ratio, Solid Angle, Minimum Solid Angle, Maximum Solid Angle, Edge Ratio, and Aspect Ratio, explained next.

Relative Volume: The relative volume of the current tetrahedron is computed as its real volume divided by the value of maximal volume in the tessellation [48].

Radius Ratio: The radius ratio is the weighed ratio between the radius of the inscribed sphere ($r$) to the radius of the circumscribed sphere ($R$), as shown in Equation (18).

$$Rrt = \frac{3r}{R}. \quad (18)$$

Solid Angle: The solid angle is the area of a spherical triangle created on the unit sphere in which the center is in the tetrahedron vertex [48]. We compute the sum of four solid angles of the tetrahedron.

Minimum Solid Angle: This returns the minimum solid angle instead of their sum.

Maximum Solid Angle: This returns the maximum solid angle instead of their sum.

Edge Ratio: The edge ratio computes the ratio between the length of the most prolonged edge $E$ to the length of the shortest edge of the tetrahedron, as shown in Equation (19).

$$Ert = \frac{E}{e}. \quad (19)$$

Aspect Ratio: The aspect ration computes the ratio between the radius of the sphere that circumscribes the tetrahedron ($R$) to the length of the longest edge ($E$), as shown in Equation (20).

$$Asprt = \frac{R}{E}. \quad (20)$$

The selection of tetrahedron quality measure to use is a hyperparameter of our algorithm. For each tetrahedron from the tesselation, we compute the selected quality measure and weight it by the proportion of the minority class examples we want to generate the new synthetic instance. If all four vertices belong to the majority class, the tetrahedron quality measure is multiplied by zero and will not be taken into account for synthetic instance creation. On the other hand, if all four vertices belong to the minority class, the quality measure is multiplied by one. Tetrahedrons with mixed class vertices are weighed accordingly. For instance, if two out of the four vertices belong to the minority class, the calculated quality measure is multiplied by 0.5.

These weighed tetrahedron quality measures are then normalized, to sum up to 1 and represent a probability distribution. Therefore, to select tetrahedrons for interpolation, we use a weighted
4.4. Synthetic Instance Generation

Once a tetrahedron has been selected, the synthetic instance generation process requires a draw of a random point inside the tetrahedron. To this end, we use a Dirichlet distribution. A \( p \)-dimensional vector \( \alpha \) of positive reals parametrizes this distribution. A random draw of a Dirichlet distribution returns a \( p \)-dimensional vector \( d_1, \ldots, d_p \), where \( d_i \in (0, 1) \), and \( \sum^n d_i = 1 \). For the symmetric Dirichlet distribution, when \( \alpha_i = 1, \forall i \in 1..p \), each point inside the simplex has an equal probability of being chosen. On the other hand, when \( \alpha_i \) is a positive constant greater than 1, the distribution favors points near the simplex center. For nonsymmetric distribution, different values of \( \alpha_i \) can be used to control the likelihood of where points can be generated (see Table 1), where, for the sake of visualization, we show density plots of three-dimensional Dirichlet distribution for different values of \( \alpha \).

| \( \alpha \) | Dirichlet distribution |
|--------------|------------------------|
| \([1,1,1]\)  | ![Density Plot](image1) |
| \([3,9,3]\)  | ![Density Plot](image2) |
| \([5,5,5]\)  | ![Density Plot](image3) |
| \([7,7,7]\)  | ![Density Plot](image4) |
| \([7.5,7.5,7.5]\) | ![Density Plot](image5) |
| \([9,3,9]\)  | ![Density Plot](image6) |

Given a tetrahedron, we use the parameter vector to establish the likelihood region where the new point will be generated. If all the tetrahedron’s vertices are associated with minority class instances, then we use a symmetric Dirichlet distribution to draw the new instance. Therefore, depending on the value of \( \alpha \), we can draw a point uniformly in the simplex (\( \alpha_i = 1 \)) or closer to the tetrahedron barycenter (\( \alpha_i = c, c > 1 \)). However, suppose not all tetrahedron vertices are associated with the minority class. In that case, the distribution is asymmetric, where \( \alpha_i = 1 \) if the class of the vertex \( i \) is the majority, and \( \alpha_i = c, c > 1 \), where \( c \) is a hyperparameter that can be set by the user.

In summary, let \( \Delta_k = [a, b, c, d] \) be the index set of the instances a tetrahedron \( k \) selected as described in Section 4.3. We draw a random vector \( \lambda = \text{Dir}(\alpha) \), where \( \alpha \) is defined as
\( \alpha_i = \begin{cases} 1, & \text{if } y_i \notin \text{minority class} \\ c, & \text{if } y_i \in \text{minority class} \end{cases} \) \hspace{1cm} (21)

Then, a new instance

\[ x_{\text{new}} = X[\Delta_k]^T \cdot \lambda, \] \hspace{1cm} (22)

where \( x_{\text{new}} \) can be interpreted as the weighted mean of the instances that are vertices of the tetrahedron \( \Delta_k \), where the weights \( \lambda \) are drawn from the Dirichlet distribution.

### 4.5. Method Description

Here, we present steps to build DTO-SMOTE. Algorithm 1 presents the algorithm. The main steps are outlined next.

**Algorithm 1: DTO-SMOTE**

**Input:** \( \mathcal{D} = X \cup Y \) // Imbalanced Dataset  
1. tetristats // Tetrahedral Quality Parameters:  
   // one of 'area','volume','area_volume_ratio','edge_ratio','radius_ratio',  
   // 'aspect_ratio','max_solid_angle','min_solid_angle','solid_angle'  
2. \( \alpha \) // Dirichlet Parameters: \( [1,..,10] \in \mathbb{R}^+ \)  
3. \( G \) // Number of synthetic instances required for balancing

**Output:** Balanced Datasets

4. \( X_{m,3}^p \leftarrow \text{PCA}(X_{m,n}) \)  
5. \( \text{DG}(X_{m,3}^p) \leftarrow \text{Tessellation}(X_{m,3}^p) \)  
6. For all \( \Delta_i \in \text{DG}(X_{m,3}^p) \) do  
7. Calculate(tetraprops)  
8. Calculate(tetrastats)  
9. \( W_i \leftarrow \text{tetraprops} \cdot \text{tetrastats} \)  
10. For \( g \leftarrow 1 \) to \( G \) do  
11. \( \Delta_g \leftarrow \text{RandomChoosing}(\text{DG}(X_{m,3}^p),W) \)  
12. \( \lambda_g \leftarrow \text{Dirichlet}(\alpha) \)  
13. \( x_{\text{new}} \leftarrow X_{\Delta_g} \cdot \lambda_g \) // Equation (22)  
14. new_samples.add(\( x_{\text{new}} \))  
15. return new_samples

1. The algorithm takes as input an imbalanced data set, a data compression algorithm, some tetrahedron quality measure (Section 4.3), and the Dirichlet hyperparameter \( c \) (Section 4.4);  
2. Using the compression technique, reduce feature space to a three-dimensional space resulting in \( X_{3}^p \);  
3. Construct the Delaunay tessellation \( DT(X_{3}^p) \) using the compressed data set \( X_{3}^p \);  
4. For all simplex from \( DT(X_{3}^p) \), compute a simplex quality measure (Section 4.3);  
5. After that, we calculate weights for all simplex in this way: \( W = (\text{tetraprops} \ast \text{tetrastats}) \) where \( \text{tetraprops} \) is the vertex’s proportion of instances belonging to the minority class in the simplex and \( \text{tetrastats} \) is the quality index calculated for the simplex, according to quality measure selected; The weights are normalized to sum 1, for representing probabilities.  
6. Randomly choose with replacement a simplex (tetrahedron) from \( DT(X_{3}^p) \), according to the probabilities calculated from step 5.
7. Once a simplex was selected, generate a new sample $x_{new}$ using a Dirichlet distribution, as described in Section 4.4.

8. Steps 6 and 7 are repeated until the number of samples in Minority Class matches the number of instances of the Majority Class.

5. Experiments and Results

This section describes the experimental evaluation carried out for evaluating our new preprocessing techniques. The experimental protocol is depicted in Figure 4. Experiments are conducted using 5-fold cross-validation. We opt to execute 5-fold cross-validation instead of 10-fold cross-validation as in imbalanced data sets because using 10-fold may lead to a test set with very few instances from the minority class. As suggested in Reference [49], preprocessing methods are only applied to the training set. Considering the random nature of DTO-SMOTE and other oversampling methods to build artificial data, we repeat the entire process three times, i.e., we ran $3 \times 5$-fold cross-validation.

![Figure 4. Delaunay Tessellation Oversampling Synthetic Minority Over-sampling Technique (DTO-SMOTE) flowchart.](image)

We compare our method with the preprocessing techniques: SMOTE [13], SMOTESVM [36], BORDERLINE SMOTE 1 [35], BORDERLINE SMOTE 2 [35], and GEOMETRIC SMOTE [38]. All these methods are executed with the default parameters. For running SMOTE, SMOTESVM, BORDERLINE SMOTE 1, and BORDERLINE SMOTE 2, we use the imbalanced-learn package [22]. For GEOMETRIC...
SMOTE, we use the implementation of the author, available at https://geometric-smote.readthedocs.io/. Furthermore, we also compare results without any preprocessing. Table 2 presents the alias used for showing the results.

To evaluate our methods’ effectiveness, we compare the original and oversampled data sets with eight different learning algorithms, as presented in Table 3. All classifiers are available in the SciKit–Learn package and Intel Python Distribution, at http://scikit-learn.org/ and https://software.intel.com/en-us/distribution-for-python, respectively, and were used with default parameters, as described in Table 3.

| Oversampling Technique | Alias       | Reference |
|------------------------|-------------|-----------|
| SMOTE                  | SMOTE       | [21]      |
| BORDERLINE SMOTE 1     | BORDINARY1  | [35]      |
| BORDERLINE SMOTE 2     | BORDINARY2  | [35]      |
| SMOTE SVM              | SMOTESVM    | [36]      |
| GEOMETRIC SMOTE        | GEOSMOTE    | [38]      |
| No preprocessing        | Original    |           |

Table 3. Machine learning algorithms.

| Algorithm                        | Alias | Parameters | Reference |
|----------------------------------|-------|------------|-----------|
| Random Forest Classifier         | RF    | n_estimators=100 | [50]      |
| k-Neighbours Classifier          | KNN   | n_neighbors=5  | [51]      |
| Decision Tree Classifier         | DTREE | criterion='gini' | [52]      |
| Logistic Regression              | LRG   | penalty='l2',C=1.0 | [53]      |
| AdaBoost Classifier              | ABC   | n_estimators=50 | [54]      |
| MultiLayer Perceptron            | MLP   | max_iter=500  | [55]      |
| Support Vector Machine           | SVM   | probability=True | [56]      |
| Stochastic Gradient Descent Classifier | SGD  | loss="hinge",penalty="l2",max_iter=500 | [57]      |

Experiments were performed with 61 bi-class data sets. Table 4 show the characteristics of the bi-class Data sets used in our experiments. For each data set, the tables present the Imbalance Ratio (IR), the number of samples (Samples), and the number of features (Features).

We use the Principal Component Analysis (PCA) algorithm to compress the data sets before applying the Delaunay Tessellation. This choice was motivated due to its simplicity and extensive use. However, any method that compresses the data to a lower dimension can be used instead of PCA. Furthermore, PCA was also used in Reference [39] for dimensionality reduction for manifold embedding.

Source Code for DTO-SMOTE can be found at https://github.com/carvalhoamc/DTOSMOTE. Furthermore, all results are available at https://github.com/carvalhoamc/DTO-SMOTE and scripts to reproduce the analysis are at https://github.com/carvalhoamc/dtosmote_performance.
Table 4. Bi-class data sets.

| Data sets | IR  | Inst. | Feat. |
|-----------|-----|-------|-------|
| page-blocks0 | 8.79 | 5472  | 10    |
| ecoli3 | 8.6 | 336  | 7     |
| Spectrometer | 11.0 | 531  | 93    |
| yeast3 | 8.1 | 1484 | 8     |
| glass6 | 6.38 | 214  | 9     |
| ecoli2 | 5.46 | 336  | 7     |
| new-thyroid2 | 5.14 | 215  | 5     |
| new-thyroid1 | 5.14 | 215  | 5     |
| ecoli1 | 3.36 | 336  | 7     |
| vehicle0 | 3.25 | 846  | 18    |
| glass-0-1-2-3 vs 4-5-6 | 3.2 | 214  | 9     |
| vehicle3 | 2.99 | 846  | 18    |
| vehicle1 | 2.9 | 846  | 18    |
| vehicle2 | 2.88 | 846  | 18    |
| yeast1 | 2.46 | 1484 | 8     |
| glass0 | 2.06 | 214  | 9     |
| iris0 | 2  | 150  | 4     |
| pima | 1.87 | 768  | 8     |
| glass1 | 1.82 | 214  | 9     |
| yeast6 | 41.4 | 1484 | 8     |
| Pen Digits | 9.4 | 10,992 | 16    |
| yeast5 | 32.73 | 1484 | 8     |
| yeast-2 vs 8 | 23.1 | 482  | 8     |
| page-blocks-1-3 vs 4 | 15.86 | 472  | 10    |
| ecoli4 | 15.8 | 336  | 7     |
| shuttle-c0-vs-c4 | 13.87 | 1829 | 9     |
| vowel0 | 9.98 | 988  | 13    |
| yeast-0-5-6-7-9 vs 4 | 9.35 | 528  | 8     |
| yeast-2 vs 4 | 9.08 | 514  | 8     |
| Wine Quality | 26 | 4898 | 11    |
| Ecoli | 8.6 | 336  | 7     |
| OIL | 22 | 937  | 49    |
| appendicitis | 4.04 | 106 | 8     |
| autoUniv-au1-1000 | 2.86 | 1000 | 20    |
| climate-simulation-craches | 10.73 | 540  | 20    |
| colon32 | 1.81 | 62   | 32    |
| fertility-diagnosis | 7.3 | 100  | 9     |
| habermans-survival | 2.78 | 306  | 3     |
| indian-liver-patient | 2.49 | 583  | 10    |
| ionosphere | 2.02 | 351  | 33    |
| lsvt-voice-rehabilitation | 2.0 | 126  | 310   |
| ozone-eighthr | 14.83 | 2533 | 72    |
| ozone-onehr | 33.73 | 2535 | 72    |
| parkinsons | 3.06 | 194  | 22    |
| phoneme | 2.4 | 5413 | 5     |
| pima-indians-diabetes | 1.86 | 767  | 8     |
| planning-relax | 2.5 | 181  | 12    |
| qsar-biodegradation | 1.96 | 1054 | 41    |
| saheart | 1.88 | 461  | 9     |
| seismic-bumps | 14.2 | 2583 | 18    |
| spambase | 1.53 | 4600 | 57    |
| spectf-heart | 2.67 | 348  | 44    |
| german-credit | 2.33 | 999  | 20    |
| german-credit-numeric | 2.33 | 999  | 24    |
| thoracic-surgery | 5.71 | 469  | 16    |
| thyroid-hypothyroid | 19.94 | 3162 | 25    |
| thyroid-sick-euthyroid | 9.8 | 3163 | 25    |
| vertebra-column-2c | 2.1 | 309  | 6     |
| wdbc | 1.68 | 568  | 30    |
| wholesale-channel | 2.1 | 439  | 7     |
| wilt | 17.54 | 4838 | 5     |
5.1. Influence of Parameters

Our method has two parameters: the tetrahedral quality metric and the Dirichlet parameter $\alpha$. The quality metric influences the selection of tetrahedral for interpolation, while the $\alpha$ influences where the interpolation will occur in a particular tetrahedral. We have experimented with all the tetrahedral quality metrics described in Section 4.3. For the parameter $\alpha$, we experiment with the parameter ranging from 1 to 9.5 with step size 0.5. In total, we made $3 \times 18 \times 61 = 3294$ runs for bi-class data sets for each of the eight classifiers used in the experiments.

Figure 5 depicts the influence of this parameter. The graphs show the variation in rank as a function of the $\alpha$ parameter for each tetrahedral quality measure, averaged over all the data sets. Results are homogeneous among the different metrics (AUC, GEO, and IBA for binary data sets). As can be seen, the quality measure based on the solid angle produces better results. Furthermore, a larger than four also produces good average results.

With these results, we choose the default values for $\alpha$ and geometry for the DTO-SMOTE algorithm. The selected values $\alpha$ for bi-class is 7, and the quality measure set is the solid angle. Therefore, we use the solid angle as the tetrahedron quality measure and $\alpha = 7$ for bi-class problems for comparison with other SMOTE methods.

![Figure 5. Bi-class DTO-SMOTE rank by Dirichlet alpha.](image)

5.2. Experimental Results for bi-class Data Sets

For binary class data sets, we evaluate three performance measures: Area Under the ROC curve (AUC), Geometric Mean (GEO), and Index of Balanced Accuracy (IBA). In this work, we adopted the $\alpha$ parameter for IBA to 0, as recommended by the authors.

Due to space limitations, we did not include in the paper numerical results for each data set. However, all the results are publicly available at https://github.com/carvalhoamc/DTO-SMOTE. To draw general conclusions about the our method’s performance, we present average rank performance and statistical analysis.

To evaluate whether the differences among methods are statistically significant, we use a non-parametric Friedman multiple comparison testing (Table 5). The Friedman test is the non-parametric equivalent of the two-way ANOVA. Under the null hypothesis, it states that all the algorithms are equivalent, so rejecting this hypothesis implies significant differences among all algorithms. When the Friedman test rejects the null hypothesis, we can proceed with a post-hoc test to detect the significant differences among the methods. For this, we used Shaffer post-hoc multiple comparisons for controlling the family-wise error.
In terms of AUC (Table 6), DTO-SMOTE performed better in terms of global rank in 7 of the eight classification algorithms (Table 3). Only the KNN algorithm, our method did not obtain the best average performance with DTO-SMOTE. However, for KNN, DTO-SMOTE was in second place, little after the SMOTE algorithm. Regarding the critical differences diagram (Table 5) DTO-SMOTE is in the first place in 7 of 8 algorithms, significantly outperforming the data set without treatment.

In terms of GEO (Table 7), DTO-SMOTE performed better in terms of rank with ABC, DTREE, LRG, and RF (Table 3). There was a tie between DTO-SMOTE and SMOTE for the SVM algorithm. For KNN, MLP, and SGD, SMOTE presented better performance. The winner in terms of global rank for bi-class problems under the GEO measure was the DTO-SMOTE algorithm. Again, DTO-SMOTE significantly outperforms the data set without treatment in terms of GEO Table 5.

Regarding IBA (Table 8), DTO-SMOTE has a better average rank with ABC, LRG, RF and SVM classifiers (Table 3). With DTREE, KNN, MLP, and SGD, SMOTE had the best rank performance. In terms of average rank, SMOTE is in first place, followed by DTO-SMOTE in second place. DTO-SMOTE is significantly better than no treating, as can be seen in Table 5.

In general, our method (DTO-SMOTE) showed a better average performance when used as a preprocessing technique for several classifiers. For all learning algorithms, it was ranked first or second in at least one performance measure for all eight learning algorithms. Furthermore, it has better overall rank performance in terms of AUC and GEO, and second (with a lower standard deviation) in IBA. These results are a strong indicator of the utility of our proposed method for imbalanced bi-class classification problems. Table 5 shows all the pairwise comparisons of the data methods evaluated in bi-class data sets in terms of AUC, GEO, and IBA. Although there are a few statistical differences between our proposed method (DTO-SMOTE) and the other SMOTE variants, our approach is ranked firstly with ABC, RF, LRG, and RF for all three performance measures AUC, GEO, and IBA. It is ranked
firstly with DTREE for AUC and GEO, and it is ranked firstly with SVM for AUC and IBA. For KNN, it is ranked in second place for all three measures and, for MLP and SGD, it is ranked in second place for GEO and IBA. Furthermore, DTO-SMOTE is statistically better than Original for all classifiers.

Table 6. Area Under the ROC curve (AUC) rank (solid angle and $\alpha = 7.0$); see Table 5.

| ALGORITHM | ORIGINAL | SMOTE | SMOTESVM | BORDERLINE1 | BORDERLINE2 | GEOSMOTE | DTO-SMOTE |
|-----------|----------|-------|----------|-------------|-------------|----------|-----------|
| ABC       | 6.36     | 3.27  | 3.42     | 3.59        | 4.28        | 3.81     | 3.26      |
| DTREE     | 5.6      | 3.43  | 3.41     | 3.96        | 4.4         | 3.79     | 3.4       |
| KNN       | 5.87     | 3.11  | 3.7      | 3.65        | 4.24        | 3.8      | 3.62      |
| LRG       | 6.43     | 3.32  | 3.69     | 3.69        | 4.42        | 3.36     | 3.08      |
| MLP       | 5.88     | 3.46  | 3.47     | 4.1         | 4.08        | 3.68     | 3.32      |
| RF        | 5.7      | 3.34  | 3.31     | 3.78        | 4.19        | 4.45     | 3.24      |
| SGD       | 5.64     | 3.53  | 3.58     | 3.84        | 3.92        | 4.02     | 3.47      |
| SVM       | 6.05     | 3.35  | 3.54     | 3.86        | 4.51        | 3.41     | 3.28      |
| average   | 5.94     | 3.35  | 3.52     | 3.81        | 4.25        | 3.79     | 3.34      |
| std       | 0.63     | 0.29  | 0.21     | 0.25        | 0.37        | 0.4      | 0.17      |

Table 7. Geometric Mean (GEO) rank (solid angle and $\alpha = 7.0$); see Table 5.

| ALGORITHM | ORIGINAL | SMOTE | SMOTESVM | BORDERLINE1 | BORDERLINE2 | GEOSMOTE | DTO-SMOTE |
|-----------|----------|-------|----------|-------------|-------------|----------|-----------|
| ABC       | 6.02     | 3.54  | 3.34     | 3.8         | 4.31        | 3.81     | 3.16      |
| DTREE     | 5.79     | 3.41  | 3.57     | 4.05        | 3.94        | 3.9      | 3.33      |
| KNN       | 5.93     | 3.35  | 3.71     | 3.69        | 3.78        | 3.94     | 3.59      |
| LRG       | 6.76     | 3.19  | 3.63     | 3.68        | 4.47        | 3.17     | 3.09      |
| MLP       | 6.44     | 3.23  | 3.67     | 3.71        | 4.17        | 3.52     | 3.26      |
| RF        | 5.75     | 3.31  | 3.6      | 4.04        | 4.01        | 4.16     | 3.13      |
| SGD       | 6.35     | 3.19  | 3.56     | 3.61        | 4.22        | 3.67     | 3.4       |
| SVM       | 6.17     | 3.2   | 3.65     | 3.69        | 4.18        | 3.91     | 3.2       |
| average   | 6.15     | 3.3   | 3.59     | 3.79        | 4.14        | 3.76     | 3.27      |
| std       | 0.59     | 0.31  | 0.19     | 0.26        | 0.38        | 0.46     | 0.17      |

Table 8. Index of Balanced Accuracy (IBA) rank (solid angle and $\alpha = 7.0$); see Table 5.

| ALGORITHM | ORIGINAL | SMOTE | SMOTESVM | BORDERLINE1 | BORDERLINE2 | GEOSMOTE | DTO-SMOTE |
|-----------|----------|-------|----------|-------------|-------------|----------|-----------|
| ABC       | 6.07     | 3.28  | 3.38     | 3.65        | 4.39        | 4.1      | 3.14      |
| DTREE     | 6.2      | 3.19  | 3.5      | 3.81        | 4.37        | 3.64     | 3.29      |
| KNN       | 5.83     | 3.11  | 3.76     | 3.72        | 4.07        | 3.67     | 3.65      |
| LRG       | 6.27     | 3.16  | 3.7      | 3.96        | 4.57        | 3.28     | 3.06      |
| MLP       | 6.5      | 3.21  | 3.4      | 3.91        | 3.95        | 3.68     | 3.36      |
| RF        | 6.0      | 3.34  | 3.52     | 3.69        | 4.93        | 4.29     | 3.24      |
| SGD       | 6.14     | 3.2   | 3.55     | 3.79        | 4.01        | 3.89     | 3.41      |
| SVM       | 5.93     | 3.47  | 3.61     | 3.86        | 4.57        | 3.37     | 3.2       |
| average   | 6.12     | 3.24  | 3.55     | 3.8         | 4.23        | 3.77     | 3.29      |
| std       | 0.6      | 0.33  | 0.18     | 0.24        | 0.38        | 0.46     | 0.19      |

6. Conclusions

This paper presents DTO-SMOTE, an oversampling algorithm that generates synthetic instances based on the Delaunay tessellation. DTO-SMOTE is an evolution of our previous work, presented in Reference [15]. In that study, we evaluated an initial version of our preprocessing method considering only the KNN learning algorithm. Furthermore, our previous version does not consider the shape of the tetrahedrons and uses their barycenters for interpolation. This new version uses tetrahedron quality indices to select tetrahedrons, and a Dirichlet distribution to randomly choose a point for interpolation. We conduct a sizeable experimental study comparing our method with five SMOTE variations evaluated with eight learning algorithms, 61 binary data sets, and three different performance measures.

Results indicate a better average performance when the method is used for class imbalanced classification problems. Besides that, we plan to study the imbalanced data set complexity by analyzing the simplex mesh and its quality indexes. The main advantage is that DTO-SMOTE proved to be more efficient than the other oversampling algorithms on average, according to the results presented. In comparison to the other algorithms, DTO-SMOTE can be a little slower as it needs to create a
three-dimensional mesh. In future work, we intend to assess whether the refinement of the mesh can lead to better results in terms of performance in the classification.

Our goal is to better understand the data set characteristics that lead to an increase in performance, as the simplex geometry could be linked to the data’s local density [14]. We also plan to investigate the SMOTE bias theoretically in the future. This procedure could reduce efforts in selecting a proper method and parameters for specific data sets. Our implementation, and all results and scripts for analysis, are publicly available on the Internet.

**Author Contributions:** A.M.d.C. and R.C.P. conceived the main idea of the algorithm and designed the tests; R.C.P. supervised the design of the system; A.M.d.C. designed the system and ran the experiments and collected data; All authors contributed in writing the original draft preparation. All authors have read and agreed to the published version of the manuscript.

**Funding:** This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES)-Finance Code 001.

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**

1. Prati, R.C.; Battista, G.E.A.P.A.; Silva, D.F. Class imbalance revisited: A new experimental setup to assess the performance of treatment methods. *Knowl. Inf. Syst.* 2015, 45, 247–270.
2. Krawczyk, B.; Galceran, M.; Jeleń, L.; Herrera, F. Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. *Appl. Soft Comput. J.* 2016, 38, 714–726, doi:10.1016/j.asoc.2015.08.060.
3. Troncoso, A.; Ribera, P.; Asencio-Cortés, G.; Vega, I.; Gallego, D. Imbalanced classification techniques for monsoon forecasting based on a new climatic time series. *Environ. Model. Softw.* 2018, 106, 48–56, doi:10.1016/j.envsoft.2017.11.024.
4. Yan, B.; Han, G. LA-GRU: Building Combined Intrusion Detection Model Based on Imbalanced Learning and Gated Recurrent Unit Neural Network. *Secur. Commun. Netw.* 2018, 2018, doi:10.1155/2018/6026878.
5. Farias, D.J.H.; Prati, R.; Herrera, F.; Rosso, P. Irony detection in Twitter with imbalanced class distributions. *J. Intell. Fuzzy Syst.* 2020, 39, 2147–2163.
6. Huang, X.; Zhang, C.Z.; Yuan, J. Predicting Extreme Financial Risks on Imbalanced Dataset: A Combined Kernel FCM and Kernel SMOTE Based SVM Classifier. *Comput. Econ.* 2020, 56, 187–216, doi:10.1007/s10614-020-09975-3.
7. Roumani, Y.F.; Nwankpa, J.K.; Tanniru, M. Predicting firm failure in the software industry. *Artif. Intell. Rev.* 2020, 53, 4161–4182, doi:10.1007/s10462-019-09789-2.
8. Zhang, X.; Li, Y.; Kotagiri, R.; Wu, L.; Tari, Z.; Cheriet, M. KRNN: K Rare-class Nearest Neighbour classification. *Pattern Recognit.* 2017, 62, 33–44.
9. Sawangarreerak, S.; Thanathamthee, P. Random Forest with Sampling Techniques for Handling Imbalanced Prediction of University Student Depression. *Information* 2020, 11, 519.
10. Oksuz, K.; Cam, B.C.; Kalkan, S.; Akbas, E. Imbalance problems in object detection: A review. *IEEE Trans. Pattern Anal. Mach. Intell.* 2020, doi:10.1109/TPAMI.2020.2981890.
11. Fiorentini, N.; Losa, M. Handling imbalanced data in road crash severity prediction by machine learning algorithms. *Infrastructures* 2020, 5, 61.
12. Patel, H.; Singh Rajput, D.; Thippa Reddy, G.; Ivendi, C.; Kashif Bashir, A.; Jo, O. A review on classification of imbalanced data for wireless sensor networks. *Int. J. Distrib. Sens. Netw.* 2020, 16, 1550147720916404.
13. Chawla, N.; Bowyer, K.; Hall, L.; Kegelmeyer, W. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* 2002, 16, 321–357.
14. Schaap, W.E.; van de Weygaert, R. Continuous fields and discrete samples: Reconstruction through Delaunay tessellations. *Astron. Astrophys.* 2000, 363, L29–L32.
15. Carvalho, A.M.D.; Prati, R.C. Improving kNN classification under Unbalanced Data. A New Geometric Oversampling Approach. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–6.
16. Fernández, A.; García, S.; Galceran, M.; Prati, R.C.; Krawczyk, B.; Herrera, F. *Learning from Imbalanced Data Sets*; Springer International Publishing: Cham, Switzerland, 2018.
17. Japkowicz, N.; Shah, M. Evaluating Learning Algorithms: A Classification Perspective; Cambridge University Press: Cambridge, UK, 2011.

18. García, V.; Sánchez, J.; Mollineda, R. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. Knowl.-Based Syst. 2012, 25, 13–21.

19. Prati, R.C.; Batista, G.E.A.P.A.; Monard, M.C. A Survey on Graphical Methods for Classification Predictive Performance Evaluation. IEEE Trans. Knowl. Data Eng. 2011, 23, 1601–1618.

20. Haixiang, G.; Yijing, L.; Shang, J.; Mingyun, G.; Yuanyue, H.; Bing, G. Learning from class-imbalanced data: Review of methods and applications. Expert Syst. Appl. 2017, 73, 220–239.

21. Fernández, A.; García, S.; Herrera, F.; Chawla, N.V. SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. J. Artif. Intell. Res. 2018, 61, 863–905.

22. Lemaître, G.; Nogueira, F.; Aridas, C.K. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. J. Mach. Learn. Res. 2017, 18, 1–5.

23. García, S.; Herrera, F. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. Evol. Comput. 2009, 17, 275–306.

24. Yu, H.; Ni, J.; Zhao, J. ACOSampling: An ant colony optimization-based undersampling method for classifying imbalanced DNA microarray data. Neurocomputing 2013, 101, 309–318, doi:10.1016/j.neucom.2012.08.018.

25. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. IEEE Comput. Intell. Mag. 2006, 1, 28–39.

26. Sun, Y.; Castellano, C.G.; Robinson, M.; Adams, R.; Rust, A.G.; Davey, N. Using pre & post-processing methods to improve binding site predictions. Pattern Recognit. 2009, 42, 1949–1958.

27. García, S.; Herrera, F. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. Evol. Comput. 2009, 17, 275–306.

28. Ramentol, E.; Caballero, Y.; Bello, R.; Herrera, F. SMOTE-RSB*: A hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. Knowl. Inf. Syst. 2012, 33, 245–265.

29. Sáez, J.A.; Luengo, J.; Stefanowski, J.; Herrera, F. SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. Inf. Sci. 2015, 291, 184–203.

30. Guo, H.; Zhou, J.; Wu, C.A. Imbalanced learning based on data-partition and SMOTE. Information 2018, 9, 238.

31. Fernández, A.; García, S.; Galar, M.; Prati, R.C.; Krawczyk, B.; Herrera, F. Cost-Sensitive Learning. In Learning from Imbalanced Data Sets; Springer: New York, NY, USA, 2018; pp. 63–78.

32. Galar, M.; Fernández, A.; Barrenechea, E.; Sola, H.; Herrera, F. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. Syst. Man Cybern. Part C Appl. Rev. IEEE Trans. 2012, 42, 463–484, doi:10.1109/TSMCC.2011.2161285.

33. Bellinger, C.; Drummond, C.; Japkowicz, N. Manifold-based synthetic oversampling with manifold conformance estimation. Mach. Learn. 2018, 107, 605–637.

34. Doñate, D.; Atiya, A.F. A Comprehensive Analysis of Synthetic Minority Oversampling TEchnique (SMOTE) for Handling Class Imbalance. Inf. Sci. 2019, 505, 32–64.

35. Gao, Z.; Yu, Z.; Holst, M. Feature-preserving surface mesh smoothing via suboptimal Delaunay triangulation. Graph. Model. 2013, 75, 23–38.
42. Samat, A.; Gamba, P.; Liu, S.; Du, P.; Abuduwalli, J. Jointly Informative and Manifold Structure Representative Sampling Based Active Learning for Remote Sensing Image Classification. *IEEE Trans. Geosci. Remote. Sens.* 2016, 54, 6803–6817.

43. Kolluri, R.; Shewchuk, J.R.; O’Brien, J.F. Spectral surface reconstruction from noisy point clouds. In Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, Nice, France, July 2004; pp. 11–21.

44. De Kok, T.; Van Kreveld, M.; Löffler, M. Generating realistic terrains with higher-order Delaunay triangulations. *Comput. Geom.* 2007, 36, 52–65.

45. Anderson, S.J.; Karumanchi, S.B.; Iagnemma, K. Constraint-based planning and control for safe, semi-autonomous operation of vehicles. In Proceedings of the 2012 IEEE Intelligent Vehicles Symposium (IV), Madrid, Spain, 3–7 June 2012; pp. 383–388.

46. Devriendt, K.; Van Mieghem, P. The simplex geometry of graphs. *J. Complex Netw.* 2019, 7, 469–490.

47. Jones, E.; Oliphant, T.; Peterson, P. SciPy: Open Source Scientific Tools for Python. 2001. Available online: [https://www.scipy.org/](https://www.scipy.org/) (accessed on 5 November 2020).

48. Maur, P. Delaunay Triangulation in 3D. Ph.D. Thesis, University of West Bohemia in Pilsen, Pilsen, Czech Republic, 2002.

49. Santos, M.S.; Soares, J.P.; Abreu, P.H.; Araujo, H.; Santos, J. Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches [research frontier]. *IEEE Comput. Intell. Mag.* 2018, 13, 59–76.

50. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Mach. Learn.* 2006, 63, 3–42.

51. Abou-Moustafa, K.; Ferrie, F.P. Local generalized quadratic distance metrics: Application to the k-nearest neighbors classifier. *Adv. Data Anal. Classif.* 2018, 12, 341–363.

52. Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. Classification and regression trees. *Classif. Regres. Trees* 2017, 1, 1–358.

53. Pearlmutter, B.A. Fast Exact Multiplication by the Hessian. *Neural Comput.* 1994, 6, 147–160.

54. Utkin, L.V.; Zhuk, Y.A. Robust boosting classification models with local sets of probability distributions. *Knowl.-Based Syst.* 2014, 61, 59–75.

55. Shen, H. Towards a Mathematical Understanding of the Difficulty in Learning with Feedforward Neural Networks. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 811–820.

56. Chang, C.C.; Lin, C.J.; Tieleman, T. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol. (TIST)* 2008, 307, 1–39.

57. Zhang, T.; Damerau, F.; Johnson, D. Text chunking based on a generalization of Winnow. *J. Mach. Learn. Res.* 2002, 2, 615–637, doi:10.1162/153244302320884560.

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).